Spherical Voronoi
Directional Appearance as a Differentiable Partition of the Sphere

Supplementary Material

Supplementary Overview

In Section 6 we expand our deferred rendering formulation, explaining probe querying, near- and far-field illumination, and the role of roughness in controlling the Spherical Voronoi temperature. We then describe in Section 7 our acceleration scheme for Spherical Voronoi evaluation, detailing the cubemap-based partitioning and truncated softmax used to reduce computational cost. Next, in Section 8 we present an analysis of the quality–efficiency tradeoff associated with varying the number of interpolated probes and sites. Section 9 evaluates all bases in a controlled environment map fitting setting, isolating the contribution of SV from the complexity of a full 3DGS pipeline. Section 10 provides additional qualitative visualizations: the adaptive SV decomposition learned when fitting environment maps, the effect of the temperature parameter τ on reflectance sharpness, and the optimization behavior of learnable light probes during training. We also clarify the evaluation protocol adopted to ensure fair comparisons across datasets and prior work in Section 11. Finally, in Section 12 we include additional quantitative results, including SV integration into other baselines and full per-scene metrics across all datasets.

6. Additional Details on Deferred Rendering

In this section, we expand the reflection formulation introduced in the main paper and provide a derivation of the near-field term C_n, the far field term C_f and the spatial blending factor α appearing in Equation (7).

6.1. Geometry Pass

Following the deferred strategy illustrated in Figure 7, we first rasterize all 2DGS primitives once to produce the per-pixel attributes required for shading. For each pixel (u, v), the geometry pass outputs:

- world position $P(u, v) \in \mathbb{R}^3$,
- surface normal $N(u, v) \in \mathbb{S}^2$,
- diffuse color $D(u, v) \in \mathbb{R}^3$,
- roughness $R(u, v) \in [0, 1]$.

These attributes are produced by splatting the corresponding Gaussian parameters, using the same weighted blending used during accumulation in standard 2DGS rendering.

6.2. Lighting Pass

Given the geometry buffer, the lighting pass computes the final shaded color:

$$C(u, v) = D(u, v) + C_{\text{spec}}(u, v), \tag{12}$$

where $C_{\text{spec}}(u, v)$ is the specular term defined in Equation (7). We report it here for brevity:

$$C_{\text{spec}}(u, v) = \alpha(u, v)C_n(u, v) + (1 - \alpha(u, v))C_f(u, v). \tag{13}$$

Both illumination terms depend on the reflected view direction:

$$\omega_r(u, v) = 2(\omega \cdot N(u, v))N(u, v) - \omega. \tag{14}$$

In the following sections we describe how each component is computed.

6.3. Near-field Illumination

Near-field reflections are modeled by a set of learnable light probes distributed in 3D space. Each probe i has:

- Position $p_i \in \mathbb{R}^3$,
- Spherical Voronoi function $f_i(\omega)$,
- Blend parameter $\alpha_i \in [0, 1]$.

Probe selection. We identify the probes that are most relevant for shading a given pixel (u, v). We select the k closest probes using Euclidean nearest-neighbor search:

$$\mathcal{N}(u, v) = k\text{NN}(P(u, v)). \tag{15}$$

This ensures that each pixel only interacts with probes that lie within a meaningful spatial neighborhood, reflecting the intuition that local geometry affects appearance only within a limited radius.

Distance-based weighting. Not all selected probes contribute equally. Probes closer to the shading point should have a stronger influence than those further away. We therefore assign each probe a weight inversely proportional to its distance:

$$w_i(u, v) = \frac{1}{\|P(u, v) - p_i\| + \epsilon}, \tag{16}$$

which is normalized as:

$$\tilde{w}(u, v) = \frac{w_i(u, v)}{\sum_{j \in \mathcal{N}(u,v)} w_j(u, v)}. \tag{17}$$

The resulting weights vary smoothly throughout the image, producing spatially consistent transitions between lighting regions.

Evaluating and aggregating near-field illumination. Each probe stores a SV representation of the reflected radiance at its location, which we query in the reflected direction $\omega_r(u, v)$ to obtain a directional estimate of the local illumination. The response of the selected probes is then combined using the normalized spatial weights $\tilde{w}_i(u, v)$. Formally, the near-field term is:

$$C_n(u, v) = \sum_{i \in \mathcal{N}(u,v)} \tilde{w}_i(u, v) f_i(\omega_r(u, v)). \tag{18}$$

This operation blends the directional information encoded at each probe according to its proximity to the shading point.

6.4. Far-field Illumination

While near-field probes capture illumination effects produced by nearby geometry, many reflective surfaces are dominated by light coming from far more distant parts of the environment, such as skylight, outdoor structures, windows, or large surrounding objects. To represent this global component, we use a learnable environment cubemap. Once the reflected direction $\omega_r(u, v)$ has been computed, obtaining the far-field illumination is straightforward: we simply sample the cubemap at that direction,

$$C_f(u, v) = \text{cubemap}(\omega_r(u, v)). \tag{19}$$

6.5. Blending Factor

The relative influence between near-field and far-field illumination must vary across the scene. Close to complex geometry, reflections are more sensitive to local variations and should rely predominantly on the probes. In open regions, distant illumination captured by the cubemap becomes more important. To achieve this adaptive behavior, each probe carries a learned blend weight α_i, which express how strongly that probe favors near-field effects. At shading time, the per-pixel blend factor $\alpha(u, v)$ is obtained by interpolating these probe parameters using the same spatial weights $\tilde{w}_i(u, v)$ computed above:

$$\alpha(u, v) = \sum_{i \in \mathcal{N}(u,v)} \tilde{w}_i(u, v) \alpha_i. \tag{20}$$

This produces a smooth spatial field that naturally transitions between locally dominated and globally dominated reflection regimes.

6.6. Roughness and Temperature

The sharpness of the Spherical Voronoi representation is controlled by the temperature parameter τ. Each Gaussian primitive stores its own roughness value R, which is splatted into a per-pixel roughness map $\mathbf{R}(\mathbf{x})$ during the geometry pass. Roughness determines how sharp or smooth the directional function should be. To couple the reflectance model with Spherical Voronoi expressivity, we linearly map roughness to temperature:

$$\tau(u, v) = (1 - R(u, v))\tau_{\text{max}} + R(u, v)\tau_{\text{min}}, \tag{21}$$

where we empirically set $\tau_{\text{min}} = 0.2$ and $\tau_{\text{max}} = 1500$. As a result:

- *low roughness* ($R \approx 0$) produces crisp, mirror-like reflections through large temperatures,
- *high roughness* ($R \approx 1$) yields broad, diffuse responses via small temperatures.

7. Speeding up Spherical Voronoi

A naïve evaluation of our Spherical Voronoi representation requires evaluating all K sites for every queried direction ω. While this is not a problem in the view-direction parameterization context, when modeling reflections which require a large number of sites, it becomes impractical. Concretely, for a function

$$f_{\text{SV}}(\omega; \tau, s, c) = \sum_{k=1}^{K} w_k(\omega; \tau)\, c_k, \tag{22}$$

where τ is the temperature controlling the sharpness of the partition, $s = \{s_k\}$ are the unit vectors defining the angular locations of the SV sites, and $c = \{c_k\}$ are their associated radiance values, with

$$w_k(\omega; \tau) = \frac{\exp(\tau\, s_k \cdot \omega)}{\sum_{k'=1}^{K} \exp(\tau\, s_{k'} \cdot \omega)}, \tag{23}$$

the cost of a single evaluation scales linearly with the number of sites K. When using thousands of sites per function, this becomes a major bottleneck during rendering, especially when evaluating directional appearance at every pixel. To mitigate this cost, we introduce a simple acceleration scheme that exploits the fact that *only a small subset of sites is relevant for any given direction*. Our key idea is to partition the unit sphere using a low-resolution cubemap, and to pre-assign to each texel a fixed set of *candidate* sites. At runtime, the softmax is restricted to this candidate set instead of all sites.

7.1. Cubemap-Based Partition of the Sphere

We discretize the sphere using a cubemap of fixed resolution. Each texel t in the cubemap is associated with a direction $\hat{\omega}_t$ corresponding to its center. In a preprocessing step, for each texel t we select a small subset of sites $\mathcal{S}(t) \subset \{1, ..., K\}$ that are the most relevant for directions

around $\hat{\omega}_t$. A natural choice is to pick the sites with highest similarity to $\hat{\omega}_t$, *i.e.*:

$$\mathcal{S}(t) = \text{TopK}_k(s_k \cdot \hat{\omega}_t), \tag{24}$$

where TopK denotes the set of indices of the k closest sites. This yields a lookup table that maps each cubemap texel to a much smaller set of candidate sites. Since the Spherical Voronoi sites are optimized jointly with the rest of the model, the cubemap-to-site assignment $\mathcal{S}(t)$ becomes outdated over the course of training. To account for this, we periodically recompute it. In all our experiments, we rebuild the assignment every 500 optimization steps, which we found to be a good compromise between accuracy of the approximation and preprocessing overhead.

7.2. Softmax Approximation

At rendering time, given a direction ω_r, we determine the cubemap texel $t(\omega_r)$ that contains ω_r. Instead of evaluating the SV weights over all sites, we restrict the computation to the precomputed candidate set. The accelerated SV evaluation becomes:

$$f_{\text{SV}}^{\text{fast}}(\omega_r) = \sum_{k \in \mathcal{S}(t(\omega_r))} \tilde{w}_k(\omega_r; \tau)\, c_k, \tag{25}$$

where the truncated weights are defined as:

$$\tilde{w}_k(\omega_r; \tau) = \frac{\exp(\tau\, s_k \cdot \omega_r)}{\sum_{k' \in \mathcal{S}(t(\omega_r))} \exp(\tau\, s_{k'} \cdot \omega_r)}. \tag{26}$$

In other words, we approximate the full softmax over K sites with a local softmax over a small, direction-dependent subset of sites. Intuitively, for sufficiently fine cubemap resolution and moderately large candidate set size, the omitted sites have negligible contribution to the softmax, as their dot product $s_k \cdot \omega_r$ is much smaller. This reduces the per-direction complexity from $\mathcal{O}(K)$ to $\mathcal{O}(|\mathcal{S}(t)|)$.

8. Quality-Efficiency Tradeoff

Number of SV sites (radiance modeling). We use 8 sites, matching the parameter count of order-3 spherical harmonics for a fair comparison. Table 5 shows results varying the number of sites from 1 to 16: performance saturates around 8–12 sites with diminishing returns beyond, while inference cost increases linearly.

Probe and site capacity (reflection modeling). We evaluate the effect of kernel capacity by varying the cardinalities $|S(K)|$ and $|N(K)|$. As shown in Figure 9, increasing these values improves reconstruction quality but reduces inference speed, exhibiting a clear quality–efficiency tradeoff. Notably, setting $|N(K)| = 32$ yields a small quality gain but nearly halves the FPS, showing rapidly diminishing returns. We therefore adopt $|S(K)| = 8$ and $|N(K)| = 8$ as a balanced configuration that preserves most of the quality benefits while maintaining real-time performance.

Table 5. Effect of the number of sites on quality and efficiency, averaged over *bonsai* and *garden* scenes of *Mip-NeRF 360*.

	#sites	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑	Train Time (min) ↓
SV	1	29.64	0.905	0.186	295	21
	2	30.74	0.912	0.178	268	22
	4	31.04	0.914	0.177	226	25
	8	31.21	0.915	0.175	173	32
	12	31.23	0.915	0.173	138	39
	16	31.28	0.915	0.173	116	47
SB	-	30.71	0.913	0.177	148	28
SG	-	30.37	0.911	0.176	104	31
SH	-	30.39	0.912	0.176	107	30

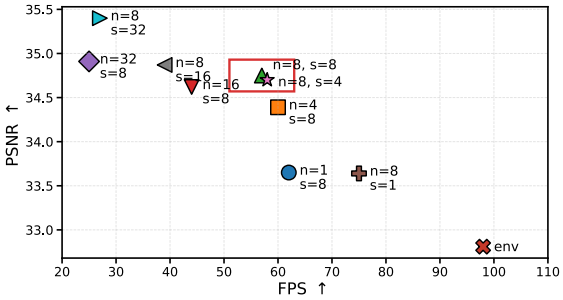

Figure 9. **Quality–Efficiency Tradeoff** - Increasing the kernel capacity ($|\mathcal{N}(K)|$ and $|\mathcal{S}(K)|$; abbreviated as n and s in the plot) leads to higher rendering quality but lower inference speed, highlighting the fundamental tradeoff between accuracy and efficiency. Results are reported for the *coffee* scene of the Ref-NeRF dataset. *Env* denotes the configuration without probes.

9. Environment Map Fitting

To isolate the contribution of the SV representation from the complexity of a full 3DGS pipeline, we evaluate all bases in a controlled setting: fitting a known environment map onto a reflective sphere. We select environment maps that contain both high-frequency content (trees, road, grass) and low-frequency regions (sky), providing a challenging yet balanced test case. We fix the parameter budget across all methods and report PSNR, SSIM, and LPIPS at convergence in Table 6 and Figure 10. SV consistently outperforms SH and SG across all scenes and metrics, with notably cleaner error maps.

10. Visualization of SV Behavior and Probe Dynamics

In this section, we provide qualitative visualizations that further illustrate the behavior of our SV representation and the dynamics of learnable light probes. Figure 11 shows the adaptive decomposition of the spherical domain when learning an environment map. Figure 12 visualizes how increasing τ sharpens SV lobes and enhances specular detail.

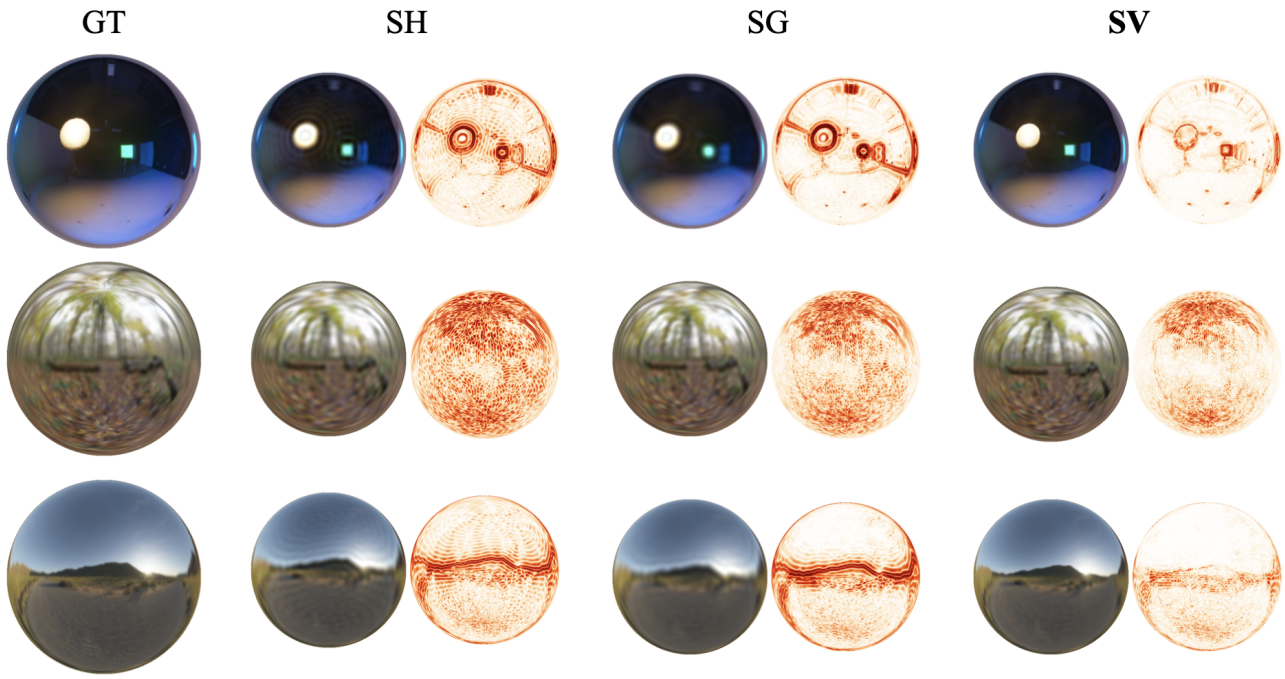

Figure 10. **Environment map fitting.** For each method we show the rendered reflective sphere (left) and the corresponding error map (right). All methods are optimized under the same parameter budget.

Table 6. **Environment map fitting.** PSNR, SSIM, and LPIPS at convergence under a fixed parameter budget.

scene	method	PSNR ↑	SSIM↑	LPIPS↓
studio	**SV**	38.58	0.978	0.087
	SG	32.66	0.940	0.180
	SH	31.37	0.946	0.185
forest	**SV**	36.42	0.944	0.087
	SG	34.17	0.917	0.136
	SH	31.78	0.872	0.179
road	**SV**	40.36	0.969	0.106
	SG	34.03	0.932	0.174
	SH	32.51	0.930	0.190

Figure 13 illustrates the optimization trajectory of the light probes.

11. Clarifications on Evaluation

To ensure a fair and consistent comparison across methods, we reviewed the evaluation protocols commonly used for the Mip-NeRF 360 and Ref-Real datasets. We found that some prior works rely on testing setups that deviate from the dataset guidelines, which can lead to inflated performance metrics. For Mip-NeRF 360, Beta Splatting [14] evaluates models on images that are downsampled on-the-fly from the high-resolution originals (using the `--r` flag in 3DGS codebase). This produces smoother inputs that are easier to fit, increasing PSNR by approximately 0.5 dB on average compared to using the official downsampled images provided in the dataset. Additionally, Beta Splatting employs a checkpoint-selection mechanism: starting from iteration 15k, the model is evaluated on the test set every 500 steps, and the best checkpoint is reported as the final result. While practical, this introduces a form of test-set selection that is not fully aligned with standard evaluation practice. A similar issue appears in the Ref-Real results reported by Ref-GS [34], where non-standard downsampling choices also deviate from the dataset's recommended evaluation setup. For a fully fair and reproducible evaluation, we strictly follow the dataset recommendations. In Mip-NeRF 360, we use `images_2` for indoor scenes and `images_4` for outdoor scenes. For Ref-Real, we use `images_4` for *garden spheres* and *toy car*, and `images_8` for *sedan*. All results in this paper follow these settings.

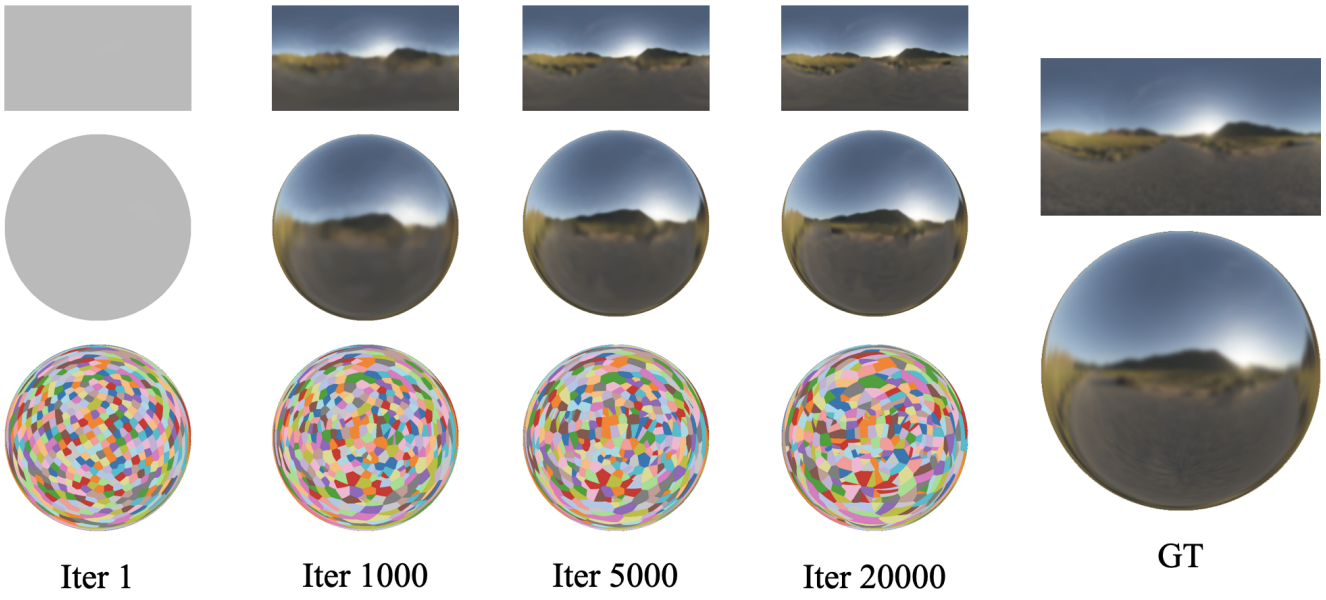

Figure 11. **SV evolution during training (3D)** - Top: predicted environment map in lat-long format. Middle: its rendering on a reflective sphere. Bottom: corresponding SV tessellation (colored by random site IDs). Across training iterations, the SV sites reorganize into a structured decomposition of the sphere. Ground truth is shown on the right.

Figure 12. **Effect of the SV temperature τ (3D)** - Increasing τ progressively sharpens the SV lobes, transitioning from an almost uniform shading ($\tau = 0$) to increasingly crisp, mirror-like reflections. Renderings shown for $\tau \in \{0, 5, 50, 100, 250\}$

12. Additional Results

12.1. Modeling Radiance

Table 7. **Radiance Modeling with SV Voronoi** - Applying SV to existing Gaussian Splatting baselines yields consistent improvements across datasets.

		Vanilla			Ours (SV)		
	Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360	*3DGS-mcmc*	27.92	0.833	0.234	28.33	0.832	0.234
	2DGS	26.86	0.798	0.301	27.41	0.800	0.300
	Beta-Splatting	28.12	0.831	0.238	28.57	0.835	0.230
NeRF-Synthetic	*3DGS-mcmc*	33.77	0.972	0.036	34.21	0.972	0.034
	2DGS	33.17	0.968	0.041	33.57	0.969	0.039
	Beta-Splatting	34.10	0.971	0.034	34.53	0.973	0.032
Tanks&Temples	*3DGS-mcmc*	24.24	0.863	0.190	24.45	0.863	0.192
	Beta-Splatting	24.54	0.871	0.171	24.75	0.871	0.171
DeepBlending	*3DGS-mcmc*	29.55	0.901	0.320	29.64	0.905	0.314
	Beta-Splatting	29.56	0.907	0.316	30.48	0.915	0.296

SV can be seamlessly integrated into other Gaussian Splatting pipelines. In Table 7, we report the improvements obtained when augmenting two popular baselines—3DGS-MCMC [10] and 2DGS [7]—with our SV modeling. The consistent gains across both methods demonstrate that SV serves as a general and effective mechanism for enhancing radiance modeling in Gaussian-based representations.

12.2. Per-scene Results

We provide a detailed per-scene evaluation for all datasets used in the main paper. Table 8 reports results on radiance-dominated datasets (Mip-NeRF 360, NeRF-Synthetic, Tanks&Temples, and Deep Blending), while Table 9 presents results for reflection-heavy datasets (Ref-NeRF, Glossy Synthetic, and Ref-Real).

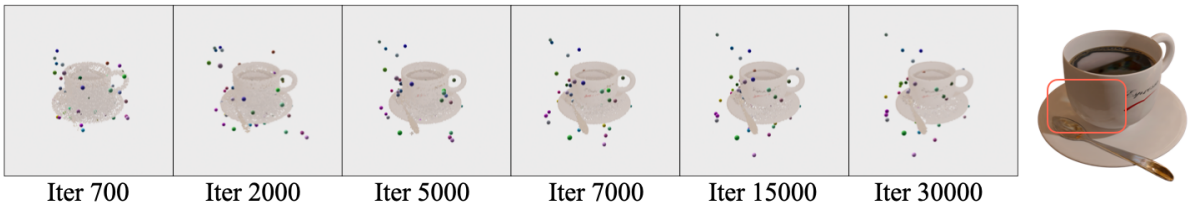

Figure 13. **Evolution of learnable light probes during training** - The probes are initialized at random positions and gradually migrate toward regions exhibiting strong near-field specular interactions. Throughout optimization, they consistently cluster around the reflective side of the object (e.g., the spoon-facing region in the Coffee scene of the Ref-NeRF dataset), which corresponds to an area rich in local specular highlights and interreflections.

Table 8. **Per-scene radiance quality** - Rendering metrics reported per scene across all evaluated datasets.

	Scene	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360	bicycle	25.63	0.791	0.196
	bonsai	34.68	0.957	0.230
	counter	30.97	0.928	0.225
	flowers	22.21	0.639	0.335
	garden	27.74	0.872	0.120
	kitchen	32.57	0.934	0.150
	room	32.97	0.936	0.259
	stump	27.25	0.803	0.212
	treehill	23.08	0.651	0.338
	mean	28.57	0.835	0.230
NeRF-Synthetic	chair	36.85	0.989	0.013
	drums	26.96	0.958	0.039
	ficus	37.01	0.991	0.009
	hotdog	38.31	0.988	0.021
	lego	36.99	0.986	0.016
	materials	30.87	0.965	0.039
	mic	37.60	0.994	0.006
	ship	31.68	0.911	0.117
	mean	34.53	0.973	0.032
Tanks& Temples	train	22.95	0.841	0.214
	truck	26.55	0.900	0.128
	mean	24.75	0.871	0.171
Deep Blending	drjohnson	29.92	0.913	0.301
	playroom	31.04	0.916	0.292
	mean	30.48	0.915	0.296

Table 9. **Per-scene reflection quality** - Rendering metrics reported per scene across all evaluated datasets.

	Scene	PSNR ↑	SSIM ↑	LPIPS ↓
Ref-NeRF	ball	39.40	0.987	0.082
	car	31.26	0.968	0.029
	coffee	34.91	0.973	0.082
	helmet	35.63	0.983	0.031
	teapot	47.59	0.997	0.007
	toaster	27.73	0.950	0.070
	mean	36.09	0.976	0.050
Glossy Synthetic	bell	32.22	0.963	0.044
	cat	33.46	0.975	0.033
	luyu	30.22	0.950	0.042
	potion	33.74	0.961	0.061
	tbell	30.90	0.966	0.050
	teapot	27.25	0.954	0.048
	mean	31.30	0.962	0.046
Ref-Real	gardenspheres	21.97	0.578	0.275
	sedan	25.94	0.757	0.215
	toycar	23.83	0.641	0.243
	mean	23.91	0.659	0.244