

A FULL IMPLEMENTATION DETAILS

We optimize the 2D Gaussian primitives in image space using the Adam Kingma (2014) optimizer, where the learning rate for image-space coordinates $\mathbf{x}_{2D}$ is set to 0.001 multiplied by the mean of the image dimensions (width and height), with learning rates of 0.01 for color $\mathbf{c}_{2D}$, 0.02 for opacity α_{2D} , 0.001 for scale $\mathbf{s}_{2D}$, and 0.02 for rotation angle r_{2D} . Every 200 iterations, 20% of Gaussian primitives are added, while those with opacity below 0.005 or covering fewer than 25 pixels are pruned. For supplementary Gaussians, the initial value of T is set to $7\bar{\theta}_i/\pi$, constrained within the range of 0.01 to 1.0.

Prior to projecting 2D Gaussian kernels into world space, the sampling pixels are clustered to remove foreground or background pixels. Upon obtaining world space coordinates corresponding to pixels covered by Gaussians, these coordinates are classified using the single linkage merging algorithm Sibson (1973), which operates by iteratively merging the closest cluster pairs based on the minimum distance between any two points from distinct clusters. The process continues until either a single cluster remains or a termination threshold is satisfied. The initial termination threshold is determined by the depth value of the Gaussian kernel’s central pixel, specifically by computing the world-space distance δd between two adjacent pixels at this depth, with the threshold set to $5 \cdot \delta d$. In cases of clustering failure, the threshold is multiplied by 1.5 and the procedure is repeated. Pseudocode is provided in Appendix C.

During the final joint optimization stage, learning rates are configured as follows: 0.000016 for position, 0.001 for diffuse color, 0.001/20 for higher-order spherical harmonic coefficients, 0.02 for opacity, 0.002 for scaling, 0.0005 for rotation, 0.001 for opacity lobe orientation $\mathbf{d}$, 0.0002 for opacity lobe width T , and 0.002 for sharpness β . Concurrently, SparseAdam is employed to optimize the opacity of original Gaussian primitives visible in the current viewpoint. All experiments were conducted on a workstation equipped with an Intel i9-13900K CPU and an NVIDIA RTX 4090 GPU. On the Mip-NeRF 360 dataset, the training and projection of 2D Gaussians averages approximately 15 minutes per scene, as detailed in Table 6.

Table 6: The time consumption for the post-enhancement stage on the Mip-NeRF 360 and NeRF Synthetic datasets is reported as follows. It should be noted that using multiple GPUs can significantly accelerate this process, as the 2D-stage training operates independently on each image.

Dataset	Mip-NeRF 360	NeRF Synthetic
2D Training and Projection	742 s	178 s
Joint Optimization	168 s	32 s

B ANALYSIS ON OPACITY LOBE

While DBS Liu et al. (2025) replaces spherical harmonics with spherical beta functions in their formulation, our approach fundamentally differs by applying generalized Phong shading to opacity modulation rather than radiance modeling. The implementation of view-dependent opacity introduces greater stability and flexibility in radiance reconstruction, because the opacity lobe can decouple each individual Gaussian kernel’s contribution to the final exitant radiance. This distinction is empirically validated through a controlled experiment simulating three overlapping Gaussian kernels in Fig. 5, where we compare two configurations: 1) optimization with spherical beta functions governing color distributions while maintaining fixed opacity values, versus 2) our methodology employing view-dependent opacity modulation with fixed color magnitudes. Under identical opacity settings ($\alpha = 0.5$ for all kernels). The spherical beta approach requires exponentially increasing color magnitudes for subsurface Gaussians to compensate for occlusion effects from overlying kernels (Fig. 5, top-left), creating instability because kernel depth ordering varies dynamically during optimization and view changes. Conversely, our method (second row) achieves target radiance distributions through adaptive opacity variations while maintaining constant color magnitudes, demonstrating remarkable resilience to permutation of depth sorting. This comparative analysis conclusively demonstrates that our Gaussian kernel formulation exhibits substantially greater stability and flexibility than DBS’s design.

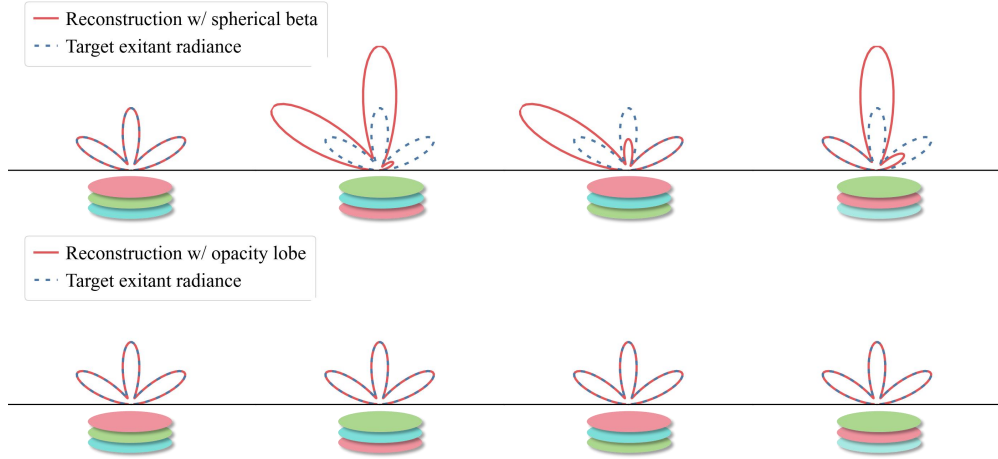


Figure 5: Comparison between Spherical Beta functions in DBS Liu et al. (2025) and opacity lobe. The three Gaussians in the first row model color using three differently oriented spherical beta functions, with maximum function values of 1, 2, and 4, all having an opacity of 0.5. The kernels in the second row use three differently oriented opacity lobes, with the same maximum amplitude and identical color intensity. Our method is more stable and flexible in reconstructing outgoing radiance, as it is less affected by the ordering of Gaussian kernels and has greater numerical stability.



Figure 6: Qualitative comparison on our synthetic shiny dataset rendered using environment maps with complex lighting conditions.

To validate that, we built a synthetic shiny dataset under extreme illumination conditions, as shown in Fig. 6. We demonstrated improved reconstruction fidelity that scales with the complexity of surface exitant radiance distributions, which supports the superiority of our enhanced Gaussians in stacking opacity lobes across different kernels.

```

1 def inverse_splatting(gaussian_2d, depth_map, camera):
2     pixels_world = pixels_covered(gaussian_2d, depth_map, camera)
3     # Step 1. Foreground-background separation
4     threshold = sample_tex(depth_map, gaussian_2d.center)
5     do:
6         threshold = threshold * A
7         groups = cluster_points(threshold, pixels_world)
8         # The PCA algorithm requires at least three input data points.
9         while groups.min_size() < 3
10        # Step2. WPCA
11        filtered_points, INCLUDE_CENTER = prime_group(groups)
12        rotation, scaling = WPCA(filtered_points)
13        # Gaussian center included in the filtered points
14        if INCLUDE_CENTER:
15            depth = sample_tex(depth_map, gaussian_2d.center)
16        else:
17            depth = weighted_mean(filtered_points.depth, filtered_points.
18                                weight)
19        position = depth_to_world(depth, gaussian_2d.center, camera)
20        # Step3. Scaling calibration
21        Q = project(rotation, scaling, position, camera)
22        k = sqrt((Q * gaussian_2d.cov).sum() / frobenius_norm(Q) ** 2)
23        scaling = k * scaling
24    return rotation, scaling, position

```

Listing 1: Pseudocode for our inverse splatting algorithm.

C INVERSE GAUSSIAN SPLATTING

By leveraging the rendered depth map and camera parameters, we project the 2D Gaussians from image space into world space. Fig. 7 compares the rendered Gaussians from a single training image before and after projecting them into world space. This process is implemented through three steps to ensure reasonable and accurate projection results, with pseudocode provided in List 1.

D ADDITIONAL COMPARISONS AND RESULTS

Fig. 8 presents more qualitative analysis of our full model. Table 8, 9, and 10 present the rendering metrics of each scene in Table 1 of the main paper, enhanced with our supplementary kernels. For outdoor scenes in Mip-NeRF 360, our method achieves approximately 0.1 dB PSNR improvement, while indoor scenes with complex lighting and material variations exhibit 1-2 dB gains. Tables 11, 12 and 13 quantify performance variations before and after radiance field augmentation, while Table 14 and 15 presents a comparison of the memory footprint for the scenes in the Mip-NeRF 360 dataset. The proposed opacity lobe introduces minimal overhead, requiring only 5 parameters per instance, with $\sim 10\%$ of primitives storing these additional parameters. This design ensures negligible memory impact compared to baseline implementations.

Fig. 9 shows more results of our diffuse-specular decomposition. To investigate the performance, we further conducted a qualitative comparison with DBS’s Liu et al. (2025) decomposition in Fig. 10. Because the dataset lacks ground-truth specular data, quantitative metrics cannot be reported. Visually, our method produces a specular component comparable to that of DBS.

We compare the training time, memory usage, and FPS with Spec-Gaussian Yang et al. (2024) in Table 7. Since Spec-Gaussian employs a neural network for real-time rendering, it significantly impacts the FPS. This also affects the scene reconstruction speed. Our method requires less memory and achieves a much higher FPS.

Table 7: Analysis of training time, memory usage, and FPS. "Total time" represents the combined duration of the original scene reconstruction and post-processing enhancement.

Method	Mip-NeRF 360			NeRF Synthetic		
	Total(Post-proc.) time	Memory	FPS	Total(Post-proc.) time	Memory	FPS
Spec-Gaussian	44.8 min	882 MB	43	12.0 min	75 MB	136
Ours (3DGS,sh=3)	27.9(15.0) min	691 MB	110	6.0(3.4) min	68 MB	446
Ours (MCMC,sh=2)	31.9(14.9)min	467 MB	96	7.0(3.2)min	44 MB	434
Ours (MCMC,sh=3)	25.3(15.2)min	722 MB	90	7.9(3.5) min	68 MB	414

Table 8: Quantitative results of rendering quality per scene on Mip-NeRF 360 dataset.

	metrics	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	mean
Zip-NeRF	PSNR $\uparrow$	25.80	22.40	28.20	27.55	23.89	32.65	29.38	32.50	34.46	28.54
	SSIM $\uparrow$	0.769	0.642	0.860	0.800	0.681	0.925	0.902	0.928	0.949	0.828
	LPIPS $\downarrow$	0.208	0.273	0.118	0.193	0.242	0.196	0.185	0.116	0.173	0.189
3DGS-MCMC	PSNR $\uparrow$	26.15	22.12	28.16	27.80	23.31	32.48	29.51	32.27	32.88	28.29
	SSIM $\uparrow$	0.81	0.64	0.89	0.82	0.66	0.94	0.92	0.94	0.95	0.84
	LPIPS $\downarrow$	0.18	0.31	0.10	0.19	0.29	0.25	0.22	0.14	0.22	0.21
DBS (30k)	PSNR $\uparrow$	26.04	22.44	28.15	27.56	23.49	32.83	30.36	32.61	33.90	28.60
	SSIM $\uparrow$	0.804	0.650	0.882	0.815	0.676	0.941	0.930	0.939	0.957	0.844
	LPIPS $\downarrow$	0.169	0.308	0.096	0.184	0.293	0.168	0.154	0.110	0.156	0.182
Ours 3DGS sh=3	PSNR $\uparrow$	25.85	21.81	28.21	27.30	23.00	32.39	30.48	32.24	34.26	28.39
	SSIM $\uparrow$	0.793	0.633	0.881	0.796	0.660	0.931	0.924	0.936	0.954	0.834
	LPIPS $\downarrow$	0.174	0.294	0.093	0.186	0.289	0.183	0.166	0.110	0.165	0.184
Ours MCMC sh=2	PSNR $\uparrow$	26.31	22.34	28.35	27.97	23.49	33.32	30.64	33.01	34.60	28.89
	SSIM $\uparrow$	0.815	0.658	0.887	0.827	0.680	0.942	0.928	0.940	0.957	0.848
	LPIPS $\downarrow$	0.158	0.273	0.088	0.159	0.262	0.169	0.160	0.108	0.161	0.171
Ours MCMC sh=3	PSNR $\uparrow$	26.32	22.44	28.36	27.90	23.46	33.45	30.82	33.15	34.78	28.96
	SSIM $\uparrow$	0.814	0.663	0.887	0.825	0.680	0.942	0.930	0.941	0.958	0.849
	LPIPS $\downarrow$	0.158	0.270	0.086	0.161	0.259	0.170	0.158	0.106	0.159	0.170

E LIMITATIONS

The back-projection of 2D Gaussians into world space relies on depth maps of training cameras, where high-quality depth estimation could improve projection accuracy. However, this work does not incorporate geometric constraints or deep neural networks to generate refined depth maps, resulting in projected Gaussian kernels that do not accurately conform to object surfaces. Additionally, the designed two-stage optimization strategy prolongs scene reconstruction time, which could be alleviated by integrating the initialization of newly added Gaussians with the training process of the original scene. Furthermore, although a fixed ration between added and original Gaussians is maintained to accommodate most scenarios, this approach may introduce redundancy in scenes dominated by diffuse materials. Adaptively determining the number of additional Gaussians per scene remains a promising direction for future improvements.

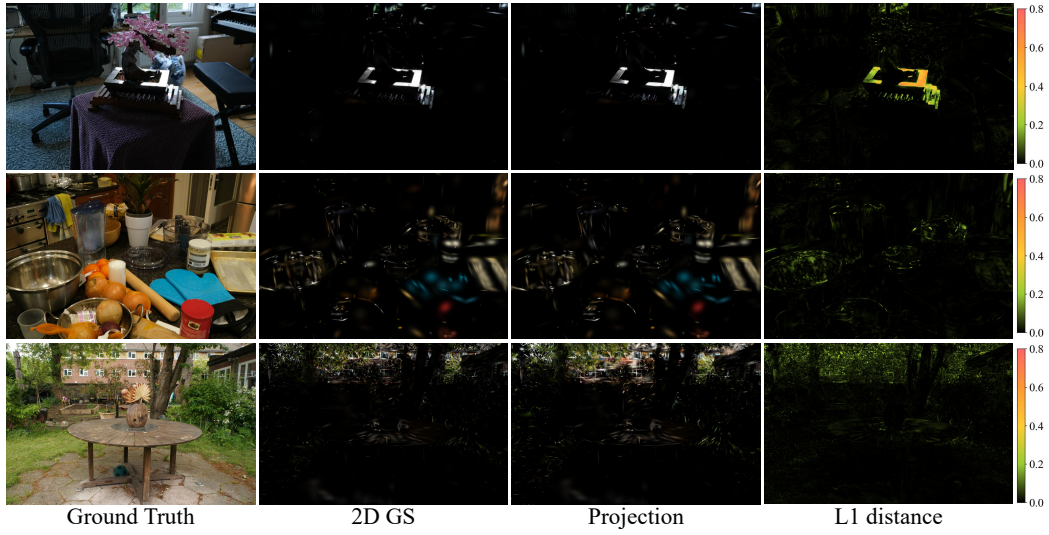


Figure 7: Projection results of 2D Gaussians. We present the optimization results of the 2D primitives in image space along with the rendered outputs after back-projection into world space. The 4th column images display the L1 distance between the renderings before and after enhancement. The additional Gaussians can also help restore the background reconstruction for outdoor scenes.

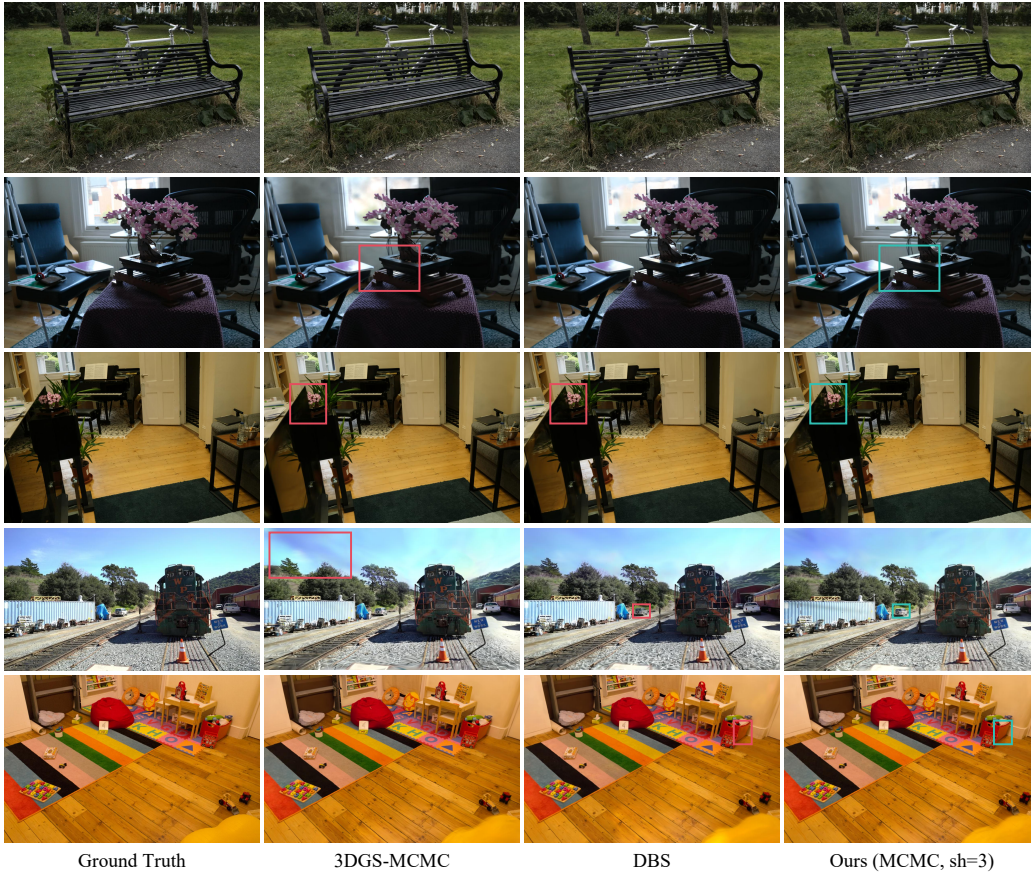


Figure 8: Qualitative analysis across real-captured scenes.



Figure 9: Diffuse specular decomposition. We convert both the diffuse color rendering of the original Gaussians and the enhanced rendering into linear color space to decouple the diffuse and specular components.

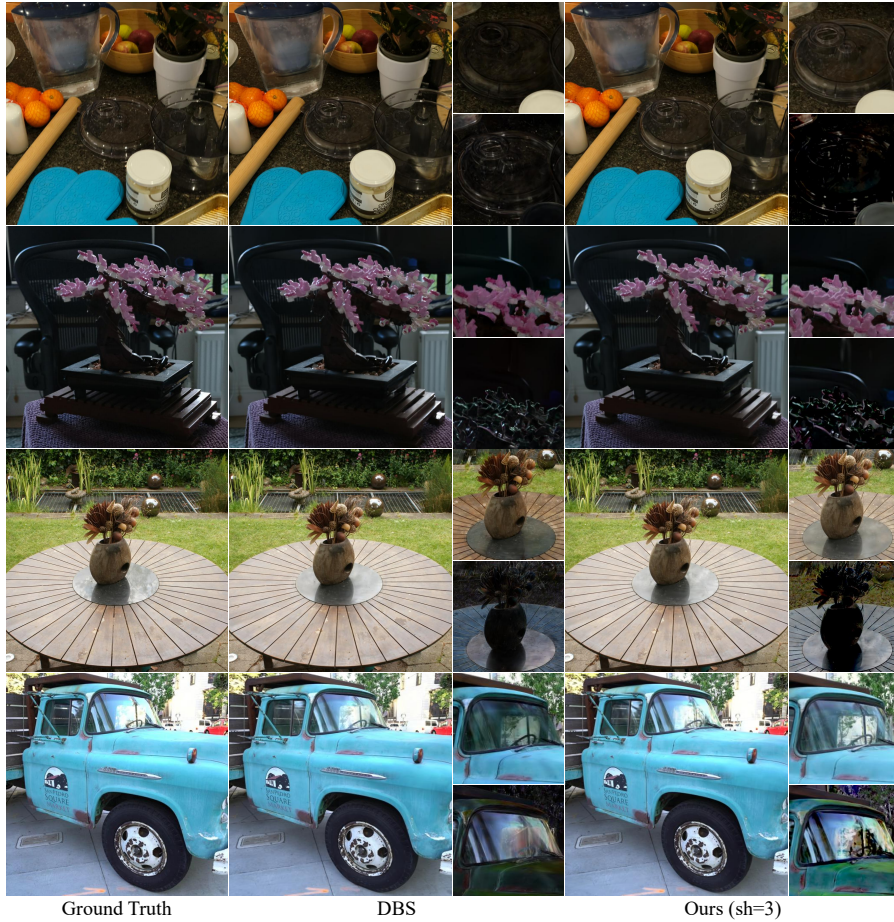


Figure 10: Qualitative comparison with DBS on diffuse-specular decomposition. The top-right and bottom-right sub-figures display the diffuse and specular components, respectively.

Table 9: Quantitative results of rendering quality per scene on Tanks and Temples and Deep Blending dataset.

	metrics	Tanks and Temples			Deep Blending		
		truck	train	mean	drjohnson	playroom	mean
Zip-NeRF	PSNR $\uparrow$	-	-	-	-	-	-
	SSIM $\uparrow$	-	-	-	-	-	-
	LPIPS $\downarrow$	-	-	-	-	-	-
3DGS-MCMC	PSNR $\uparrow$	26.11	22.47	24.29	29.00	30.33	29.67
	SSIM $\uparrow$	0.89	0.83	0.86	0.89	0.90	0.89
	LPIPS $\downarrow$	0.14	0.24	0.19	0.33	0.31	0.32
DBS (30k)	PSNR $\uparrow$	26.44	23.13	24.79	29.39	30.82	30.10
	SSIM $\uparrow$	0.897	0.839	0.868	0.908	0.912	0.910
	LPIPS $\downarrow$	0.110	0.185	0.147	0.240	0.240	0.240
Ours 3DGS sh=3	PSNR $\uparrow$	25.71	22.09	23.90	29.31	30.50	29.90
	SSIM $\uparrow$	0.886	0.826	0.856	0.902	0.905	0.903
	LPIPS $\downarrow$	0.131	0.185	0.158	0.236	0.237	0.237
Ours MCMC sh=2	PSNR $\uparrow$	26.78	23.30	25.04	29.90	30.77	30.33
	SSIM $\uparrow$	0.899	0.842	0.871	0.904	0.914	0.909
	LPIPS $\downarrow$	0.110	0.182	0.146	0.234	0.236	0.235
Ours MCMC sh=3	PSNR $\uparrow$	26.86	23.27	25.06	29.85	30.58	30.22
	SSIM $\uparrow$	0.900	0.844	0.872	0.904	0.913	0.909
	LPIPS $\downarrow$	0.106	0.181	0.144	0.235	0.237	0.236

Table 10: Quantitative results of rendering quality per scene on NeRF synthetic dataset.

	metrics	chair	drums	figus	hotdog	lego	materials	mic	ship	mean
Zip-NeRF	PSNR $\uparrow$	34.84	25.84	33.90	37.14	34.84	31.66	35.15	31.38	33.10
	SSIM $\uparrow$	0.983	0.944	0.985	0.984	0.980	0.969	0.991	0.929	0.971
	LPIPS $\downarrow$	0.017	0.050	0.015	0.020	0.019	0.032	0.007	0.091	0.031
3DGS-MCMC	PSNR $\uparrow$	36.51	26.29	35.07	37.82	36.01	30.59	37.29	30.82	33.80
	SSIM $\uparrow$	0.99	0.95	0.99	0.99	0.98	0.96	0.99	0.91	0.97
	LPIPS $\downarrow$	0.02	0.04	0.01	0.02	0.02	0.04	0.01	0.12	0.04
DBS (30k)	PSNR $\uparrow$	36.74	26.78	36.75	38.85	37.12	31.12	37.66	32.13	34.64
	SSIM $\uparrow$	0.990	0.958	0.990	0.988	0.985	0.966	0.994	0.910	0.973
	LPIPS $\downarrow$	0.010	0.033	0.010	0.015	0.014	0.032	0.005	0.103	0.028
Ours 3DGS sh=3	PSNR $\uparrow$	35.73	26.59	35.71	38.21	36.39	30.74	37.13	31.63	34.02
	SSIM $\uparrow$	0.988	0.955	0.987	0.985	0.982	0.961	0.993	0.896	0.969
	LPIPS $\downarrow$	0.011	0.036	0.012	0.020	0.016	0.036	0.006	0.113	0.031
Ours MCMC sh=2	PSNR $\uparrow$	36.05	26.39	35.47	38.34	36.44	30.69	37.27	31.59	34.03
	SSIM $\uparrow$	0.989	0.955	0.987	0.987	0.985	0.963	0.993	0.904	0.970
	LPIPS $\downarrow$	0.011	0.038	0.012	0.016	0.016	0.035	0.006	0.106	0.030
Ours MCMC sh=3	PSNR $\uparrow$	36.35	26.57	35.87	38.72	36.67	31.11	37.79	31.68	34.35
	SSIM $\uparrow$	0.989	0.956	0.988	0.988	0.985	0.965	0.994	0.902	0.971
	LPIPS $\downarrow$	0.011	0.036	0.011	0.016	0.015	0.033	0.006	0.106	0.029

Table 11: Quantitative analysis of rendering quality improvements after enhancement per scene on Mip-NeRF 360 dataset.

Method	Metrics	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Mean
Ours (3DGS, sh=3)	Δ PSNR $\uparrow$	0.110	-0.067	0.307	0.089	0.168	0.720	1.230	0.806	1.765	0.570
	Δ SSIM $\uparrow$ (10^{-3})	3.26	5.17	2.84	4.77	3.19	2.77	5.49	2.11	5.25	3.87
	Δ LPIPS $\downarrow$ (10^{-3})	-5.09	-28.6	-3.00	-9.09	-14.9	-5.04	-8.08	-3.16	-7.23	-9.35
Ours (MCMC, sh=2)	Δ PSNR $\uparrow$	0.227	0.007	0.397	0.212	0.085	0.967	1.622	0.976	2.225	0.746
	Δ SSIM $\uparrow$ (10^{-3})	4.44	3.39	4.83	3.25	2.96	4.52	8.66	3.14	6.39	4.62
	Δ LPIPS $\downarrow$ (10^{-3})	-5.58	-12.5	-5.24	-4.62	-7.04	-6.97	-11.7	-4.94	-7.93	-7.38
Ours (MCMC, sh=3)	Δ PSNR $\uparrow$	0.174	0.003	0.239	0.093	0.079	0.978	1.376	0.798	1.975	0.635
	Δ SSIM $\uparrow$ (10^{-3})	3.24	2.75	3.48	2.24	1.60	3.88	6.60	2.59	5.11	3.50
	Δ LPIPS $\downarrow$ (10^{-3})	-6.26	-12.3	-4.57	-4.11	-6.91	-6.35	-9.66	-4.33	-6.83	-6.81

Table 12: Quantitative analysis of rendering quality improvements after enhancement per scene on NeRF synthetic dataset.

Method	Metrics	chair	drums	figus	hotdog	lego	materials	mic	ship	Mean
Ours (3DGS, sh=3)	Δ PSNR $\uparrow$	0.130	0.239	0.132	0.080	0.160	0.137	0.146	0.113	0.142
	Δ SSIM $\uparrow$ (10^{-4})	1.82	4.45	1.97	2.37	3.57	4.38	1.50	6.20	3.28
	Δ LPIPS $\downarrow$ (10^{-4})	-0.724	-5.99	-1.73	-5.69	-3.56	-3.32	-1.25	-6.17	-3.55
Ours (MCMC, sh=2)	Δ PSNR $\uparrow$	0.265	0.288	0.419	0.231	0.364	0.400	0.464	0.235	0.333
	Δ SSIM $\uparrow$ (10^{-4})	3.42	4.74	6.54	5.85	6.45	12.5	4.36	3.86	5.97
	Δ LPIPS $\downarrow$ (10^{-4})	-4.92	-13.0	-5.97	-7.44	-6.47	-11.5	-3.69	-13.9	-8.36
Ours (MCMC, sh=3)	Δ PSNR $\uparrow$	0.174	0.251	0.252	0.150	0.271	0.185	0.278	0.114	0.209
	Δ SSIM $\uparrow$ (10^{-4})	2.38	4.72	3.70	2.57	4.88	5.31	2.51	3.04	3.64
	Δ LPIPS $\downarrow$ (10^{-4})	-4.15	-9.39	-4.50	-5.79	-5.41	-6.68	-2.32	-9.15	-5.92

Table 13: Quantitative analysis of rendering quality improvements after enhancement per scene.

Method	Metrics	Tanks and Temples			Deep Blending		
		truck	train	Mean	drjohnson	playroom	Mean
Ours (3DGS, sh=3)	Δ PSNR $\uparrow$	0.348	0.356	0.352	0.169	0.059	0.114
	Δ SSIM $\uparrow$ (10^{-3})	2.90	3.04	2.97	1.29	0.537	0.915
	Δ LPIPS $\downarrow$ (10^{-3})	-7.44	-3.34	-5.39	-5.96	-6.31	-6.14
Ours (MCMC, sh=2)	Δ PSNR $\uparrow$	0.460	0.522	0.491	0.256	0.126	0.191
	Δ SSIM $\uparrow$ (10^{-3})	4.07	6.18	5.12	2.17	1.28	1.72
	Δ LPIPS $\downarrow$ (10^{-3})	-7.40	-7.87	-7.63	-6.14	-7.87	-7.01
Ours (MCMC, sh=3)	Δ PSNR $\uparrow$	0.442	0.618	0.530	0.231	0.238	0.235
	Δ SSIM $\uparrow$ (10^{-3})	3.62	6.58	5.10	1.83	1.35	1.59
	Δ LPIPS $\downarrow$ (10^{-3})	-7.45	-7.00	-7.22	-5.80	-8.08	-6.94

Table 14: Quantitative results of memory footprint per scene on Mip-NeRF 360 dataset.

Method	Metrics	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Mean
3DGS	Primitives (10^6)	4.8	3.0	4.1	4.5	3.3	1.4	1.2	1.5	1.2	2.8
	Memory (MB)	1081	675	928	1017	732	316	263	328	275	624
DBS	Primitives (10^6)	6.0	3.0	5.0	4.5	3.5	1.5	1.5	1.5	1.5	3.1
	Memory (MB)	618	309	515	464	361	155	155	155	155	320
Ours (3DGS, sh=3)	Primitives (10^6)	5.3	3.3	4.5	5.0	3.6	1.5	1.3	1.6	1.3	3.0
	Memory (MB)	1198	748	1029	1128	811	350	291	363	305	691

Table 15: Quantitative results of memory footprint per scene on Mip-NeRF 360 dataset.

		bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Mean
Method	Primitives (10^6)	5.9	3.3	5.2	4.8	3.7	1.5	1.2	1.8	1.3	3.2
3DGS-MCMC	Memory (MB)	1328	743	1170	1069	833	338	270	405	293	716
Ours (MCMC, sh=2)		865	484	763	697	543	220	176	264	191	467
Ours (MCMC, sh=3)		1338	748	1179	1077	839	340	272	408	295	722