

Compact Neural Appearance Models for Efficient Gaussian Splatting

Florian Hahlbohm¹ Jorge Condor² Linus Franke³ Martin Eisemann¹ Marcus Magnor^{1,4}

¹TU Braunschweig ²Università della Svizzera italiana ³University of Würzburg ⁴University of New Mexico

<https://fhahlbohm.github.io/efficient-gaussian-appearance>

Abstract

Explicit primitive-based radiance fields such as 3D Gaussian Splatting typically model view-dependent appearance using low-order spherical harmonics (SH). Although efficient to evaluate, SH coefficients dominate per-primitive storage and memory traffic, while their band-limited basis restricts angular detail. We present a thorough, end-to-end comparison of SH and recent spherical appearance models and introduce an implicit alternative that decodes compact per-primitive latent codes using a tiny shared MLP. We integrate all models into the same optimized pipeline, fusing their forward and backward passes into a differentiable CUDA rasterizer and provide a portable WebGL viewer for laptop and mobile GPUs. Our evaluation across reconstruction quality, memory use, and optimization and rendering performance shows that recent spherical models offer the strongest overall quality–efficiency trade-off. Our neural representation is the most compact model evaluated and, compared to third-degree SH, reduces the per-primitive appearance footprint from 192 to 28 bytes, accelerates optimization by 1.3×, while improving reconstruction quality. We further analyze how appearance parametrization shapes optimization, identifying differences in recovered geometry and the tendency of expressive models to absorb non-static scene content. Together, our framework and analysis provide practical guidance for replacing SH beyond what image metrics alone can capture.

1. Introduction

Radiance fields have driven rapid progress in 3D reconstruction and novel view synthesis, with applications in robotics, virtual and augmented reality, and interactive graphics. Early methods represented scenes primarily with neural networks [41], while subsequent work increasingly adopted explicit representations for more efficient optimization and rendering [6, 8, 18, 20, 25, 45]. This trend culminated in 3D Gaussian Splatting (3DGS) [31], whose explicit anisotropic primitives and differentiable rasterizer combine high reconstruction quality with real-time rendering. 3DGS has since

become the blueprint for many primitive-based radiance field methods [12, 19, 26, 28, 29, 38, 57].

Across these representations, modeling view-dependent appearance remains essential for high-quality novel view synthesis. 3DGS uses low-order spherical harmonics (SH), which are inexpensive to evaluate, straightforward to optimize, and support a coarse-to-fine schedule that suppresses view dependence early in training [31]. However, third-degree SH requires 48 coefficients per primitive, constituting most parameters stored by each Gaussian [3]. Appearance thus dominates model size, optimizer state, and memory traffic during optimization and rendering. At high primitive counts, updating SH coefficients alone accounts for up to 50% of optimization time [22]. Moreover, low-degree SH are band-limited and therefore unable to represent high-frequency angular appearance.

Recent work has proposed replacing SH with more expressive spherical models based on beta kernels [36], soft Voronoi diagrams [14], as well as anisotropic spherical Gaussian or Gabor kernels [40]. Although these methods report better quality at equal or lower parameter counts, their practical trade-offs remain difficult to assess without a controlled evaluation under a common optimization protocol, consistently optimized end-to-end implementations, and measurements across desktop and mobile GPUs. Existing evaluations also focus primarily on image metrics, which reveal little about how appearance parametrization affects the geometry recovered during optimization.

We therefore present a controlled, end-to-end comparison of view-dependent appearance models for Gaussian splatting. We integrate SH and recent spherical alternatives into Faster-GS [22], using runtime code generation [43] to JIT-compile each model’s forward and backward passes into its differentiable rasterizer, retaining extensibility while avoiding separate kernel launches and global memory round-trips. We further implement all evaluated models in a portable WebGL viewer, extending the comparison to laptop and mobile GPUs, where appearance evaluation faces different computational and memory constraints.

We also introduce a basis-free alternative that decodes compact per-primitive latent codes using a tiny direction-

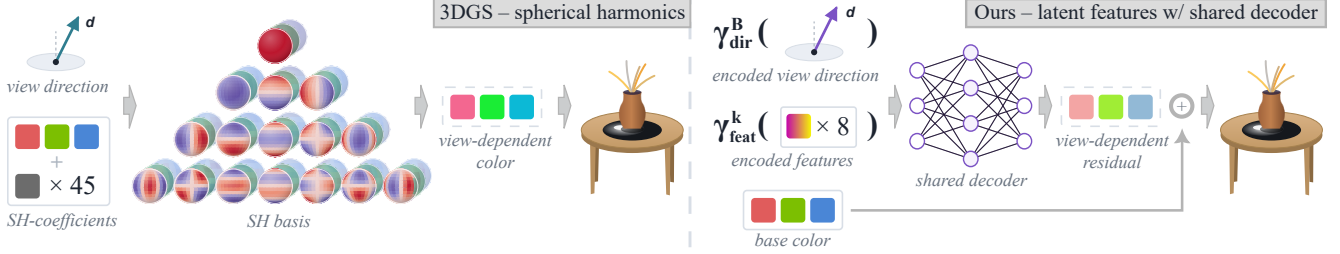


Figure 1. 3DGS [31] employs third-degree spherical harmonics, totaling 45 view-dependent appearance parameters per Gaussian (left). Our model (right) achieves faster training/rendering with similar quality using only 8 features per Gaussian and a tiny MLP with shared weights.

conditioned MLP shared across the scene (*cf.* Fig. 1). Unlike explicit spherical models, the decoder is not restricted to a predefined basis. This approach has received little attention in explicit radiance fields, likely due to the belief that network evaluation is too expensive in real-time rendering pipelines. We show that this assumption no longer holds on modern consumer GPUs: the fused decoder executes efficiently on Tensor Cores, while on mobile GPUs, inexpensive FMA operations and the cache locality of weights shared across primitives yield competitive rendering performance.

Our evaluation reveals distinct trade-offs among the considered models: recent spherical representations provide strong overall quality and efficiency compared to classical spherical harmonics. Our neural representation is, on the other hand, the most compact, and comparable to them in speed and quality. We further analyze how appearance parametrization shapes optimization, identifying differences in recovered geometry and a tendency of expressive models to absorb non-static scene content. In summary, we contribute:

- a controlled comparison of SH and recent parametric appearance models in a common optimized pipeline, spanning quality, memory use, optimization, and rendering,
- a compact, basis-free neural appearance model based on per-primitive latent codes and a shared MLP, fused into the differentiable rasterizer, and
- an analysis of how appearance parametrization shapes optimization, including its effects on recovered geometry and the absorption of non-static scene content.

Our full implementation is publicly available. Its appearance models and neural architecture remain configurable beyond the settings evaluated in this paper, facilitating future work on alternative representations and platform-specific quality-efficiency trade-offs.

2. Related Work

Radiance Field Representations. Radiance field methods span a spectrum from implicit neural fields to fully explicit scene representations. NeRF [41] and its derivatives [5] use MLPs to represent density and view-dependent radiance, while hybrid approaches accelerate optimization

and rendering using explicit spatial structures such as hash grids [6, 45], voxel grids [53], or factorized tensors [8]. Fully explicit radiance fields eliminate the neural scene representation entirely [18, 31]. In particular, 3D Gaussian Splatting (3DGS) [31] established differentiable rasterization of explicit primitives as an efficient paradigm for reconstruction and rendering, inspiring variants based on surfels [29], triangles [27, 28], convexes [26], tetrahedra [39, 57], and Voronoi cells [19], as well as ray-traced alternatives [11, 38, 42]. Complementary compression methods reduce the storage of these representations by pruning, quantizing, or vector-quantizing Gaussian parameters, including their SH coefficients [3]. Our work instead studies which appearance parametrization should be used in the first place, retaining the explicit geometry and rasterization pipeline of 3DGS.

View-Dependent Appearance. Explicit radiance fields commonly model angular appearance using spherical functions. Plenoxels [18] and 3DGS [31] employ spherical harmonics (SH), whose inexpensive evaluation and simple coarse-to-fine optimization have made them the de facto standard. Recent work proposes more expressive alternatives based on von Mises–Fisher distributions [7], beta kernels [36], soft Voronoi diagrams [14], as well as anisotropic spherical Gaussian and Gabor kernels [40]. Reflection-oriented methods instead introduce specialized parametrizations based on reflected or secondary rays to model challenging specular appearance [47, 55, 56, 60]. These methods primarily target novel-view quality, whereas we compare general-purpose appearance models under a common optimization and implementation framework, including their memory and rendering costs across desktop and mobile GPUs. We additionally study how their parametrization and expressiveness affect optimization beyond image metrics.

Neural Appearance Evaluation. Neural appearance models differ in where they are evaluated within the rendering pipeline. Deferred neural rendering [54] decodes appearance in image space after rasterization using CNNs or per-pixel MLPs [1, 9, 12, 17, 20, 25, 35, 49, 50], tying evaluation cost to rendering resolution. Forward neural rendering instead evaluates appearance at sampled 3D positions before compositing [8, 41, 45]. Scaffold-GS [37] similarly decodes

primitive attributes from compact latent features before rasterization. In contrast, our neural appearance model follows the forward paradigm but retains the standard per-primitive 3DGS representation, decoding only view-dependent color once per visible Gaussian as a direct replacement for spherical appearance models. To achieve practical frame rates, we rely on efficient on-GPU inference: tiny-cuda-nn [43] introduced fully fused networks and runtime code generation, enabling compact MLPs to be inlined into CUDA kernels for applications ranging from radiance caching [44] and layered material models [61] to visibility prediction [63]. Building on these developments, we integrate the forward and backward passes of our neural appearance model directly into a differentiable Gaussian rasterizer.

3. Preliminaries

We will first provide an overview of the standard view-dependent appearance model in 3D Gaussian Splatting, and then further expand to recently proposed alternatives.

3.1. Appearance Modelling in Gaussian Splatting

In 3D Gaussian Splatting (3DGS) [31], a scene is represented by a set of anisotropic Gaussians with position, covariance, opacity, and view-dependent color. Rendering proceeds in two stages. A per-primitive *preprocess* kernel projects each Gaussian onto the image plane, determines the overlapped 16×16 pixel tiles, and computes its view-dependent color. A subsequent tile-based *blend* kernel alpha-composites the Gaussians in each pixel in depth-sorted order.

Spherical Harmonics. Standard 3DGS represents appearance with spherical harmonics (SH). With the normalized direction $\mathbf{d}$ from the camera center to a Gaussian mean and RGB coefficients $\mathbf{c}_\ell^m \in \mathbb{R}^3$, the view-dependent color is

$$\mathbf{c}(\mathbf{d}) = \text{ReLU} \left(\underbrace{0.5 + Y_0 \mathbf{c}_0}_{\text{base color } \mathbf{c}_b} + \underbrace{\sum_{\ell=1}^L \sum_{m=-\ell}^{\ell} Y_\ell^m(\mathbf{d}) \mathbf{c}_\ell^m}_{\text{view-dependent residual } \mathbf{c}_r} \right). \quad (1)$$

Training begins with $L = 0$ and activates one additional band every 1000 iterations, usually up to $L = 3$. Degree-3 SH requires 48 scalar coefficients per Gaussian, of which 45 describe view dependence. The degree-zero coefficient is initialized from the SfM point color $\mathbf{c}_{\text{sfm}} \in [0, 1]^3$ as $\mathbf{c}_0 = (\mathbf{c}_{\text{sfm}} - 0.5)/Y_0$, while the remaining coefficients are initialized to zero. For a controlled comparison, we express all representations through a common interface

$$\mathbf{c}(\mathbf{d}) = \phi(\mathbf{c}_0 + \mathcal{A}_\psi(\mathbf{d})), \quad (2)$$

where $\mathcal{A}_\psi$ is the view-dependent appearance model and ϕ the color activation, including any constant shift. For standard 3DGS, $\phi(\hat{\mathbf{c}}) = \text{ReLU}(\hat{\mathbf{c}} + 0.5)$ and $\mathcal{A}_\psi$ is the SH expansion over degrees $\ell \geq 1$. We drop the constant factor Y_0 from the forward pass, initializing $\mathbf{c}_0 = \phi^{-1}(\mathbf{c}_{\text{sfm}})$

and multiplying its learning rate by Y_0 instead. Since the commonly used optimizer Adam [33] normalizes the gradient magnitude per parameter, both formulations take practically the same step in color space. All models retain the base color exactly as in the baseline, including its initialization and densification handling, and replace only $\mathcal{A}_\psi$. The color activation ϕ also affects optimization independently of $\mathcal{A}_\psi$. Beyond the baseline ReLU, natural alternatives include a smooth rectifier $\phi(\hat{\mathbf{c}}) = \text{softplus}(\hat{\mathbf{c}} + 0.5)$ ($\beta = 10$) [38], a scaled sigmoid $\phi(\hat{\mathbf{c}}) = \sigma(4\hat{\mathbf{c}})$, and a hard sigmoid $\phi(\hat{\mathbf{c}}) = \text{clamp}(\hat{\mathbf{c}} + 0.5, 0, 1)$. All four map zero to 0.5, the midpoint of the target RGB range, and are scaled to closely match the baseline ReLU near this operating point. All view-dependent parameters are initialized to produce a zero residual. Unless stated otherwise, we use the baseline ReLU for all appearance models in this work. However, as we will show in our experiments, the choice of color activation has a significant effect on optimization dynamics.

3.2. Alternative Explicit Models

Spherical Voronoi. Spherical Voronoi (SV) [14] represents appearance as a soft interpolation over a set of sites $\{\mathbf{s}_i\}$ on the sphere with associated RGB values $\{\mathbf{a}_i\}$,

$$\mathcal{A}_\psi(\mathbf{d}) = \sum_i \frac{\exp(-\tau_i \rho(\mathbf{d}, \mathbf{s}_i))}{\sum_j \exp(-\tau_j \rho(\mathbf{d}, \mathbf{s}_j))} \mathbf{a}_i, \quad (3)$$

where $\rho(\mathbf{d}, \mathbf{s}_i)$ is the distance between the viewing direction and site i , and τ_i controls the sharpness of its partition. Each site is parametrized by $(\mathbf{a}_i, \mathbf{s}_i, \tau_i) \in \mathbb{R}^7$. Since the softmax assigns a single site unit weight regardless of direction, at least two are needed for view-dependent variation. Note that the reference implementation does not separate a base color, but since the weights sum to unity, a shared base color can be factored out of the site values, a property that we use to align SV with the common interface in Eq. (2).

Normalized Anisotropic Spherical Gaussians. Normalized anisotropic spherical Gaussians (NASG), originally introduced for neural path guiding [30] and adapted to primitive-based reconstruction by Miazga *et al.* [40], represent appearance as a weighted sum of spherical kernels,

$$\mathcal{A}_\psi(\mathbf{d}) = \sum_i G_i(\mathbf{d}) \tanh(\mathbf{a}_i), \quad (4)$$

where each G_i has a learned orientation $\in \mathbb{R}^3$ and anisotropic angular extent $\in \mathbb{R}^2$, and $\mathbf{a}_i \in \mathbb{R}^3$ is its RGB amplitude, resulting in eight parameters per lobe. Normalization decouples a kernel’s contribution from its angular extent, while anisotropy allows different rates of variation along its two principal tangent directions. The $\tanh$ bounds each lobe’s color contribution, constraining the residual to a range compatible with the base color in Eq. (2). Miazga *et al.*

further extend G_i with a non-negative cosine carrier, yielding NASGabor with kernel $\tilde{G}_i$ and one additional frequency parameter per lobe. The carrier allows a single kernel to represent a multimodal signal while maintaining a compact memory footprint.

For all explicit models, we follow the parametrizations and initialization of the respective reference works, adapting them to the common interface in Eq. (2). We provide more detailed definitions and discuss all implementation-level differences in Secs. A and B.

4. Neural View-Dependent Appearance

In contrast to explicit, parametric models, which impose constraints or assumptions on the angular signal to be modelled in order to achieve their compactness, we replace the view-dependent parameters with a per-Gaussian latent code and decode appearance with a small network shared across the entire scene, on a per-primitive basis.

Each Gaussian stores a feature vector $\mathbf{f} \in \mathbb{R}^F$, and a shared MLP with parameters θ maps $\mathbf{f}$ and the viewing direction to an RGB residual. Its input is

$$\mathbf{x} = (\gamma_{\text{dir}}^{\mathcal{B}}(\mathbf{d}), \gamma_{\text{feat}}^k(\mathbf{f})) \in \mathbb{R}^D, \quad (5)$$

where $\gamma_{\text{dir}}^{\mathcal{B}}$ evaluates the SH basis functions of degrees $\mathcal{B}$ and γ_{feat}^k applies a frequency encoding [41] independently to each feature component. For a scalar f , the latter is

$$\gamma_{\text{feat}}^k(f) = (\sin(2^j f), \cos(2^j f))_{j=0}^{k-1}. \quad (6)$$

How the network inputs are split between the two encodings is the central design choice. The direction encoding is computed on the fly and costs no storage, but determines the *angular resolution* of the view dependence. The features are stored per Gaussian and determine the *conditioning capacity*, i.e., how much the shared decoder’s response can differ between primitives: a small F forces all Gaussians into a low-dimensional code space, while a large F makes the model behave more like classic per-primitive coefficients at a linear storage cost. Efficient evaluation on Tensor Cores requires the network input to be a multiple of 16. Since the degree-3 direction encoding inherited from the baseline already occupies 16 inputs, $D = 32$ is the smallest viable input size that accommodates per-Gaussian features. The remaining 16 dimensions encode $F = 8$ feature values with $k = 1$. The decoder is a fully connected MLP with two hidden layers of 16 neurons, three outputs, ReLU hidden activations, and no biases. Bounding its predicted residual with a tanh, we compute

$$\mathcal{A}_{\psi}(\mathbf{d}) = \tanh(\text{MLP}_{\theta}(\mathbf{x})). \quad (7)$$

The bounded residual discourages the decoder from compensating for an arbitrarily displaced base color. To retain the

baseline’s progressive optimization schedule, we initially disable the residual and enable the directional encoding bands according to the SH schedule. Inactive inputs are set to zero and consequently receive no gradient. The model stores three base-color values and eight latent features per Gaussian, while the shared MLP adds 816 parameters for the entire scene. Moreover, since both the forward and backward pass carry out all computations in half precision, MLP weights and features can be deployed in 16-bit without quality loss. With 32-bit base colors, the deployed appearance footprint is 28 bytes per Gaussian, compared with 192 bytes for 48 single-precision SH coefficients.

CUDA Implementation. We expose all appearance models through a common device interface whose forward and backward evaluations are inlined into the differentiable CUDA rasterizer. Using the runtime code-generation facilities of tiny-cuda-nn [43], the interface is specialized for each model and JIT-compiled with NVRTC on first use. This allows the appearance model to be selected via configuration without sacrificing compile-time optimizations, and makes the framework readily extensible to new representations while retaining the state-of-the-art training performance of Faster-GS [22]. Appearance is evaluated only for primitives that survive culling, directly within the preprocess kernel, requiring no separate pass or global-memory round trip. Directional gradients from the appearance evaluation are propagated to the Gaussian positions, preserving the appearance-dependent positional gradients used by the baseline’s densification procedure. We additionally retain a reference path that evaluates appearance in a separate pass and use it to validate each fused implementation. The remainder of Faster-GS, including tile assignment, sorting, blending, densification, and pruning, is unchanged. We provide implementation details, including optimization hyperparameters, in Secs. A and B.

WebGL Viewer. To evaluate deployment beyond CUDA-capable hardware, we implement all appearance models in a WebGL viewer built from scratch on Three.js. Depth sorting proceeds asynchronously on the CPU, matching the approach of state-of-the-art web viewers such as SuperSplat [46]. At the beginning of each frame, a fragment-shader prepass evaluates the view-dependent color for each Gaussian and writes the result to a texture consumed by the splatting pass, isolating appearance cost from splat blending. For the established SH model, the view-dependent coefficients are quantized to the compact bit-packed layout used by Spark [59], compressing a full degree-3 parametrization to 40 bytes per Gaussian. The neural model requires no additional quantization: features and MLP weights are stored and evaluated in their native half precision, and the view-independent contribution of the first MLP layer is pre-computed per Gaussian at load time. Note that because the network is shared by all primitives, its weights exhibit

Table 1. Quantitative results on established benchmark datasets using MCMC densification [32]. We employ PPISP [13] on real scenes to handle photometric variations and measure frames per second (FPS) at the native scene resolution. Since image metrics vary between runs, we report average values across five runs in this table to improve reproducibility.

Basis	Mip-NeRF 360 [5]							Tanks & Temples [34] / Deep Blending [24]						NeRF Synthetic [41]					
	Bytes/G	PSNR [↑]	SSIM [↑]	LPIPS [↓]	Train [↓]	VRAM [↓]	FPS [↑]	PSNR [↑]	SSIM [↑]	LPIPS [↓]	Train [↓]	VRAM [↓]	FPS [↑]	PSNR [↑]	SSIM [↑]	LPIPS [↓]	Train [↓]	VRAM [↓]	FPS [↑]
None	12	27.73	0.825	0.234	3m32s	4.6GiB	443.1	27.65	0.888	0.237	2m17s	3.3GiB	763.3	31.46	0.959	0.048	57s	1.3GiB	1734
SH	192	28.36	0.836	0.222	6m12s	6.7GiB	378.9	27.96	0.893	0.228	4m10s	4.9GiB	683.2	34.01	0.971	0.036	1m22s	1.5GiB	1696
SV	208	28.64	0.834	0.225	6m30s	6.8GiB	385.0	27.98	0.893	0.232	4m20s	5.0GiB	696.6	33.45	0.968	0.038	1m28s	1.6GiB	1576
NASG	44	28.50	0.836	0.223	3m54s	4.9GiB	439.9	27.97	0.893	0.227	2m36s	3.6GiB	757.4	32.69	0.966	0.043	59s	1.3GiB	1826
NASGabor	48	28.54	0.836	0.223	4m00s	5.0GiB	436.9	27.96	0.893	0.227	2m39s	3.6GiB	743.6	32.95	0.967	0.041	59s	1.3GiB	1852
Neural	28	28.55	0.831	0.227	4m17s	5.3GiB	443.6	28.13	0.893	0.228	2m56s	3.9GiB	743.8	34.04	0.970	0.038	1m02s	1.3GiB	1869

Table 2. Average frame time at 720p in milliseconds across scenes, devices, and rendering backends. CPU-side depth sorting is excluded. For WebGL, we load the same checkpoint for all models, *i.e.*, geometry is identical, assigning dummy values for the appearance parameters, and exclude the CPU-side depth sorting as well as application of the PPISP module. Values rounded roughly w.r.t. to measurement precision.

		Bicycle (6M Gaussians)					Stump (4.5M Gaussians)					Bonsai (1.5M Gaussians)				
Device		SH	SV	NASG	NASGabor	Neural	SH	SV	NASG	NASGabor	Neural	SH	SV	NASG	NASGabor	Neural
CUDA	Desktop (RTX 5090)	2.11	2.16	1.85	1.87	1.97	2.27	2.29	2.11	2.00	1.98	1.46	1.42	1.32	1.28	1.36
	Desktop (RTX 4090)	2.75	2.80	2.25	2.26	2.38	2.95	3.00	2.55	2.43	2.36	1.76	1.71	1.51	1.46	1.55
	Desktop (RTX 3090)	4.27	4.36	3.82	3.83	4.06	4.54	4.66	4.28	4.06	4.02	2.90	2.85	2.66	2.60	2.75
WebGL	Desktop (RTX 4090)	4.0	4.5	4.0	4.0	4.0	3.6	4.0	3.6	3.6	3.9	1.3	1.4	1.2	1.3	1.2
	MacBook (M1 Pro)	12.2	13.1	12.0	12.0	13.2	11.1	12.1	11.0	11.0	11.9	6.1	6.4	6.1	6.1	6.4
	iPad (M2 Pro)	35.9	40.0	35.6	35.6	38.1	31.7	34.8	31.3	31.3	33.4	15.6	16.6	15.5	15.5	16.0
	iPhone (A18 Pro)	45.0	OOM	44.3	44.3	45.1	41.0	OOM	40.5	40.4	41.4	17.7	19.7	17.5	17.5	18.5

high cache locality throughout the pre-pass. For SV, NASG, and NASGabor, all parameters are stored in half precision and view-independent quantities are precomputed at export time, leaving only the direction-dependent evaluation for the per-frame prepass. This enables a consistent performance comparison across desktop, laptop, and mobile GPUs.

5. Evaluation

Our aim is to provide a thorough evaluation on all possible axes of the appearance modelling problem in primitive-based novel view synthesis: reconstruction quality, VRAM usage, and rendering performance of the implemented appearance models on established benchmarks. We will further discuss on the specific advantages and disadvantages of explicit *vs.* implicit models, and finally report WebGL viewer performance for a mobile perspective.

5.1. Setup

We evaluate on a total of 21 scenes from Mip-NeRF 360 [5], Tanks & Temples [34], Deep Blending [24], and NeRF Synthetic [41], following the image resolutions and train/test splits of the original 3DGS paper [31]. All configurations are optimized with Faster-GS [22] using MCMC densification [32] with established scene-specific primitive counts [36] and identical hyperparameters apart from the appearance model. In particular, optimization and densification strictly follow the protocol of the baseline [13, 22, 32], with no appearance-model-specific tuning of non-appearance parameters. For SV we use seven sites, and for NASG and

NASGabor one lobe, matching the default configurations of the respective reference implementations. While the reference implementations of SV and NASGabor default to beta kernels [36], our evaluation uses Gaussian kernels throughout, as they are more widely adopted and we found no meaningful quality advantage with beta kernels. For detailed comparisons of alternative spherical functions, we refer to Miazga *et al.* [40]. To prevent more expressive appearance models from absorbing photometric variations in training images as a function of viewing direction, we use per-image signal processing (PPISP) [13] for the quality comparison. The WebGL performance evaluation omits PPISP, as evaluating its web deployment is outside the scope of this work.

5.2. Results

In general, our results showcase the bottleneck imposed by heavy appearance models, substantially diminishing training and inference performance. While on the other hand, compact yet flexible parametric models, and our proposed neural model, outperform them by virtue of their smaller memory footprint, while still managing comparable quality to the most expressive explicit models. We include a detailed qualitative analysis across the different models in Tab. 3. All representations support rendering the view-independent base color and the view-dependent residual separately (*cf.* Fig. 2), a property enabled by the common interface in Eq. (2). Unlike prior work, we visualize the residual as $|\mathbf{c}(\mathbf{d}) - \phi(\mathbf{c}_0)|$ rather than $\phi(\mathcal{A}_\psi(\mathbf{d}))$, since the residual is signed and, *e.g.*, $\phi = \text{ReLU}$ would hide negative contributions.

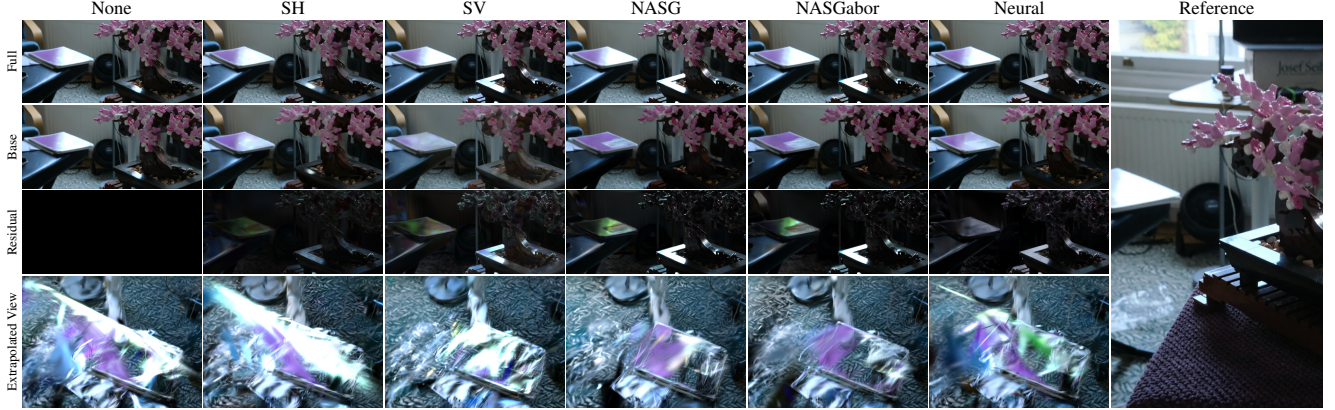


Figure 2. Greater angular expressivity can reduce geometric artifacts. Models with high angular expressivity separate view-dependent reflections from diffuse color, whereas less expressive models compensate through geometry. SH uses elongated white Gaussians to fake the reflection on the sketch block and misses the reflection on the bonsai pot, while NASGabor achieves the best diffuse/specular decomposition.

Table 3. Study of the evaluated view-dependent appearance models. Ranking goes from best (++) to worst (--).

Method	t _{Train}	t _{Render}	Angular Res.	Robustness	Simplicity	Compactness	Key Takeaway
SH	—	—	--	++	+	—	Overfits the least (strong SSIM/LPIPS); geometry must compensate for sharp reflections. ¹
SV	—	—	+	—	—	—	Parameter-efficient for sharp diffuse reflections; least robust method in our evaluation. ²
NASG(abor)	+	+	++	+	—	+	Best quality/speed trade-off; its sharp lobes generalize well and can explain capture artifacts. ³
Neural	+	o	+	o	++	++	Most compact in general settings; hard to regularize per scene. ⁴

¹Costly Adam updates must also touch Gaussians invisible in the current iteration; rendering incurs many memory fetches.

²Ineffective with few sites (softmax requires ≥ 2), making it slow and a poor fit for compact representations; explicit many-lobe models are hard to optimize via gradient descent.

³Weaker with a single lobe when primitives are few and lighting is controlled (*e.g.*, NeRF Synthetic); Gabor term is not worth it outside standard benchmarks.

⁴Fits training images far better than explicit spherical functions; capacity reduction through regularization helps low view-dependence scenes but worsens view-dependent ones.

Reconstruction Quality. Tab. 1 reports reconstruction quality, optimization time, peak memory, and CUDA rendering speed on an RTX 4090. All appearance models achieve similar quality across the evaluated datasets, with differences largely within run-to-run variation. The neural representation matches or exceeds degree-3 SH while storing 28 instead of 192 bytes per Gaussian, resulting in faster optimization, lower peak memory, and higher rendering speed due to reduced memory traffic during parameter updates.

WebGL Performance. Tab. 2 reports frame times at 720p, averaged over multiple renders per test view. In CUDA, NASG/NASGabor are the fastest models, rendering up to 18% faster than SH on the RTX 4090 due to reduced memory traffic, with the neural model ranking third. For the WebGL measurements, we load the same checkpoint for all models so that geometry is identical and differences stem exclusively from appearance evaluation. Without Tensor Cores, differences narrow and all representations except SV render within a few percent of each other. SV is the slowest model on all devices and runs out of memory on the iPhone for the two largest scenes.

Color Activations. Tab. 4 evaluates the color activations introduced in Sec. 3.1 across SH, NASGabor, and the neural model on Mip-NeRF 360. As shown in Fig. 3, differences manifest not only in metrics but in how well the optimization

Table 4. Quality metrics under different color activations and opacity learning rates across Mip-NeRF 360 scenes [5] using MCMC densification and PPISP. The variant with lr=auto adjusts the opacity learning rate between 0.05 and 0.025 based on the scene-specific training view distribution as proposed by Miazga *et al.* [40].

Method	SH		NASGabor		Neural	
	PSNR [†]	SSIM [†]	PSNR [†]	SSIM [†]	PSNR [†]	SSIM [†]
ReLU, lr=0.05	28.35	0.836	28.51	0.836	28.53	0.831
ReLU, lr=0.025	28.33	0.831	28.59	0.833	28.54	0.827
Softplus, lr=0.05	28.42	0.837	28.55	0.837	28.54	0.831
Softplus, lr=0.025	28.41	0.833	28.60	0.834	28.56	0.827
Hardsigmoid, lr=0.05	28.31	0.834	28.49	0.835	28.50	0.831
Hardsigmoid, lr=0.025	28.30	0.829	28.52	0.832	28.48	0.826
Sigmoid, lr=0.05	28.38	0.837	28.43	0.835	28.53	0.832
Sigmoid, lr=0.025	28.40	0.833	28.57	0.833	28.52	0.827
ReLU, lr=auto [40]	28.38	0.834	28.66	0.835	28.58	0.829

recovers geometric detail, *e.g.* the brick wall in the *garden* scene. Sigmoid achieves the strongest results in SSIM and qualitative sharpness, as its bounded range prevents the degenerate solution, where unbounded activations fit opaque white regions with low opacity and colors far exceeding the unit value. However, its vanishing gradients may cause worse fits in bright regions such as a white sky, where the source pixels are clipped to one. When investigating per-scene metrics, we find that the optimal configuration varies strongly across scenes: higher opacity learning rates gener-

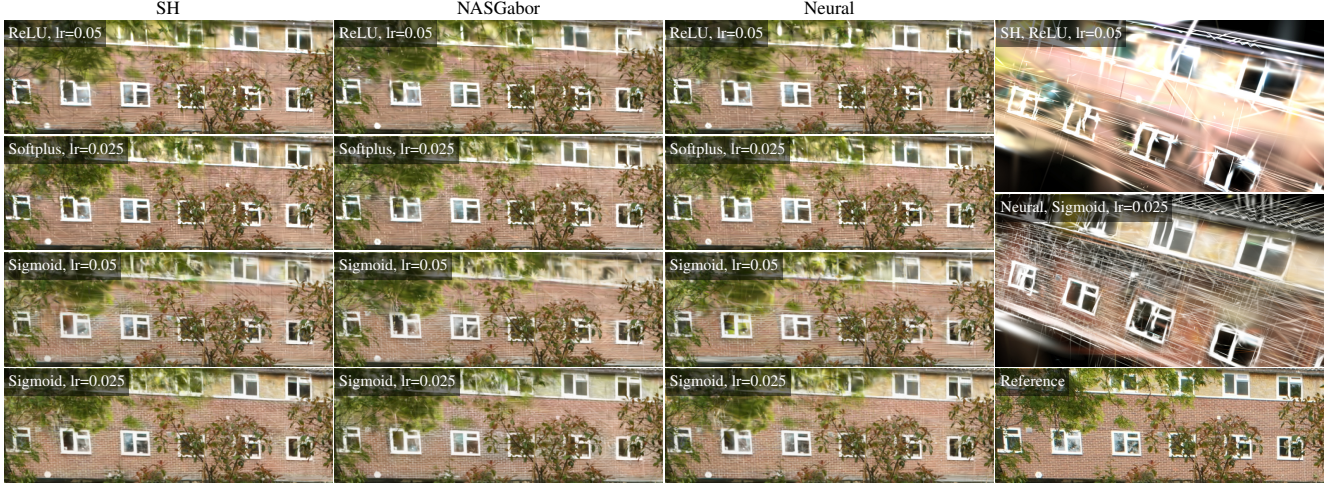


Figure 3. Effect of color activation and opacity learning rate on a test view of the *garden* scene. The choice of parametrization has strong influence on how well the optimization recovers meaningful geometry as evident from the significantly improved details of the brick wall.

ally favor SSIM, whereas a lower ones favor PSNR, indicating that the color activation and the opacity parametrization interact during optimization. This underscores that primitive-based radiance field methods require per-scene hyperparameter tuning to maximize quality, consistent with the findings of Miazga *et al.* [40], whose per-scene adaptive learning rate mitigates this and achieves the best PSNR for NASGabor and the neural model.

6. Discussion

While our evaluation provides a broad comparison among appearance models for novel-view synthesis, several factors complicate a definitive assessment of their relative strengths. In particular, we argue that *view-dependent appearance* should not necessarily be equated with purely modelling outgoing radiance. On real captures, view-dependent residuals can instead encode inconsistencies in the observations. These can be caused by changing illumination, shadows, exposure, scene motion, or imperfect geometry. An expressive appearance model can absorb these inconsistencies since, along a finite camera trajectory, temporal or spatial changes may be correlated with viewing direction.

This effect is visible, *e.g.*, in Mip-NeRF 360 scenes (*cf.* Figs. 4 and 5). In *bonsai*, the top-most book moved during capture. NASGabor’s expressive angular model can partially represent this discrepancy as a view-dependent effect, whereas the more constrained SH representation has less capacity to do so and consequently needs to abuse deficiencies of the rendering algorithm such as popping [48]. Similarly, in *garden*, the view-dependent SH residual reveals a region where the camera operator stepped on the grass during capture. Here, the residual is not a reflectance property, but rather the representation of a scene change that happened to correlate with the camera trajectory.

This raises a broader question: to what extent does the advantage of expressive models on real-world benchmarks measure capacity to model outgoing radiance, or does it simply measure robustness to violations of the static-scene assumption? In other words, a model may obtain better novel-view synthesis by being better at explaining inconsistent observations, without necessarily recovering a more faithful representation of the underlying scene. This ambiguity is largely absent from synthetic benchmarks with fixed geometry, lighting, and camera parameters, but these introduce a different bias. Their predominantly diffuse, hard-surface content can penalize explicit representations whose primitives have limited angular expressivity, unless additional parametric lobes are introduced.

Consequently, current benchmarks make it difficult to isolate the intrinsic quality of an appearance representation. We therefore believe that future evaluations should distinguish *reflectance* and *lighting-dependent angular variation* from *observation inconsistency*, rather than treating both simply as view-dependent appearance. Despite this ambiguity in image quality, our memory and performance measurements provide a more direct metric: compact spherical-parametric and implicit representations offer a substantial practical advantage over storing increasingly expressive but heavy appearance models explicitly per primitive.

Limitations. All appearance models in this comparison share the limitations inherent to spherical color functions in 3DGS: they cannot accurately model mirror reflections, which require explicit tracing of secondary rays [56, 60]. For the neural representation, the shared decoder introduces an additional constraint: unlike an explicit spherical basis, its response cannot be analytically rotated, complicating scene composition and editing [51]. This can be addressed by applying the inverse model transformation to the view direction



Figure 4. Base color and view-dependent residual along a camera path between two training views of the *bonsai* scene, cropped to the books on the floor. The first and last crop show the reference views, the others show in-between views. The topmost book moved during the data capture. NASGabor’s expressivity allows partially fitting this artifact as a view-dependent effect, while SH requires the geometry to compensate through popping [48], an insight that motivates employing expressive appearance models with pixel-sorted approaches [21, 52].



Figure 5. Base color and view-dependent residual with SH appearance rendered separately for a top-down view of the *garden* scene. The residual reveals where the camera operator inadvertently modified the scene by stepping on the grass during the capture, *i.e.*, the optimizer fit changes in geometry through the appearance model.

before querying the network. The shared decoder may also become a capacity bottleneck as scene scale increases. One possible solution is to store multiple sets of decoder weights, as in SMERF [16], which partitions the scene into a grid and trilinearly interpolates per-cell weights at the camera origin. Besides increasing capacity, such a formulation naturally supports scene composition. Finally, WebGL does not expose the hardware acceleration that makes neural decoding inexpensive in CUDA, making parametric models faster in this setting. As mobile hardware increasingly incorporates neural accelerators and APIs such as WebGPU provide access to them, we expect this gap to narrow.

Future Work. The compact neural representation is particularly attractive for feedforward reconstruction, since predicting an 8-dimensional latent code is substantially simpler than predicting 48 SH coefficients, while the decoder itself can be shared across scenes. More broadly, our unified framework makes it possible to study the interaction between

appearance representation, color activation, and densification strategy in a controlled manner. We believe this joint design space, rather than treating these components independently, is a promising direction for future work.

7. Conclusion

We presented a controlled comparison of view-dependent appearance models for Gaussian splatting, integrating spherical harmonics, spherical Voronoi, normalized anisotropic spherical Gaussians, NASGabor, and a compact neural decoder into a common pipeline with optimized CUDA and WebGL implementations. Our evaluation shows that recent spherical models provide strong quality–efficiency trade-offs, while our neural representation is the most compact, reducing the per-primitive appearance footprint from 192 to 28 bytes compared to third-degree SH while matching or improving reconstruction quality. Beyond image quality, we find that appearance parametrization and color activation influence optimization and, consequently, recovered geometry, highlighting the need for controlled comparisons when evaluating this design space. At the same time, non-static scenes and transient capture artifacts in common benchmarks complicate the interpretation of view-dependent appearance, as models may explain such inconsistencies through appearance or geometry. Because all evaluated models operate on per-primitive view directions, our findings transfer directly to other explicit primitive-based radiance-field representations, extending their relevance beyond Gaussian splatting. We release our configurable framework to facilitate further research into appearance representations and their quality, memory, and platform-specific trade-offs.

Acknowledgments

We thank Timon Scholz for his contributions to the software infrastructure supporting this work and Jannis Möller for identifying and helping us fix a bug in tiny-cuda-nn. This work was partially funded by the DFG projects “Real-Action VR” (ID 523421583) and “Increasing Realism of Omnidirectional Videos in Virtual Reality” (ID 491805996).

References

- [1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, pages 696–712, 2020. 2
- [2] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. FLIP: A difference evaluator for alternating images. *HPG*, 3(2), 2020. 15
- [3] Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3DGS.zip: A survey on 3D Gaussian splatting compression methods. *Comput. Graph. Forum*, 44(2):e70078, 2025. 1, 2
- [4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5835–5844, 2021. 11
- [5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5460–5469, 2022. 2, 5, 6, 14, 15, 17
- [6] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *ICCV*, pages 19640–19648, 2023. 1, 2
- [7] Hugo Blanc, Jean-Emmanuel Deschaut, and Alexis Paljic. RayGauss: Volumetric Gaussian-based ray casting for photo-realistic novel view synthesis. In *WACV*, pages 1808–1817, 2025. 2
- [8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensorRF: Tensorial radiance fields. In *ECCV*, pages 333–350, 2022. 1, 2
- [9] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, pages 16569–16578, 2023. 2
- [10] Ugur Cogalan, Mojtaba Bermana, Karol Myszkowski, Hans-Peter Seidel, and Colin Groth. MILO: A lightweight perceptual quality metric for image and latent-space optimization. *ACM TOG*, 44(6), 2025. 15
- [11] Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. Don’t splat your Gaussians: Volumetric ray-traced primitives for modeling and rendering scattering and emissive media. *ACM TOG*, 44(1), 2025. 2
- [12] Jorge Condor, Nicolas Moenne-Loccoz, Merlin Nimier-David, Piotr Didyk, Zan Gojcic, and Qi Wu. Neural harmonic textures for high-quality primitive based neural reconstruction. In *ECCV*, 2026. 1, 2
- [13] Isaac Deutsch, Nicolas Moëne-Loccoz, Gavriel State, and Zan Gojcic. PPISP: Physically-plausible compensation and control of photometric variations in radiance field reconstruction. In *CVPR*, pages 7289–7298, 2026. 5, 12, 14
- [14] Francesco Di Sario, Daniel Rebain, Dor Verbin, Marco Grangetto, and Andrea Tagliasacchi. Spherical Voronoi: Directional appearance as a differentiable partition of the sphere. In *CVPR*, pages 22529–22538, 2026. 1, 2, 3, 11
- [15] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. Image quality assessment: Unifying structure and texture similarity. *TPAMI*, 44(5):2567–2581, 2022. 15
- [16] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. SMERF: Streamable memory efficient radiance fields for real-time large-scene exploration. *ACM TOG*, 43(4), 2024. 8, 13
- [17] Linus Franke, Darius Rückert, Laura Fink, and Marc Stamminger. TRIPS: Trilinear point splatting for real-time radiance field rendering. *Comput. Graph. Forum*, 43(2):e15012, 2024. 2
- [18] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5491–5500, 2022. 1, 2
- [19] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant Foam: Real-time differentiable ray tracing. In *ICCV*, pages 4135–4145, 2025. 1, 2
- [20] Florian Hahlbohm, Linus Franke, Moritz Kappel, Susana Castillo, Martin Eisemann, Marc Stamminger, and Marcus Magnor. INPC: Implicit neural point clouds for radiance field rendering. In *3DV*, pages 168–178, 2025. 1, 2
- [21] Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. Efficient perspective-correct 3D Gaussian splatting using hybrid transparency. *Comput. Graph. Forum*, 44(2):e70014, 2025. 8
- [22] Florian Hahlbohm, Linus Franke, Martin Eisemann, and Marcus Magnor. Faster-GS: Analyzing and improving Gaussian splatting optimization. In *CVPR*, pages 18946–18957, 2026. 1, 4, 5, 12
- [23] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D Gaussian splatting with sparse pixels and sparse primitives. In *CVPR*, pages 21537–21546, 2025. 14
- [24] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM TOG*, 37(6):257:1–257:15, 2018. 5, 18
- [25] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, pages 5855–5864, 2021. 1, 2
- [26] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi,

- Bernard Ghanem, and Marc Van Droogenbroeck. 3D convex splatting: Radiance field rendering with 3D smooth convexes. In *CVPR*, pages 21360–21369, 2025. 1, 2
- [27] Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. Mesh-Splatting: Differentiable rendering with opaque meshes. In *CVPR*, pages 7320–7329, 2026. 2
- [28] Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Daniel Rebain, Anthony Cioppa, Bernard Ghanem, Andrea Vedaldi, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In *3DV*, pages 1248–1257, 2026. 1, 2
- [29] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, pages 32:1–32:11, 2024. 1, 2
- [30] Jiawei Huang, Akito Iizuka, Hajime Tanaka, Taku Komura, and Yoshifumi Kitamura. Online neural path guiding with normalized anisotropic spherical Gaussians. *ACM TOG*, 43(3), 2024. 3
- [31] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 1, 2, 3, 5, 11, 14
- [32] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov chain Monte Carlo. In *NeurIPS*, 2024. 5, 12, 14
- [33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3, 12
- [34] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 36(4), 2017. 5, 18
- [35] Georgios Kopanas, Julien Philip, Thomas Leimkuehler, and George Drettakis. Point-based neural rendering with per-view optimization. *Comput. Graph. Forum*, 40(4):29–43, 2021. 2
- [36] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting. In *SIGGRAPH*, pages 101:1–101:11, 2025. 1, 2, 5, 12, 14
- [37] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. In *CVPR*, pages 20654–20664, 2024. 2
- [38] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. EVER: Exact volumetric ellipsoid rendering for real-time view synthesis. In *ICCV*, pages 4930–4939, 2025. 1, 2, 3
- [39] Alexander Mai, Trevor Hedstrom, George Kopanas, Janne Kontkanen, Falko Kuester, and Jonathan T. Barron. Radiance meshes for volumetric reconstruction. In *CVPR*, pages 8267–8277, 2026. 2
- [40] Ewa Miazga, Jorge Condor, and Piotr Didyk. Beyond spherical harmonics: Rethinking appearance models for radiance reconstruction. *Comput. Graph. Forum*, 2026. 1, 2, 3, 5, 6, 7, 11, 12
- [41] 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*, pages 405–421, 2020. 1, 2, 4, 5, 19
- [42] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D Gaussian Ray Tracing: Fast tracing of particle scenes. *ACM TOG*, 43(6), 2024. 2
- [43] Thomas Müller. tiny-cuda-nn, 2021. 1, 3, 4, 12
- [44] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM TOG*, 40(4), 2021. 3
- [45] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4), 2022. 1, 2, 12
- [46] PlayCanvas. SuperSplat, 2024. 4, 13
- [47] Yohan Poirier-Ginter, Jeffrey Hu, Jean-François Lalonde, and George Drettakis. Editable physically-based reflections in raytraced gaussian radiance fields. In *SIGGRAPH Asia*, pages 201:1–201:12, 2025. 2
- [48] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. *ACM TOG*, 43(4), 2024. 7, 8
- [49] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM TOG*, 42(4), 2023. 2
- [50] Darius Rückert, Linus Franke, and Marc Stamminger. ADOP: Approximate differentiable one-pixel point rendering. *ACM TOG*, 41(4), 2022. 2
- [51] Markus Schütz, Christoph Peters, Florian Hahlbohm, Elmar Eisemann, Marcus Magnor, and Michael Wimmer. Splatshop: Efficiently editing large Gaussian splat models. *Comput. Graph. Forum*, page e70214, 2025. 7
- [52] Michael Steiner, Thomas Köhler, Lukas Radl, Felix Windisch, Dieter Schmalstieg, and Markus Steinberger. AAA-Gaussians: Anti-aliased and artifact-free 3D Gaussian rendering. In *ICCV*, pages 27650–27659, 2025. 8
- [53] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5449–5459, 2022. 2
- [54] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG*, 38(4), 2019. 2
- [55] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *CVPR*, pages 5481–5490, 2022. 2
- [56] Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T. Barron. NeRF-Casting: Improved view-dependent appearance with consistent reflections. In *SIGGRAPH Asia*, pages 15:1–15:10, 2024. 2, 7

- [57] Nicolas von Lützwow and Matthias Nießner. LinPrim: Linear primitives for differentiable volumetric rendering. In *NeurIPS*, 2025. 1, 2
- [58] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 15
- [59] World Labs. Spark, 2025. 4, 13
- [60] Tao Xie, Xi Chen, Zhen Xu, Yiman Xie, Yudong Jin, Yujun Shen, Sida Peng, Hujun Bao, and Xiaowei Zhou. EnvGS: Modeling view-dependent appearance with environment Gaussian. In *CVPR*, pages 5742–5751, 2025. 2, 7
- [61] Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, and Aaron Lefohn. Real-time neural appearance models. *ACM TOG*, 43(3), 2024. 3
- [62] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 15
- [63] Brent Zoomers, Florian Hahlbohm, Joni Vanherck, Lode Jorissen, Marcus Magnor, and Nick Michiels. NVGS: Neural visibility for occlusion culling in 3D Gaussian splatting. In *CVPR*, pages 29824–29833, 2026. 3

A. Appearance Model Details

This section complements Secs. 3.1 and 4 with the parametrization, initialization, and optimization of each appearance model, and lists where our implementation deviates from the respective reference. We intentionally restrict all model-specific adaptations to the appearance parameters, *i.e.*, the base color and the view-dependent parameters. Changes that the reference implementations make to the geometry, such as different learning rates or schedules for means, scales, rotations, and opacities, are omitted. This isolates the effect of the appearance model, at the cost of not reproducing every reference recipe exactly. Beyond integrating each model into the fused rasterizer, we improve upon the reference implementations at the kernel level, which we point out in the respective subsections. Tab. 5 summarizes the appearance learning rates and schedules. All other hyperparameters are shared and described in Sec. B.1.

A.1. Spherical Harmonics

The SH model is the standard 3DGS [31] parametrization in the common interface of Eq. (2): $\mathbf{c}_0 = \phi^{-1}(\mathbf{c}_{\text{sfm}})$ and 45 zero-initialized coefficients for degrees one to three, activated at iterations 1000, 2000, and 3000. As described in Sec. 3.1, the constant $Y_0 = 1/\sqrt{4\pi} \approx 0.282$ is folded into the base color, whose learning rate is scaled by Y_0 accordingly, while the coefficients use the standard rate of 0.000125. Both stay constant during optimization. The model without view-dependent appearance in Tab. 1 is this model restricted to degree zero.

Table 5. Learning rates and schedules of the appearance parameters. *Cosine* follows Miazga *et al.* [40]: decay from iteration 7k toward zero at 37k, *i.e.*, about 13% of the initial rate remain at the end of optimization. *Delayed cosine* follows Di Sario *et al.* [14]: frozen until 1k, decay to 0.1× at 27k, frozen after. *Warm-up* follows Mip-NeRF [4]: sinusoidal warm-up over 3k, then log-linear decay to 0.1× at 30k. $\bar{d}$ is the mean distance between each training camera and its three nearest neighbors.

Model	Parameter	Learning rate	Schedule
SH	base color	$0.0025 \cdot Y_0$	constant
	coefficients	0.000125	constant
SV	base color	0.0008	constant
	site colors	0.0008	constant
	sites	0.002	delayed cosine
	temperatures	0.006	delayed cosine
NASG(abor)	base color	0.00025	cosine
	lobes	$0.0025 \cdot (0.3536/\bar{d})^2$	cosine
Neural	base color	0.00025	cosine
	features	0.008	cosine
	weights	0.003	warm-up

A.2. Spherical Voronoi

Each site stores seven raw parameters, *i.e.*, an unnormalized site direction, a temperature, and an RGB value, which are mapped to $\mathbf{s}_i = \text{normalize}(\cdot)$, $\tau_i = \exp(\cdot)$, and $\mathbf{a}_i$ with the identity residual activation. The distance in Eq. (3) is the chord distance $\rho(\mathbf{d}, \mathbf{s}_i) = \|\mathbf{s}_i - \mathbf{d}\|$. We improve the evaluation of the sites in two respects. First, the reference implementation [14] normalizes the sites and exponentiates the temperatures in PyTorch before every rasterizer call, with automatic differentiation handling their gradients in separate passes. Our kernels instead take the raw parameters and fuse both activations and their derivatives into the forward and backward pass. Second, the reference evaluates the numerically stable softmax in three passes over the sites, *i.e.*, maximum, normalizer, and weighted sum, each recomputing the distances. Our forward pass reads the site parameters once and maintains a running maximum, and our backward pass requires two passes. Following the reference, sites are initialized on a Fibonacci lattice shared by all Gaussians with $\tau_i = 1$, all seven sites are active from the first iteration, and sites and temperatures follow the delayed cosine schedule in Tab. 5. The reference stores the full color per site, whereas our base color takes the role of their shared component. With softmax weights $w_i(\mathbf{d})$ that sum to one, both are equivalent:

$$\sum_i w_i(\mathbf{d}) (\mathbf{c}_0 + \mathbf{a}_i) = \mathbf{c}_0 + \sum_i w_i(\mathbf{d}) \mathbf{a}_i = \mathbf{c}_0 + \mathcal{A}_\psi(\mathbf{d}). \quad (8)$$

Accordingly, the base color uses the site color learning rate and is initialized from the SfM color, with all site colors initialized to zero, reproducing the reference initialization exactly. We follow the reference implementation based on 3DGS-MCMC, which uses Gaussian kernels and propagates the color gradient into the Gaussian mean, whereas their

beta-kernel implementation does not. Unlike the reference, we do not freeze scales, rotations, and opacities during the last 3000 iterations.

A.3. NASG and NASGabor

We use the kernel definition, parameter activations, and numerical safeguards of Miazga *et al.* [40] unchanged. Raw frame angles are mapped by $\tanh$ to the cosines of the lobe orientation, sharpness and anisotropy by a clamped exp, and the Gabor frequency by $20(\tanh(\cdot) + 1)$, while the lobe amplitudes pass through the $\tanh$ residual activation of Eq. (4). As for SV, we fuse all activations into the kernels, whereas the reference applies the $\tanh$ of the orientation angles and amplitudes in PyTorch and only the exponentials in-kernel. Initialization also follows the reference, including raw amplitudes of 0.5, so that a lobe contributes a nonzero residual of $\tanh(0.5)$ once it is activated at iteration 1000. The lobe learning rate is scaled by the reference’s scene-adaptive factor derived from the training view spacing (*cf.* Tab. 5). The reference implementation differs from ours in three respects: it renders beta-kernel primitives [36], composites unclamped and possibly negative colors instead of applying a color activation, and does not propagate the color gradient into the Gaussian mean. We use Gaussian kernels for all models and prevent negative colors through the color activation, since PPISP [13] requires non-negative per-pixel radiance and we observed a small improvement from doing so even without PPISP. Moreover, we derive and implement the missing gradient with respect to the Gaussian mean for both lobe models, so that all models drive densification through the same appearance-dependent gradients.

A.4. Neural

The decoder input is assembled as in Eq. (5): the $F = 8$ features are frequency-encoded with $k = 1$ into 16 values, and the view direction is encoded by the SH basis of degrees zero to three into 16 values. The constant degree-zero term acts as a bias input, since the network itself has no biases. Directional degrees are activated one by one every 1000 iterations following the SH schedule and zeroed while inactive, and the residual is disabled entirely until the first degree activates at iteration 1000. Hidden layers use the default initialization of tiny-cuda-nn [43], whereas the output layer is initialized to zero, so that the residual is exactly zero when it activates. The features are initialized to zero as well. The base color and features follow the cosine schedule of the lobe models, whereas the shared weights use the warm-up schedule in Tab. 5. For the weights, we further set the second moment coefficient of Adam [33] to $\beta_2 = 0.99$ instead of 0.999, since they receive dense gradients from every visible Gaussian in every iteration.

B. Implementation Details

This section describes the shared optimization protocol (Sec. B.1), the differentiable CUDA rasterizer with a focus on the fused evaluation of the neural model (Sec. B.2), and the WebGL viewer including its per-model payloads and the frame time measurement (Sec. B.3).

B.1. Optimization

All appearance models are optimized for 30k iterations with the protocol of Faster-GS [22] using MCMC densification [32], and PPISP [13] is enabled on real scenes. Following 3DGS, the schedule activates one additional appearance degree, *i.e.*, one SH degree, one lobe, or one directional encoding degree, every 1000 iterations, whereas SV has all sites active from the first iteration as in its reference. Since sites and temperatures are frozen for the first 1000 iterations, this initially only trains the site colors while the Adam moments of the other parameters warm up. For all models, the color gradient is propagated through the view direction into the Gaussian mean (*cf.* Sec. 4). The model without view-dependent appearance disables this path. The only change to the fused Adam implementation of Hahlbohm *et al.* [22] is that it accepts one learning rate per column of a parameter tensor. This applies the reference learning rates of SV to the seven parameters of each site without splitting them into separate tensors.

B.2. Differentiable CUDA Rasterizer

Each appearance model contributes a pair of device functions, the residual and its backward, against the common interface described in Sec. 4. The rasterizer combines the selected pair with its preprocess kernels and JIT-compiles the result with the model configuration, *e.g.*, the number of sites or lobes and the activations, as compile-time constants. Adding a model therefore requires no changes to the rasterization pipeline, and since the compiled kernels are fully specialized, this configurability costs no performance. The explicit models are evaluated per primitive with scalar arithmetic in single precision and require no further measures. The neural model is evaluated as a tiny-cuda-nn fully fused MLP [43] in half precision on Tensor Cores, one warp per 32 Gaussians, which requires all lanes of a warp to participate even if some primitives were culled. Following Müller *et al.* [45], the weights are stored and updated by Adam in single precision, whereas the forward and backward pass operate on a half-precision copy created before each iteration. The weight gradient is accumulated in half precision using a loss scale of 128. As a power of two, the scale only shifts the floating-point exponent and is removed without rounding error when the gradient is converted back to single precision. The gradients with respect to the features and the base color are computed in single precision. The forward pass stores the layer activations of each Gaussian for the backward pass.

We validate all fused implementations against the reference path that evaluates the appearance model in a separate pass. Images and gradients agree to within floating-point accumulation error, and for the neural model, the last two rows of Tab. 8 show that both paths yield the same quality.

B.3. WebGL Viewer

Each frame, the prepass of our Three.js-based viewer evaluates the color of every Gaussian in a fragment shader and writes it to a texture with one 8-bit RGBA texel per Gaussian. The subsequent splatting pass reads this color and alpha-composites the Gaussians in the depth order computed asynchronously on the CPU, following SuperSplat [46]. Geometry is packed into 16 bytes per Gaussian following Spark [59]: a half-precision mean, 8-bit logarithmic scale codes, a 24-bit octahedral-folded quaternion, and the 8-bit base color and opacity. The per-model payloads in Tab. 6 therefore hold only the view-dependent parameters, stored in 32-bit integer textures that pack eight half-precision values or bit-packed coefficient codes per texel. All view-independent computations, *i.e.*, parameter activations and normalization constants, are precomputed when a model is exported for the viewer, so the per-frame work of every model is only its direction-dependent evaluation. The residual is composed with the base color through the same activation ϕ as in CUDA. Notably, the neural model is the only one whose residual is evaluated at the same precision as in CUDA. Its features and weights are stored in half precision in both implementations, and the shader arithmetic on mobile GPUs is half precision as well. The explicit models are evaluated in single precision in CUDA but stored in half precision for the viewer, except for SH, whose coefficients use an elaborate quantization scheme (see below). We leave proposing and evaluating similarly elaborate quantization schemes for the other models to future work.

SH. The coefficients are quantized to the packed layout of Spark [59]. Degree one stores its nine coefficients at 7 bits in a two-channel texel, and degrees two and three store their 15 and 21 coefficients at 8 and 6 bits in one four-channel texel each. Quantization is symmetric with one shared scale per degree, the maximum coefficient magnitude across the scene, which is stored once per scene and used for dequantization in the shader.

SV. Sites are stored normalized with activated temperatures and colors, and the shader evaluates the softmax in a single pass with a running maximum as in CUDA.

NASG and NASGabor. Lobes are stored fully activated as the two frame axes, the shape parameters, the precomputed normalization constant, and the amplitude. The per-frame evaluation of a lobe thus reduces to two dot products, one power, and one exponential, plus one cosine for NASGabor.

Neural. The network is evaluated as in SMERF [16]:

Table 6. Per-Gaussian appearance payload of the WebGL viewer for the default configuration of each model and the per-frame work of the shader. Byte counts are padded to whole 16-byte texels, the base color is packed with the geometry, and the contents of each payload are described in the text.

Model	Payload	Bytes/G	Per-frame work
SH	45 coefficients at 7/8/6 bit	40	SH basis, weighted sum
SV	7 sites $\times$ 7 fp16	112	site distances, softmax
NASG	1 lobe $\times$ 12 fp16	32	lobe response
NASGabor	1 lobe $\times$ 13 fp16	32	lobe response, carrier
Neural	16 fp16 pre-activations	32	MLP

weights are stored as half-precision texels holding four consecutive input weights of one neuron, and each layer accumulates one 4×4 matrix product per block of four outputs. The weights reside in a uniform array when the device’s uniform limit permits and otherwise in a half-precision texture. Since the network is shared by all Gaussians, its weights stay cache-resident throughout the pass. We propose an additional optimization that exploits the view-independence of the 16 encoded features and the constant degree-zero input. Their 17 columns of the first weight matrix are folded into a per-Gaussian vector of 16 half-precision pre-activations, one per neuron, which is stored in place of the features at the same size. Only the 15 columns of the direction-dependent degrees are kept, padded to 16. The shader thus evaluates only the direction-dependent part of the first layer, reducing the per-Gaussian cost from 816 to 560 multiply-accumulate operations, *i.e.*, by 31%. This precomputation only pays off while its result is no larger than the features it replaces. With 32 neurons, it would store 32 values per Gaussian, twice the 16 encoded feature values, so we store the features and the full first layer instead. This is why the 32-neuron variant in Tab. 8 lists the full count.

Frame Time Measurement. For Tab. 2, the viewer renders the test views of each scene at 1280×720 with the native camera intrinsics. After a warm-up pass over all views, it waits for the asynchronous depth sort of each view to complete and renders once untimed before timing repeated renders, so that sorting and the upload of the sort order are excluded and the measurements reflect the steady-state cost at a novel view. The reported frame time is the mean over all test views and all repeated renders of each view. On the iPad and iPhone, sustained rendering triggers an unavoidable slowdown after a few dozen seconds, which is why we render each view only 10 times on these devices, whereas all other frame time and FPS measurements use 100 renders per view.

C. Additional Experiments

This section extends the evaluation of the main paper with an ablation of the densification strategy (Sec. C.1), an ablation of the neural model architecture (Sec. C.2), qualitative

Table 7. Quality metrics across Mip-NeRF 360 scenes [5] with different densification strategies and PPISP [13]. #Gs refers to the number of potentially visible Gaussians in the final model, *i.e.*, those with opacity $\geq 1/255$ after training.

Method	SH				NASGabor				Neural			
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	#Gs $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	#Gs $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	#Gs $\downarrow$
MCMC [32]	28.35	0.836	0.222	2.91M	28.51	0.836	0.223	2.90M	28.53	0.831	0.227	2.96M
$\hookrightarrow$ with $0.1 \times \#Gs$	27.30	0.798	0.309	0.32M	27.32	0.796	0.312	0.32M	27.30	0.793	0.316	0.32M
ADC [31]	27.88	0.818	0.250	2.79M	28.14	0.818	0.252	2.80M	28.04	0.812	0.258	2.88M
$\hookrightarrow$ with [23]	27.23	0.777	0.346	0.33M	27.29	0.779	0.343	0.33M	27.24	0.777	0.349	0.34M

examples of the decomposition into base color and residual (Sec. C.3), and per-scene results with additional metrics (Sec. C.4). We first provide additional details on how the numbers reported in the main paper and in this section are obtained. Unless stated otherwise, all experiments follow the setup of Sec. 5, *i.e.*, MCMC densification [32] with scene-specific primitive budgets [36] and PPISP [13] on real scenes. Since image metrics vary between training runs, Tab. 1 and the per-scene results in Tabs. 9 to 11 report averages over five runs, whereas all other tables report a single run. We measure optimization time, peak memory, and frame rates only during the first run, which we execute for all methods on the same workstation with an RTX 4090. The reported optimization times exclude the PPISP controller distillation, which runs after optimization and is identical for all models. The remaining runs are distributed across shared compute servers, where concurrent jobs compete for CPU resources and render timings unreliable. As timings and memory consumption vary far less between runs than image metrics, a single measurement is sufficient for these quantities.

C.1. Densification Strategies

To verify that our findings are not tied to the densification strategy, Tab. 7 repeats the Mip-NeRF 360 comparison for SH, NASGabor, and the neural model with the adaptive density control (ADC) of 3DGS [31]. We further include two low-budget variants: MCMC with a tenth of the primitive budget and ADC combined with the pruning of Speedy-Splat [23]. Since the two high-budget and the two low-budget configurations arrive at almost identical numbers of visible Gaussians, this also compares the densification strategies at equal model size. The relative ranking of the appearance models is stable across densification strategies: NASGabor and the neural model match or exceed SH in PSNR, whereas SH and NASGabor share the best SSIM and LPIPS, mirroring Tab. 1. The differences between appearance models shrink with the primitive budget, as reconstruction quality becomes limited by geometric coverage, which no amount of angular expressivity can compensate. Conversely, the advantage of the expressive models is largest with ADC. A plausible explanation is that ADC decides where to clone and split primitives based on positional gradients, which include the directional gradients propagated through the appearance model (*cf.* Sec. 4), whereas MCMC

relocates primitives based on opacity alone. Finally, MCMC outperforms ADC at equal model size for all appearance models. Its advantage in SSIM and LPIPS persists at the low budget, as opacity-driven relocation distributes a limited budget more evenly than a pruned ADC model.

C.2. Neural Model Architecture

Tab. 8 varies the architecture of the neural appearance model. Starting from the default decoder (*cf.* Sec. 4), we vary depth, width, and input encoding, where $F = 16$ raw features without frequency encoding yield the same 16 inputs, and additionally evaluate the default configuration through the unfused reference path. Besides image metrics, optimization time, peak memory, and CUDA frame rate on the RTX 4090, we report the appearance footprint per Gaussian and the number of multiply-accumulate operations (MAC) needed to decode a single Gaussian. The latter is the relevant cost for the WebGL viewer since it cannot use Tensor Cores.

Capacity. All fused configurations achieve similar quality, so the decoder is far from being the limiting factor. The differences that do exist are concentrated in the indoor scenes, which contain the glossy surfaces for which angular resolution matters (*cf.* Fig. 2). Outdoor scenes are dominated by diffuse content, and their quality is limited by geometry and observation inconsistency rather than by appearance capacity (*cf.* Sec. 6). Accordingly, a single hidden layer loses quality indoors, while wider or deeper decoders gain. Doubling the latent code without frequency encoding performs on par with a deeper decoder but is the only configuration that increases the per-Gaussian footprint. Additional capacity in the shared decoder is free in terms of storage and memory traffic, whereas additional per-primitive features are paid for by every Gaussian. We therefore consider wider or deeper decoders the preferable way of scaling the model.

Cost. Optimization time and peak memory vary only marginally across the fused configurations, as the decoder accounts for a small fraction of the per-iteration cost. CUDA frame rates behave counterintuitively: the widest decoder is the fastest to render despite requiring almost four times as many MACs as the default. In the fused CUDA renderer, the decoder is evaluated once per visible Gaussian on Tensor Cores in the preprocess kernel, whose cost is small compared to the subsequent blending, which scales with the number of

Table 8. Comparison of different model configurations for the neural appearance model across Mip-NeRF 360 scenes [5]. Our default configuration uses two hidden layers with 16 neurons and $F = 8$ frequency-encoded input features. MAC/G is the number of multiply-accumulate operations needed to evaluate the model for a single Gaussian in our WebGL implementation.

Configuration	PSNR [↑]			SSIM [↑]			LPIPS [↓]			Train [↓]	VRAM [↓]	FPS [↑]	Bytes/G	MAC/G
	Outdoor	Indoor	All	Outdoor	Indoor	All	Outdoor	Indoor	All					
1 hidden layer	25.37	32.15	28.39	0.753	0.933	0.833	0.232	0.219	0.226	4m14s	5.2GiB	432.8	28	304
3 hidden layers	25.25	32.71	28.57	0.745	0.935	0.829	0.238	0.217	0.229	4m24s	5.4GiB	435.7	28	816
32 neurons/layer	25.38	32.67	28.62	0.749	0.935	0.832	0.234	0.217	0.226	4m29s	5.5GiB	449.0	28	2144
$F = 16$ w/o frequency encoding	25.31	32.63	28.56	0.748	0.935	0.831	0.236	0.217	0.227	4m42s	5.7GiB	438.5	44	560
Default w/o fusion	25.34	32.58	28.56	0.748	0.935	0.831	0.234	0.218	0.227	16m08s	5.7GiB	112.1	28	560
Default	25.35	32.51	28.53	0.748	0.935	0.831	0.235	0.218	0.227	4m17s	5.3GiB	443.6	28	560

pixel–Gaussian pairs. We hypothesize that a more expressive decoder relieves the geometry from compensating for missing angular capacity, resulting in more compact primitives and fewer blending operations, consistent with the geometric differences observed in Fig. 2. In WebGL, in contrast, the MAC count translates directly into the cost of the appearance prepass, which matters most on the mobile GPUs in Tab. 2. The default configuration balances both deployment targets, while a single hidden layer offers an even cheaper option for mobile GPUs at a small loss in indoor scenes.

Fusion. The unfused reference path computes the appearance in a separate pass and exchanges inputs and outputs through global memory. Evaluating the default architecture through it yields identical quality within run-to-run variation, confirming the correctness of the fused forward and backward passes. However, it is roughly four times slower to optimize and render, and requires more memory. This gap is what makes the neural model practical, and it explains the common assumption that neural decoding is too expensive for real-time rendering.

C.3. Appearance Decomposition

Fig. 6 shows the base color and residual of all models for a test view of four scenes, extending Fig. 2. Consistent with the results in the main paper, NASGabor achieves the best decomposition: its residual only contains reflections, and its base color produces a reasonable diffuse rendering. For SV, the residual often does too much work and absorbs part of the diffuse color, which could probably be partially resolved by using fewer sites. The compact neural representation usually ranks somewhere between SH and NASGabor. However, in the *Dr Johnson* scene it fails to model the highlight on the desk through the residual, *i.e.*, as a view-dependent effect.

C.4. Per-Scene Results

Tabs. 9 to 11 list per-scene results for all 21 scenes, averaged over five training runs together with the standard deviation across runs. In addition to PSNR, SSIM [58], and LPIPS [62], we report the more recent perceptual metrics $\mathcal{F}$ LIP [2], DISTs [15], and MILO [10]. Several patterns are hidden by the dataset averages in Tab. 1.

Outdoor vs. Indoor. On the outdoor scenes of Mip-NeRF 360, SH achieves the best PSNR on most scenes, and even the model without view-dependent appearance trails only slightly. On the indoor scenes, the picture reverses: the expressive models clearly outperform SH, and the gap to the model without view-dependent appearance grows substantially. View-dependent appearance thus matters primarily in indoor scenes with glossy surfaces and controlled lighting, whereas outdoor scenes are dominated by diffuse content and their remaining error is largely geometric. This is consistent with our findings on decoder capacity in Tab. 8. The neural model follows the expressive models indoors but ranks last among the view-dependent models in SSIM on most outdoor scenes, in line with the weaker regularization noted in Tab. 3.

Run-to-Run Variation. On Mip-NeRF 360, dataset averages are stable to a few hundredths of a dB, whereas Tanks & Temples and Deep Blending are noisier, in particular *train* and *playroom*, the two scenes with the strongest photometric variation. Most notably, the synthetic scenes expose a robustness difference between models that the averages conceal. Whereas SH and the neural model barely vary on every NeRF Synthetic scene, SV fluctuates by several dB on *figus* and *hotdog*, and NASG on *mic*. In these cases, individual runs settle into noticeably worse local minima, which is what Tab. 3 summarizes as robustness. The band-limited SH basis is the most forgiving representation, whereas sharper angular primitives such as Voronoi cells and anisotropic lobes can lock into poor configurations early in the optimization from which gradient descent does not recover.

Perceptual Metrics. $\mathcal{F}$ LIP, DISTs, and MILO largely agree with the established metrics and do not change the conclusions of the main paper. DISTs slightly favors SH and MILO the spherical models on real scenes, whereas $\mathcal{F}$ LIP, like PSNR, favors the neural model on Tanks & Temples and Deep Blending. Across all six metrics and three datasets, no model is consistently superior and the differences remain within a narrow band. This supports the conclusion of the main paper that memory footprint and rendering performance are the more discriminative axes when choosing an appearance model.



Figure 6. Base color and view-dependent residual, visualized as in Fig. 2, for a test view of four scenes. NASGabor achieves the best decomposition, with the residual limited to reflections such as those on the desk in *Dr Johnson*. The SV residual often also models diffuse appearance. The neural model incorrectly represents the highlight on the desk in *Dr Johnson* with the base color instead of the residual.

PSNR↑ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	25.65 ±0.0507	21.66 ±0.0276	27.50 ±0.0226	27.14 ±0.0364	23.29 ±0.0578	31.89 ±0.0919	28.39 ±0.0394	31.47 ±0.0310	32.58 ±0.0373	27.73 ±0.0230
SH	26.19 ±0.0337	22.15 ±0.0238	27.96 ±0.0167	27.73 ±0.0346	23.40 ±0.0275	32.97 ±0.0251	29.49 ±0.0348	32.35 ±0.0354	32.98 ±0.0550	28.36 ±0.0094
SV	26.10 ±0.0368	22.08 ±0.0293	28.04 ±0.0201	27.46 ±0.0383	23.23 ±0.0928	34.11 ±0.0272	30.77 ±0.0379	32.81 ±0.0617	33.16 ±0.0591	28.64 ±0.0266
NASG	26.12 ±0.0150	21.98 ±0.0145	27.94 ±0.0313	27.53 ±0.0628	23.32 ±0.0362	33.61 ±0.0446	30.28 ±0.0328	32.52 ±0.0595	33.20 ±0.0505	28.50 ±0.0117
NASGabor	26.12 ±0.0522	21.96 ±0.0326	27.98 ±0.0160	27.63 ±0.0152	23.30 ±0.0634	33.66 ±0.0807	30.43 ±0.0274	32.60 ±0.0829	33.20 ±0.0814	28.54 ±0.0255
Neural	25.93 ±0.0211	21.87 ±0.0252	28.06 ±0.0344	27.53 ±0.0418	23.28 ±0.0190	33.94 ±0.0662	30.38 ±0.0872	32.69 ±0.0108	33.23 ±0.0616	28.55 ±0.0210
SSIM↑ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	0.791 ±0.0004	0.620 ±0.0003	0.865 ±0.0002	0.796 ±0.0003	0.653 ±0.0008	0.941 ±0.0001	0.905 ±0.0002	0.925 ±0.0002	0.928 ±0.0001	0.825 ±0.0001
SH	0.802 ±0.0002	0.643 ±0.0003	0.874 ±0.0004	0.809 ±0.0004	0.659 ±0.0003	0.949 ±0.0001	0.919 ±0.0001	0.933 ±0.0001	0.932 ±0.0001	0.836 ±0.0001
SV	0.797 ±0.0004	0.638 ±0.0007	0.873 ±0.0002	0.802 ±0.0003	0.652 ±0.0004	0.951 ±0.0001	0.926 ±0.0001	0.934 ±0.0003	0.931 ±0.0001	0.834 ±0.0002
NASG	0.804 ±0.0003	0.639 ±0.0005	0.873 ±0.0004	0.812 ±0.0005	0.660 ±0.0005	0.950 ±0.0002	0.922 ±0.0002	0.932 ±0.0002	0.932 ±0.0001	0.836 ±0.0001
NASGabor	0.804 ±0.0003	0.639 ±0.0002	0.873 ±0.0003	0.813 ±0.0001	0.659 ±0.0003	0.951 ±0.0001	0.923 ±0.0001	0.933 ±0.0002	0.933 ±0.0003	0.836 ±0.0001
Neural	0.788 ±0.0007	0.633 ±0.0007	0.872 ±0.0004	0.798 ±0.0012	0.649 ±0.0007	0.951 ±0.0002	0.922 ±0.0004	0.933 ±0.0002	0.933 ±0.0002	0.831 ±0.0002
LPIPS↓ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	0.196 ±0.0003	0.333 ±0.0007	0.124 ±0.0003	0.216 ±0.0008	0.325 ±0.0008	0.242 ±0.0005	0.249 ±0.0002	0.160 ±0.0003	0.264 ±0.0002	0.234 ±0.0001
SH	0.184 ±0.0004	0.317 ±0.0004	0.112 ±0.0002	0.201 ±0.0007	0.312 ±0.0007	0.232 ±0.0005	0.232 ±0.0003	0.148 ±0.0002	0.258 ±0.0002	0.222 ±0.0001
SV	0.191 ±0.0003	0.322 ±0.0003	0.114 ±0.0002	0.207 ±0.0004	0.322 ±0.0009	0.231 ±0.0003	0.226 ±0.0004	0.147 ±0.0002	0.260 ±0.0001	0.225 ±0.0001
NASG	0.185 ±0.0003	0.318 ±0.0003	0.116 ±0.0003	0.200 ±0.0005	0.318 ±0.0006	0.231 ±0.0002	0.230 ±0.0003	0.151 ±0.0002	0.259 ±0.0002	0.223 ±0.0001
NASGabor	0.185 ±0.0002	0.318 ±0.0006	0.115 ±0.0003	0.200 ±0.0003	0.319 ±0.0007	0.231 ±0.0003	0.229 ±0.0003	0.151 ±0.0002	0.259 ±0.0004	0.223 ±0.0002
Neural	0.194 ±0.0007	0.324 ±0.0007	0.118 ±0.0004	0.209 ±0.0010	0.328 ±0.0013	0.232 ±0.0003	0.232 ±0.0006	0.149 ±0.0003	0.258 ±0.0004	0.227 ±0.0001
FLIP↓ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	0.137 ±0.0024	0.212 ±0.0008	0.121 ±0.0010	0.143 ±0.0016	0.175 ±0.0028	0.083 ±0.0013	0.113 ±0.0011	0.100 ±0.0004	0.085 ±0.0005	0.130 ±0.0005
SH	0.129 ±0.0012	0.201 ±0.0011	0.112 ±0.0004	0.133 ±0.0015	0.167 ±0.0010	0.076 ±0.0005	0.102 ±0.0008	0.089 ±0.0011	0.081 ±0.0005	0.121 ±0.0002
SV	0.131 ±0.0011	0.205 ±0.0009	0.110 ±0.0004	0.135 ±0.0013	0.172 ±0.0067	0.073 ±0.0004	0.093 ±0.0004	0.085 ±0.0004	0.082 ±0.0009	0.121 ±0.0009
NASG	0.128 ±0.0004	0.204 ±0.0005	0.115 ±0.0005	0.140 ±0.0027	0.171 ±0.0019	0.073 ±0.0005	0.096 ±0.0004	0.091 ±0.0009	0.079 ±0.0000	0.122 ±0.0004
NASGabor	0.129 ±0.0011	0.204 ±0.0000	0.114 ±0.0004	0.136 ±0.0008	0.172 ±0.0033	0.074 ±0.0013	0.095 ±0.0004	0.090 ±0.0011	0.079 ±0.0011	0.121 ±0.0003
Neural	0.133 ±0.0027	0.210 ±0.0007	0.113 ±0.0005	0.133 ±0.0009	0.173 ±0.0016	0.073 ±0.0009	0.095 ±0.0011	0.086 ±0.0005	0.079 ±0.0007	0.122 ±0.0005
DISTS↓ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	0.054 ±0.0000	0.113 ±0.0004	0.037 ±0.0004	0.074 ±0.0008	0.115 ±0.0007	0.077 ±0.0004	0.058 ±0.0000	0.052 ±0.0004	0.088 ±0.0000	0.074 ±0.0001
SH	0.045 ±0.0004	0.108 ±0.0004	0.032 ±0.0000	0.067 ±0.0004	0.111 ±0.0007	0.075 ±0.0004	0.056 ±0.0000	0.050 ±0.0000	0.088 ±0.0005	0.070 ±0.0001
SV	0.047 ±0.0000	0.110 ±0.0005	0.032 ±0.0005	0.070 ±0.0011	0.116 ±0.0008	0.075 ±0.0000	0.053 ±0.0000	0.050 ±0.0000	0.088 ±0.0000	0.071 ±0.0003
NASG	0.049 ±0.0004	0.110 ±0.0004	0.034 ±0.0000	0.070 ±0.0009	0.115 ±0.0009	0.076 ±0.0000	0.055 ±0.0004	0.050 ±0.0000	0.088 ±0.0000	0.072 ±0.0002
NASGabor	0.048 ±0.0005	0.111 ±0.0009	0.034 ±0.0004	0.068 ±0.0005	0.115 ±0.0000	0.076 ±0.0000	0.054 ±0.0004	0.050 ±0.0000	0.088 ±0.0000	0.072 ±0.0001
Neural	0.049 ±0.0000	0.112 ±0.0007	0.035 ±0.0004	0.072 ±0.0011	0.121 ±0.0011	0.075 ±0.0005	0.055 ±0.0004	0.050 ±0.0000	0.088 ±0.0004	0.073 ±0.0001
MILO↑ on Mip-NeRF 360 [5]										
Method	Bicycle	Flowers	Garden	Stump	Treehill	Bonsai	Counter	Kitchen	Room	Average
None	3.403 ±0.0038	2.579 ±0.0035	3.660 ±0.0019	3.185 ±0.0011	2.868 ±0.0031	3.641 ±0.0018	3.100 ±0.0041	3.626 ±0.0026	3.468 ±0.0020	3.281 ±0.0015
SH	3.495 ±0.0034	2.637 ±0.0055	3.730 ±0.0025	3.281 ±0.0039	2.925 ±0.0038	3.784 ±0.0040	3.259 ±0.0068	3.706 ±0.0074	3.512 ±0.0020	3.370 ±0.0016
SV	3.476 ±0.0046	2.630 ±0.0051	3.728 ±0.0054	3.246 ±0.0037	2.865 ±0.0076	3.823 ±0.0029	3.363 ±0.0100	3.723 ±0.0037	3.523 ±0.0027	3.375 ±0.0013
NASG	3.475 ±0.0021	2.633 ±0.0011	3.720 ±0.0049	3.278 ±0.0023	2.901 ±0.0034	3.795 ±0.0049	3.311 ±0.0099	3.703 ±0.0050	3.516 ±0.0043	3.370 ±0.0011
NASGabor	3.475 ±0.0033	2.632 ±0.0063	3.723 ±0.0041	3.282 ±0.0022	2.895 ±0.0040	3.796 ±0.0019	3.318 ±0.0069	3.702 ±0.0053	3.517 ±0.0057	3.371 ±0.0020
Neural	3.438 ±0.0047	2.606 ±0.0072	3.720 ±0.0033	3.233 ±0.0040	2.881 ±0.0015	3.801 ±0.0044	3.293 ±0.0065	3.709 ±0.0033	3.524 ±0.0052	3.356 ±0.0011

Table 9. Per-scene image quality metrics on Mip-NeRF 360 [5], averaged over five training runs with the standard deviation across runs. The three best results are highlighted in green in descending order of saturation.

PSNR↑ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	24.10 ±0.1286	26.78 ±0.0207	29.34 ±0.0503	30.39 ±0.1534	27.65 ±0.0207
SH	24.48 ±0.1266	27.13 ±0.0268	29.43 ±0.0867	30.80 ±0.0714	27.96 ±0.0294
SV	24.54 ±0.0993	27.37 ±0.0263	29.33 ±0.0875	30.68 ±0.1853	27.98 ±0.0357
NASG	24.39 ±0.2059	27.21 ±0.0164	29.55 ±0.1153	30.73 ±0.1417	27.97 ±0.0396
NASGabor	24.53 ±0.1562	27.22 ±0.0288	29.51 ±0.0779	30.60 ±0.3347	27.96 ±0.0855
Neural	24.49 ±0.0959	27.23 ±0.0056	29.72 ±0.2136	31.08 ±0.1384	28.13 ±0.0702
SSIM↑ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	0.846 ±0.0017	0.898 ±0.0002	0.900 ±0.0004	0.906 ±0.0017	0.888 ±0.0004
SH	0.856 ±0.0005	0.904 ±0.0002	0.902 ±0.0006	0.910 ±0.0002	0.893 ±0.0001
SV	0.856 ±0.0010	0.904 ±0.0001	0.900 ±0.0006	0.910 ±0.0007	0.893 ±0.0002
NASG	0.853 ±0.0015	0.905 ±0.0001	0.904 ±0.0004	0.912 ±0.0002	0.893 ±0.0003
NASGabor	0.853 ±0.0010	0.905 ±0.0001	0.904 ±0.0004	0.911 ±0.0015	0.893 ±0.0004
Neural	0.854 ±0.0015	0.904 ±0.0001	0.903 ±0.0004	0.910 ±0.0020	0.893 ±0.0008
LPIPS↓ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	0.211 ±0.0018	0.127 ±0.0001	0.309 ±0.0002	0.300 ±0.0007	0.237 ±0.0005
SH	0.197 ±0.0008	0.117 ±0.0003	0.305 ±0.0004	0.290 ±0.0005	0.228 ±0.0003
SV	0.199 ±0.0006	0.121 ±0.0002	0.311 ±0.0005	0.296 ±0.0010	0.232 ±0.0003
NASG	0.204 ±0.0009	0.117 ±0.0002	0.301 ±0.0004	0.288 ±0.0007	0.227 ±0.0002
NASGabor	0.203 ±0.0004	0.117 ±0.0003	0.300 ±0.0005	0.287 ±0.0008	0.227 ±0.0004
Neural	0.202 ±0.0013	0.118 ±0.0002	0.303 ±0.0018	0.289 ±0.0001	0.228 ±0.0004
FLIP ↓ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	0.218 ±0.0053	0.117 ±0.0000	0.117 ±0.0007	0.140 ±0.0043	0.148 ±0.0008
SH	0.209 ±0.0037	0.110 ±0.0011	0.119 ±0.0011	0.134 ±0.0013	0.143 ±0.0007
SV	0.211 ±0.0038	0.108 ±0.0005	0.120 ±0.0015	0.134 ±0.0021	0.143 ±0.0010
NASG	0.214 ±0.0073	0.109 ±0.0011	0.116 ±0.0005	0.135 ±0.0030	0.144 ±0.0015
NASGabor	0.209 ±0.0050	0.109 ±0.0005	0.115 ±0.0013	0.138 ±0.0064	0.143 ±0.0016
Neural	0.213 ±0.0038	0.110 ±0.0005	0.114 ±0.0040	0.129 ±0.0013	0.141 ±0.0014
DISTS↓ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	0.073 ±0.0005	0.035 ±0.0000	0.121 ±0.0000	0.113 ±0.0008	0.086 ±0.0001
SH	0.067 ±0.0004	0.032 ±0.0000	0.118 ±0.0000	0.107 ±0.0000	0.081 ±0.0001
SV	0.068 ±0.0004	0.033 ±0.0005	0.121 ±0.0005	0.111 ±0.0004	0.083 ±0.0002
NASG	0.071 ±0.0005	0.032 ±0.0000	0.117 ±0.0000	0.107 ±0.0000	0.082 ±0.0001
NASGabor	0.071 ±0.0005	0.032 ±0.0000	0.117 ±0.0005	0.107 ±0.0008	0.082 ±0.0003
Neural	0.070 ±0.0007	0.032 ±0.0000	0.120 ±0.0039	0.107 ±0.0004	0.082 ±0.0008
MILO↑ on Tanks & Temples [34] / Deep Blending [24]					
Method	<i>Train</i>	<i>Truck</i>	<i>Drjohnson</i>	<i>Playroom</i>	<i>Average</i>
None	3.088 ±0.0164	3.472 ±0.0038	3.180 ±0.0036	3.391 ±0.0090	3.283 ±0.0037
SH	3.169 ±0.0115	3.544 ±0.0050	3.212 ±0.0021	3.426 ±0.0061	3.338 ±0.0032
SV	3.187 ±0.0068	3.568 ±0.0028	3.188 ±0.0042	3.419 ±0.0133	3.340 ±0.0031
NASG	3.145 ±0.0098	3.548 ±0.0053	3.234 ±0.0024	3.452 ±0.0053	3.345 ±0.0021
NASGabor	3.148 ±0.0076	3.552 ±0.0034	3.234 ±0.0045	3.451 ±0.0109	3.346 ±0.0040
Neural	3.157 ±0.0136	3.547 ±0.0034	3.230 ±0.0057	3.444 ±0.0084	3.345 ±0.0044

Table 10. Per-scene image quality metrics on Tanks & Temples [34] / Deep Blending [24], averaged over five training runs with the standard deviation across runs. The three best results are highlighted in green in descending order of saturation.

PSNR↑ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	33.60 ±0.0199	25.40 ±0.0390	32.04 ±0.0585	33.49 ±0.0451	34.83 ±0.0125	28.22 ±0.0821	34.70 ±0.0382	29.36 ±0.1582	31.46 ±0.0873
SH	35.98 ±0.0111	26.28 ±0.0259	35.46 ±0.3506	38.35 ±0.0920	36.14 ±0.0059	30.91 ±0.0060	37.48 ±0.0638	31.44 ±0.2776	34.01 ±0.0478
SV	36.34 ±0.0074	27.00 ±0.0316	30.65 ±2.3306	36.53 ±4.2332	36.36 ±0.0174	31.08 ±0.0142	37.80 ±0.0144	31.88 ±0.0505	33.45 ±0.6306
NASG	34.77 ±0.0090	26.34 ±0.0389	33.40 ±0.0802	36.35 ±0.9041	35.60 ±0.0560	28.94 ±0.0239	35.25 ±3.1155	30.87 ±0.1773	32.69 ±0.3724
NASGabor	34.79 ±0.0145	26.32 ±0.0196	33.35 ±0.0367	37.03 ±0.6098	35.70 ±0.0144	28.96 ±0.0865	36.59 ±0.0413	30.89 ±0.1202	32.95 ±0.0763
Neural	35.70 ±0.0637	26.75 ±0.0596	35.60 ±0.0372	38.35 ±0.0588	36.10 ±0.0389	30.88 ±0.0439	37.68 ±0.0339	31.28 ±0.1358	34.04 ±0.0241
SSIM↑ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	0.983 ±0.0000	0.949 ±0.0002	0.978 ±0.0001	0.964 ±0.0057	0.981 ±0.0001	0.948 ±0.0005	0.989 ±0.0001	0.881 ±0.0041	0.959 ±0.0012
SH	0.989 ±0.0000	0.956 ±0.0002	0.987 ±0.0008	0.987 ±0.0005	0.984 ±0.0000	0.964 ±0.0001	0.994 ±0.0001	0.903 ±0.0022	0.971 ±0.0003
SV	0.989 ±0.0000	0.959 ±0.0002	0.970 ±0.0086	0.978 ±0.0210	0.984 ±0.0000	0.966 ±0.0001	0.994 ±0.0000	0.906 ±0.0014	0.968 ±0.0029
NASG	0.986 ±0.0000	0.955 ±0.0002	0.983 ±0.0001	0.980 ±0.0070	0.983 ±0.0001	0.953 ±0.0002	0.987 ±0.0126	0.897 ±0.0022	0.966 ±0.0017
NASGabor	0.986 ±0.0000	0.955 ±0.0001	0.983 ±0.0000	0.983 ±0.0021	0.983 ±0.0000	0.953 ±0.0004	0.993 ±0.0000	0.898 ±0.0014	0.967 ±0.0004
Neural	0.988 ±0.0001	0.955 ±0.0003	0.987 ±0.0001	0.987 ±0.0002	0.984 ±0.0001	0.963 ±0.0001	0.994 ±0.0001	0.899 ±0.0024	0.970 ±0.0003
LPIPS↓ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	0.020 ±0.0000	0.052 ±0.0001	0.021 ±0.0001	0.042 ±0.0022	0.023 ±0.0001	0.060 ±0.0005	0.014 ±0.0001	0.148 ±0.0020	0.048 ±0.0006
SH	0.014 ±0.0000	0.043 ±0.0002	0.014 ±0.0010	0.021 ±0.0003	0.019 ±0.0000	0.042 ±0.0001	0.008 ±0.0001	0.130 ±0.0006	0.036 ±0.0001
SV	0.013 ±0.0000	0.038 ±0.0001	0.034 ±0.0095	0.029 ±0.0192	0.019 ±0.0000	0.041 ±0.0001	0.007 ±0.0000	0.126 ±0.0011	0.038 ±0.0028
NASG	0.017 ±0.0000	0.046 ±0.0002	0.018 ±0.0000	0.029 ±0.0048	0.021 ±0.0001	0.055 ±0.0004	0.018 ±0.0191	0.137 ±0.0009	0.043 ±0.0024
NASGabor	0.017 ±0.0000	0.046 ±0.0001	0.018 ±0.0001	0.026 ±0.0024	0.021 ±0.0000	0.055 ±0.0005	0.009 ±0.0001	0.136 ±0.0006	0.041 ±0.0003
Neural	0.015 ±0.0001	0.043 ±0.0001	0.014 ±0.0001	0.022 ±0.0002	0.021 ±0.0001	0.045 ±0.0002	0.008 ±0.0001	0.135 ±0.0009	0.038 ±0.0001
FID↓ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	0.032 ±0.0004	0.064 ±0.0005	0.039 ±0.0005	0.039 ±0.0039	0.032 ±0.0000	0.053 ±0.0004	0.018 ±0.0000	0.072 ±0.0016	0.044 ±0.0007
SH	0.023 ±0.0000	0.057 ±0.0005	0.027 ±0.0009	0.023 ±0.0000	0.028 ±0.0000	0.035 ±0.0000	0.011 ±0.0000	0.055 ±0.0019	0.032 ±0.0003
SV	0.022 ±0.0000	0.052 ±0.0000	0.040 ±0.0063	0.030 ±0.0177	0.029 ±0.0005	0.034 ±0.0000	0.011 ±0.0000	0.052 ±0.0005	0.034 ±0.0024
NASG	0.027 ±0.0000	0.057 ±0.0004	0.034 ±0.0004	0.028 ±0.0046	0.030 ±0.0000	0.047 ±0.0005	0.017 ±0.0080	0.060 ±0.0015	0.038 ±0.0011
NASGabor	0.027 ±0.0000	0.058 ±0.0000	0.034 ±0.0005	0.026 ±0.0016	0.030 ±0.0000	0.047 ±0.0007	0.013 ±0.0000	0.060 ±0.0013	0.037 ±0.0003
Neural	0.026 ±0.0004	0.058 ±0.0005	0.028 ±0.0000	0.023 ±0.0000	0.030 ±0.0000	0.038 ±0.0000	0.012 ±0.0000	0.058 ±0.0012	0.034 ±0.0001
DISTS↓ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	0.038 ±0.0005	0.075 ±0.0005	0.048 ±0.0004	0.065 ±0.0114	0.034 ±0.0000	0.095 ±0.0008	0.034 ±0.0004	0.109 ±0.0103	0.062 ±0.0027
SH	0.032 ±0.0004	0.057 ±0.0004	0.030 ±0.0011	0.039 ±0.0000	0.030 ±0.0000	0.068 ±0.0004	0.020 ±0.0000	0.089 ±0.0016	0.046 ±0.0002
SV	0.027 ±0.0000	0.047 ±0.0004	0.096 ±0.0271	0.045 ±0.0195	0.028 ±0.0000	0.063 ±0.0004	0.019 ±0.0000	0.084 ±0.0004	0.051 ±0.0045
NASG	0.037 ±0.0004	0.065 ±0.0004	0.037 ±0.0005	0.050 ±0.0077	0.032 ±0.0000	0.088 ±0.0009	0.045 ±0.0483	0.095 ±0.0016	0.056 ±0.0060
NASGabor	0.037 ±0.0000	0.066 ±0.0004	0.038 ±0.0004	0.046 ±0.0022	0.032 ±0.0000	0.088 ±0.0008	0.023 ±0.0000	0.094 ±0.0012	0.053 ±0.0004
Neural	0.032 ±0.0000	0.059 ±0.0005	0.031 ±0.0004	0.040 ±0.0004	0.030 ±0.0005	0.074 ±0.0005	0.020 ±0.0000	0.093 ±0.0012	0.047 ±0.0002
MILO↑ on NeRF Synthetic [41]									
Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
None	4.133 ±0.0013	3.311 ±0.0020	4.150 ±0.0023	3.928 ±0.0246	4.183 ±0.0011	3.277 ±0.0025	4.179 ±0.0028	3.378 ±0.0194	3.817 ±0.0058
SH	4.255 ±0.0019	3.461 ±0.0022	4.262 ±0.0147	4.205 ±0.0019	4.217 ±0.0009	3.619 ±0.0014	4.288 ±0.0004	3.636 ±0.0090	3.993 ±0.0017
SV	4.260 ±0.0004	3.574 ±0.0044	3.664 ±0.2948	4.081 ±0.2748	4.218 ±0.0010	3.667 ±0.0048	4.292 ±0.0013	3.673 ±0.0062	3.929 ±0.0556
NASG	4.214 ±0.0008	3.404 ±0.0061	4.214 ±0.0011	4.114 ±0.0648	4.208 ±0.0005	3.328 ±0.0038	4.068 ±0.4383	3.566 ±0.0110	3.890 ±0.0533
NASGabor	4.214 ±0.0029	3.408 ±0.0062	4.212 ±0.0005	4.150 ±0.0313	4.211 ±0.0008	3.331 ±0.0041	4.264 ±0.0011	3.573 ±0.0033	3.920 ±0.0037
Neural	4.238 ±0.0029	3.480 ±0.0050	4.268 ±0.0008	4.204 ±0.0023	4.212 ±0.0011	3.610 ±0.0031	4.287 ±0.0021	3.603 ±0.0048	3.988 ±0.0017

Table 11. Per-scene image quality metrics on NeRF Synthetic [41], averaged over five training runs with the standard deviation across runs. The three best results are highlighted in green in descending order of saturation.