SVG-IR: Spatially-Varying Gaussian Splatting for Inverse Rendering

Supplementary Material

We propose SVG-IR, a novel inverse rendering framework based on our proposed Spatially-varying Gaussian representation with spatially-varying material attributes. Besides we introduce our physically-based illumination to better decouple material properties and illumination. In this supplementary material, we provide implementation details Sec. 8, along with additional results Sec. 9 and ablation study Sec. 10.

8. Implementation details

Spatially-varying Gaussian representation. As described in Sec. 4.1. In our Spatially-varying Gaussian, we define several Gaussian vertices with different material attributes on a single Gaussian primitive. The Gaussian vertices are parameterized on the tangent space of each Gaussian defined by the rotation matrix R and the scaling matrix S. In implementation, we create a square on each Gaussian surfel, where the length of each side of the square is determined by twice the scaling S along the two axes.

Ray tracing. We perform ray tracing on Gaussian surfels in Sec. 4.3 to obtain the incoming radiance of each Gaussian. When constructing the BVH, we treat each Gaussian as an elliptical disk, where the lengths of the two axes are set to three times the scaling factor, and the thickness of each disk is defined as 1×10^{-12}. We perform uniform sample on the up hemisphere of each Gaussian K (K=64 in our experiments) times to get K sampled directions. For each Gaussian, we emit rays along the K sampled directions as well as along the upper hemisphere to perform ray intersections. When a ray hits a Gaussian, the new ray's starting point is the hit point, plus an offset of ϵ (set as 0.05) times the ray direction, and the ray tracing continues from there. The radiance and transmittance are accumulated as Eqn. (12) and Eqn. (3). The ray tracing results are stored in the micro-buffers of each Gaussian for later direct queries, eliminating the need to re-trace the rays. Besides, during ray tracing, we also record the index I of the first Gaussian hit and its coordinates in the tangent space U, which are stored in the Gaussian's buffer as well.

One-bounce indirect illumination. We replace the indirect illumination from the radiance colors of Gaussians with one-bounce indirect illumination in Sec. 4.4. Thanks to the micro-buffers stored in ray-tracing, we can utilize the index buffers I to query the bounce between Gaussians quickly with the tangent space coordinates U to determine the weights of interpolation. Then we can compute the one-bounce indirect illumination in Eqn. (13) with a fast speed.

Radiance consistency loss. We leverage one-bounce indirect illumination as a supervision when training by sampling the specular direction as

$$k_j = \mathrm{argmax}(< \omega_o^k, 2\boldsymbol{N}_o^g - \omega_i > \&\ V_j^k = 0) \quad (16)$$

where ω_o^k represents the view direction on Gaussian j. We only sample on directions that are not visible to direct light, ensuring the presence of indirect lighting from bounces between Gaussians. Thus, We can compute the radiance consistency loss as Eqn. (14). As described in Sec. 8, the index micro buffers also reduce the computation time this loss less than 1 ms.

Loss details. We train Gaussian vertex attributes using loss terms in Eqn. (15). The $\mathcal{L}_1$ and $\mathcal{L}_{\text{ssim}}$ reprensent L_1 loss and SSIM loss between rendered image and ground truth, which are commonly used rendering loss by previous methods [9, 19, 22]. $\mathcal{L}_{\text{ec}}$ is the radiance consistency loss defined in Eqn. (14). $\mathcal{L}_{s,a}$ is TV-loss on albedo for smoothness, defined as

$$\begin{aligned} \Delta_{ij}^{\hat{\alpha}} =& \exp\left(-|I_{i,j} - I_{i-1,j}|\right)(\hat{\alpha}_{i,j} - \hat{\alpha}_{i-1,j})^2 + \\ & \exp\left(-|I_{i,j} - I_{i,j-1}|\right)(\hat{\alpha}_{i,j} - \hat{\alpha}_{i,j-1})^2, \\ \mathcal{L}_{s,a} =& \frac{1}{|\hat{\alpha}|}\sum_{i,j}\Delta_{ij}^{\hat{\alpha}}, \end{aligned} \quad (17)$$

where $\hat{\alpha}$ is the albedo map obtained by the SVG splatting in Sec. 4.2. $\mathcal{L}_{s,r}$ is the TV-loss on roughness similar to $\mathcal{L}_{s,a}$. $\mathcal{L}_n$ is the normal consistency loss in Gaussian surfels [6] by

$$\mathcal{L}_n = (1 - \hat{n}^\top \hat{n}_{\hat{D}}) \quad (18)$$

where $\hat{n}$ is normal map and $\hat{n}_{\hat{D}}$ is the pseudo normal map from the depth map. $\mathcal{L}_{\text{reg},n}$ is L_2 regular term of normal offsets from Gaussian Shader [13] as

$$\mathcal{L}_{\text{reg},n} = ||\Delta N^{\{M\}}||^2 \quad (19)$$

The loss weights $\{\lambda_1, \lambda_{\text{ssim}}, \lambda_{\text{rc}}, \lambda_n, \lambda_{s,a}, \lambda_{s,r}, \lambda_{reg,n}\}$ are set as $\{0.9, 0.1, 0.05, 0.02, 0.1, 0.05, 0.01\}$.

9. More results

Results on TensoIR Synthetic dataset. We show more inverse rendering results on Figs. 12 to 15. The metrics on albedo and normal are shown in Tab. 5. Relightable

Table 5. Comparison of albedo and normal on TensoIR Synthetic and ADT datasets. Numbers in red represent the best performance, while orange numbers denote the second best.

	Method	Albedo PSNR↑ / SSIM↑ / LPIPS↓	Normal MAE↓
TensoIR	MII	27.293 / 0.933 / 0.101	5.076
	TensoIR	29.275 / 0.950 / 0.087	4.098
	Gsshader	25.026 / 0.923 / 0.087	5.757
	GS-IR	30.286 / 0.941 / 0.084	5.341
	RelightGS	28.537 / 0.922 / 0.087	5.064
	Ours	30.341 / 0.951 / 0.074	4.358
ADT	MII	29.150 / 0.952 / 0.068	3.027
	TensoIR	29.295 / 0.954 / 0.056	2.688
	Gsshader	30.432 / 0.960 / 0.036	1.995
	GS-IR	32.711 / 0.968 / 0.037	2.665
	RelightGS	21.047 / 0.911 / 0.039	2.179
	Ours	33.630 / 0.980 / 0.023	1.703

3DGS [9] conducts over-smooth normal and albedo. GaussianShader [13] produces unnatural relighting results. Due to the residual color terms and the approximation of PBR. GS-IR [22] produces coarse normals and albedo with baked-in lighting effects. TensoIR [16] lacks the details in rendering, e.g. the texture on the bread in the hotdog scene. Our method leverages Spatially-varying Gaussians and physically-based illumination to enhance representational capacity and lighting decoupling, achieving high-quality results on both relighting and NVS. We also provide detailed per-scene results in Tab. 12.

Results on ADT dataset. We show more inverse rendering results on Figs. 16 to 19. The metrics on albedo and normal are shown in Tab. 5. We achieve excellent inverse rendering and relighting results thanks to our SVG-IR framework in ADT dataset. We also provide detailed per-scene results in Tab. 13.

Results on DTU dataset. In Fig. 20, we demonstrate the inverse rendering results of our method on the real-world DTU dataset [12]. Utilizing our SVG-IR framework, we recover the material properties and achieve high-quality relighting. Besides, our approach produces natural indirect lighting, thanks to our physically-based illumination model.

Results on NeILF++ dataset. To verify its robustness in relighting, we further evaluate our method on the real-world dataset NeILF++ [33], scomparing it against GS-IR [21] and Relightable 3DGS [9], as shown in Fig. 21. The results demonstrate the robustness of our SVG-IR framework, achieving high-quality relighting even on the relatively sparse-view real-world dataset.

Table 6. Comparison of NVS quality between our method and others on NeRF Synthetic dataset. Red numbers represent the best performance, and orange denotes the second best.

Method	Relightable	PSNR↑	SSIM↑	LPIPS↓
NeRF [24]	✗	31.012	0.947	0.056
Plenoxels [7]	✗	31.714	0.958	0.053
TensoRF [4]	✗	33.140	0.963	0.042
3DGS [19]	✗	33.883	0.971	0.031
GaussianSurfels [6]	✗	33.053	0.961	0.036
TensoIR [16]	✓	29.537	0.943	0.067
GaussianShader [13]	✓	33.367	0.960	0.042
RelightGS [9]	✓	28.238	0.938	0.056
GS-IR [22]	✓	30.133	0.937	0.059
Ours	✓	30.338	0.946	0.051

Results on Mip-NeRF360 dataset. Fig. 22 showcases the inverse rendering and relighting results of our method on MipNeRF360 datasets [1]. Our SVG-IR framework achieves high-quality reconstruction with the Spatially-varying Gaussian and physically-based illumination, demonstrating robust performance on scene-level real-world datasets.

Results on NeRF Synthetic dataset. We evaluate the NVS performance of our method on the NeRF Synthetic dataset [24] and compare it with both relightable and non-relightable approaches in Tab. 6. Among relightable methods, our approach achieves near SOTA NVS quality. GaussianShader builds upon the radiance SH representation in 3DGS by incorporating BRDF, rather than replacing radiance SH with BRDF as other methods do. This enables it to achieve the NVS quality comparable to the original 3DGS at the expense of relighting performance. In contrast, our Spatially-varying Gaussian representation and physically-based illumination enable competitive NVS quality while preserving relighting fidelity.

Results of normal and albedo in ablation study. The impact of Spatially-varying Gaussian (SVG), visibility (Vis.), and indirect ill. (Ind.) on the normal and albedo quality is in Tab. 7.

Table 7. Nor./albedo quality in ablation study (TensoIR dataset).

Component			Albedo			Normal
SVG	Vis.	Ind.	PSNR↑	SSIM↑	LPIPS↓	MAE ↓
✗	✗	✗	26.84	0.926	0.099	4.77
✓	✗	✗	27.94	0.931	0.080	4.53
✓	✓	✗	29.26	0.945	**0.073**	4.39
✓	✓	✓	**30.34**	**0.951**	0.074	**4.36**

Results of R3DG combined with 2D Gaussian. We combine R3DG with 2DGS, similar to our method, and compare it to R3DG, and ours in Tab. 8. While R3DG(2D)

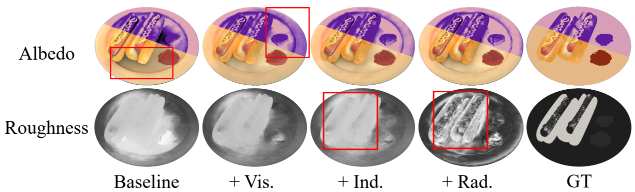


Figure 11. Ablation of indirect illumination components and radiance consistency loss. “Vis.” means the visibility, “Ind.” means the indirect illumination and “Rad.” means the radiance consistency loss. The right part of albedo maps are processed with higher contrast for better observation. The GT roughness is from the “hotdog” blender scene in NeRF Synthetic dataset [24] rather than the TensoIR dataset. [16]

shows an improvement in relighting quality compared to R3DG, it is 1.2dB lower than ours due to its limited representation capacity and lack of physical constraints for indirect illumination.

Table 8. R3DG vs. R3DG(2D) vs. ours (TensoIR dataset).

Method	Gaussian Count	Relight PSNR↑	NVS PSNR ↑	Albedo PSNR ↑	Normal MAE ↓
R3DG	$\sim$ 20k	27.60	33.35	28.54	5.06
R3DG(2D)	$\sim$ 14k	28.96	32.82	29.25	4.72
Ours	$\sim$ 14k	**31.10**	**36.71**	**30.34**	**4.36**

10. More ablation study

Loss. We perform ablation experiments on the loss functions to analyze their impact. The loss terms outlined in Sec. 8 are divided into three categories: (1) the normal loss $\mathcal{L}_N$, which includes $\mathcal{L}_n$; (2) the smoothness loss $\mathcal{L}_s$, comprising $\mathcal{L}_{s,a}$ and $\mathcal{L}_{s,r}$; and (3) our proposed radiance consistency loss $\mathcal{L}_{rc}$. The metrics are shown in Tab. 9. Normals require constraints to prevent overfitting due to their inherent ambiguity in appearance representation. Smooth loss terms on material parameters lead to cleaner rendering results than those without such regularization, which is beneficial for relighting. The radiance consistency loss leverages the pre-trained radiance field to provide supervision from additional viewpoints, improving the quality of both relighting and NVS.

Ray sample counts and cost. We conduct additional experiments on the counts of the sampled directions for per Gaussian. We evaluate the quality, memory cost and rendering time on “armadillo” from TensoIR Synthetic as shown in Tab 10. Under the observation that the quality reaches a plateau at $K = 64$, while maintaining real-time rendering speed and acceptable memory usage, we finally select $K = 64$ for the balance of the quality and the cost.

Table 9. Ablation study of the loss terms. Numbers in red represent the best performance, while orange numbers denote the second best.

Component			Relighting			NVS		
$\mathcal{L}_N$	$\mathcal{L}_s$	$\mathcal{L}_{rc}$	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	✓	✓	30.002	0.939	0.062	36.379	0.972	0.034
✓	✗	✓	30.753	0.941	0.054	36.722	0.975	0.035
✓	✓	✗	30.662	0.943	0.061	36.444	0.974	0.035
✓	✓	✓	31.087	0.946	0.055	36.709	0.975	0.033

Table 10. Ablation on the sample counts. We present the relighting quality and corresponding cost at different sample counts K. Numbers in red represent the best performance, while orange numbers denote the second best. In practice, we set $K = 64$.

Sample count	Relighting			Cost	
	PSNR↑	SSIM↑	LPIPS↓	Memory↓	Rendering Time↓
$K = 16$	34.114	0.9587	0.05627	11.1GB	10ms
$K = 32$	34.842	0..96112	0.05542	12.1GB	11ms
$K = 64$	35.010	0..96289	0.05401	14.3GB	13ms
$K = 128$	35.009	0.96311	0.05392	20.4GB	21ms

Indirect illumination. Fig. 11 presents the ablation results for albedo and roughness maps. By modeling visibility, we alleviate the issue of shadows being baked into the albedo in baseline methods. Indirect illumination modeling further helps decoupling of material and lighting, preventing discrepancies in albedo caused by differing lighting conditions from the left to right. Moreover, indirect illumination serves as the foundation of our proposed radiance consistency loss $\mathcal{L}_{rad}$, which ensures roughness aligns more closely with the ground truth. This improvement is achieved through the additional viewpoint guidance provided by $\mathcal{L}_{rad}$.

Gaussian vertex count. As shown in Tab. 11, more Gaussian vertices result in higher quality with more storage. We choose M=4 in our experiments as a trade-off.

Table 11. Ablation study on GV count (TensoIR dataset). Red dot means a GV. Lagrange interpolation is used.

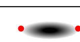
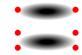
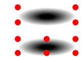
Vertex Count	Vertex Distrib.	Relight PSNR↑	NVS PSNR ↑	Albedo PSNR ↑	Normal MAE ↓
M = 2		30.01	36.37	28.02	4.90
M = 4		31.10	36.71	30.34	4.36
M = 6		**31.17**	**36.77**	**30.74**	**4.33**

Table 12. Per-scene results of normal, albedo and NVS on TensoIR Synthetic dataset. For albedo results, we follow NeRFactor [37] by scaling each RGB channel by a global scalar.

Scene	Method	Normal	Albedo			Novel View Synthesis		
		MAE↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Armadillo	InvRender	1.732	35.573	0.959	0.076	36.681	0.971	0.056
	TensoIR	1.960	34.360	0.989	0.059	39.070	0.986	0.039
	GSshader	2.107	31.092	0.938	0.053	42.445	0.989	0.024
	GS-IR	3.105	38.572	0.986	0.051	38.530	0.972	0.041
	RelightGS	2.224	34.435	0.933	0.067	39.440	0.980	0.042
	Ours	1.974	36.851	0.973	0.047	41.057	0.983	0.031
Ficus	InvRender	4.884	25.335	0.942	0.072	25.498	0.939	0.062
	TensoIR	4.400	27.130	0.964	0.044	29.770	0.973	0.041
	GSshader	4.513	28.239	0.966	0.028	35.256	0.990	0.012
	GS-IR	5.104	30.867	0.948	0.053	33.258	0.960	0.039
	RelightGS	4.991	28.597	0.912	0.057	32.405	0.974	0.028
	Ours	3.408	31.580	0.972	0.032	34.899	0.978	0.025
Hotdog	InvRender	3.708	27.028	0.950	0.094	32.219	0.952	0.070
	TensoIR	4.050	30.370	0.947	0.099	36.780	0.976	0.046
	GSshader	8.315	18.149	0.909	0.127	36.897	0.980	0.029
	GS-IR	4.774	26.745	0.941	0.088	34.843	0.969	0.051
	RelightGS	5.399	25.277	0.939	0.087	30.371	0.943	0.045
	Ours	4.016	27.252	0.952	0.078	36.329	0.977	0.034
Lego	InvRender	9.980	21.435	0.882	0.160	28.277	0.887	0.133
	TensoIR	5.980	25.240	0.900	0.145	35.040	0.970	0.033
	GSshader	8.094	22.625	0.877	0.140	35.403	0.976	0.024
	GS-IR	8.380	24.958	0.889	0.143	33.455	0.954	0.042
	RelightGS	7.643	25.838	0.902	0.135	30.371	0.943	0.045
	Ours	8.032	25.681	0.901	0.139	34.551	0.964	0.041

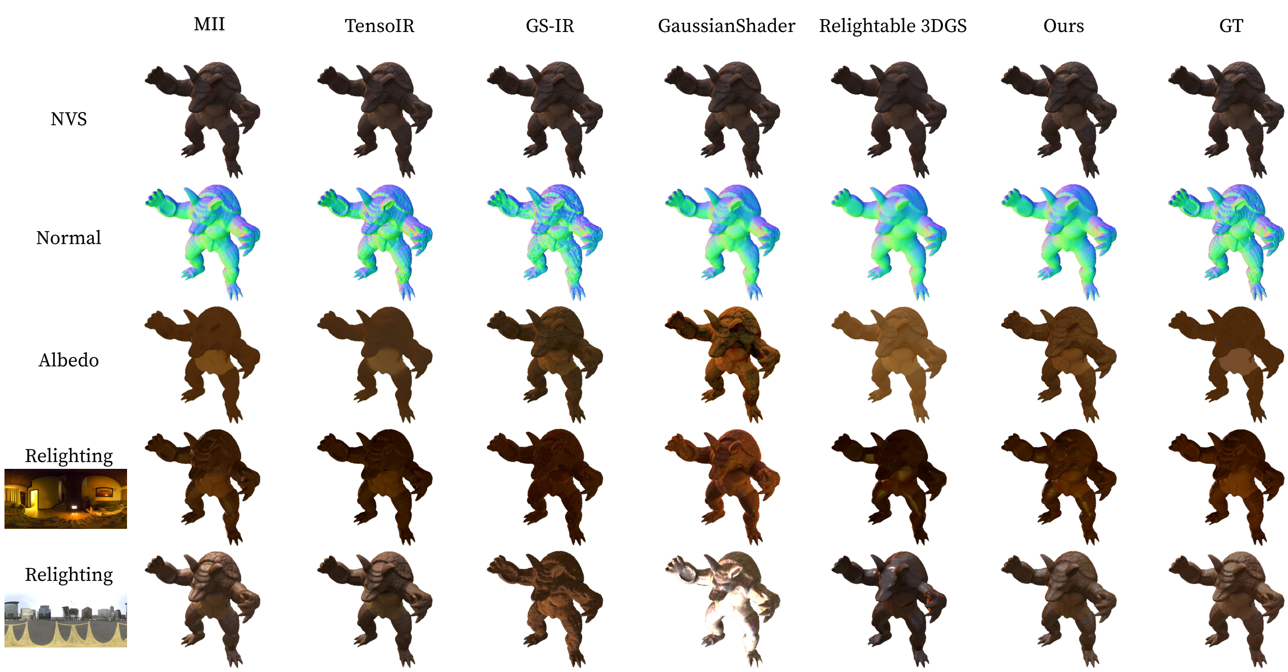


Figure 12. Qualitative comparison of NVS, normal, albedo and relighting on **armadillo** of TensoIR Synthetic dataset.

Table 13. Per-scene results of normal, albedo and NVS on ADT dataset. For albedo results, we follow NeRFactor [37] by scaling each RGB channel by a global scalar.

Scene	Method	Normal	Albedo			Novel View Synthesis		
		MAE ↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Airplane	InvRender	1.688	30.240	0.978	0.037	32.794	0.985	0.022
	TensoIR	1.320	32.400	0.983	0.022	40.370	0.995	0.011
	GSshader	1.207	33.233	0.974	0.024	44.640	0.997	0.004
	GS-IR	1.584	35.449	0.978	0.035	38.755	0.985	0.020
	RelightGS	1.298	35.375	0.973	0.034	37.982	0.991	0.01
	Ours	0.876	36.172	0.987	0.017	42.568	0.994	0.007
Birdhouse	InvRender	3.912	27.770	0.948	0.107	31.237	0.943	0.076
	TensoIR	2.960	29.350	0.961	0.084	39.350	0.986	0.031
	GSshader	3.148	25.984	0.929	0.061	42.167	0.990	0.016
	GS-IR	4.811	28.466	0.944	0.057	37.057	0.977	0.033
	RelightGS	3.083	25.245	0.939	0.055	36.935	0.982	0.027
	Ours	2.911	29.674	0.963	0.04	40.395	0.987	0.019
Gargoyle	InvRender	2.982	29.064	0.924	0.066	29.874	0.945	0.054
	TensoIR	3.310	28.430	0.923	0.067	39.050	0.993	0.010
	GSshader	1.616	30.846	0.972	0.024	42.497	0.996	0.004
	GS-IR	1.711	31.955	0.973	0.022	35.904	0.984	0.013
	RelightGS	2.253	31.424	0.931	0.025	38.910	0.989	0.007
	Ours	1.581	35.09	0.989	0.012	42.079	0.995	0.005
Calculator	InvRender	3.526	29.526	0.956	0.061	29.854	0.954	0.050
	TensoIR	3.160	27.000	0.949	0.051	40.100	0.993	0.016
	GSshader	2.007	31.665	0.965	0.036	43.857	0.996	0.005
	GS-IR	2.553	34.973	0.976	0.032	37.470	0.983	0.022
	RelightGS	2.081	27.216	0.948	0.042	33.915	0.986	0.013
	Ours	1.445	33.582	0.980	0.021	40.881	0.993	0.009

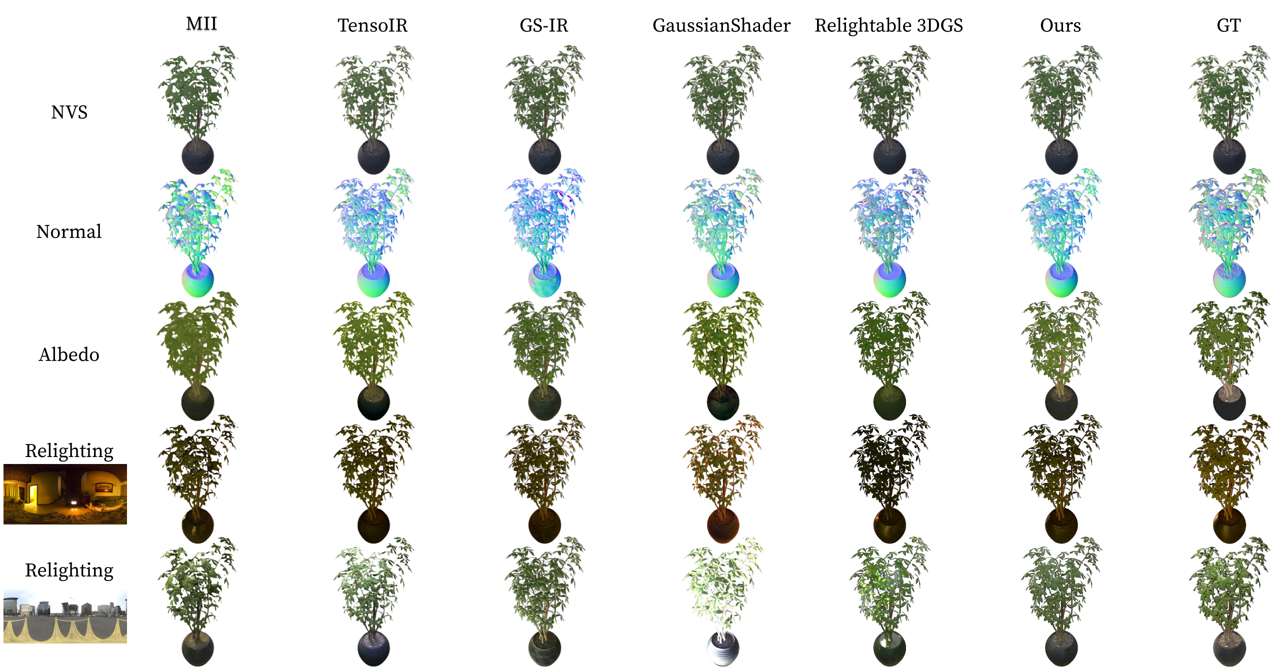


Figure 13. Qualitative comparison of NVS, normal, albedo and relighting on **ficus** of TensoIR Synthetic dataset.

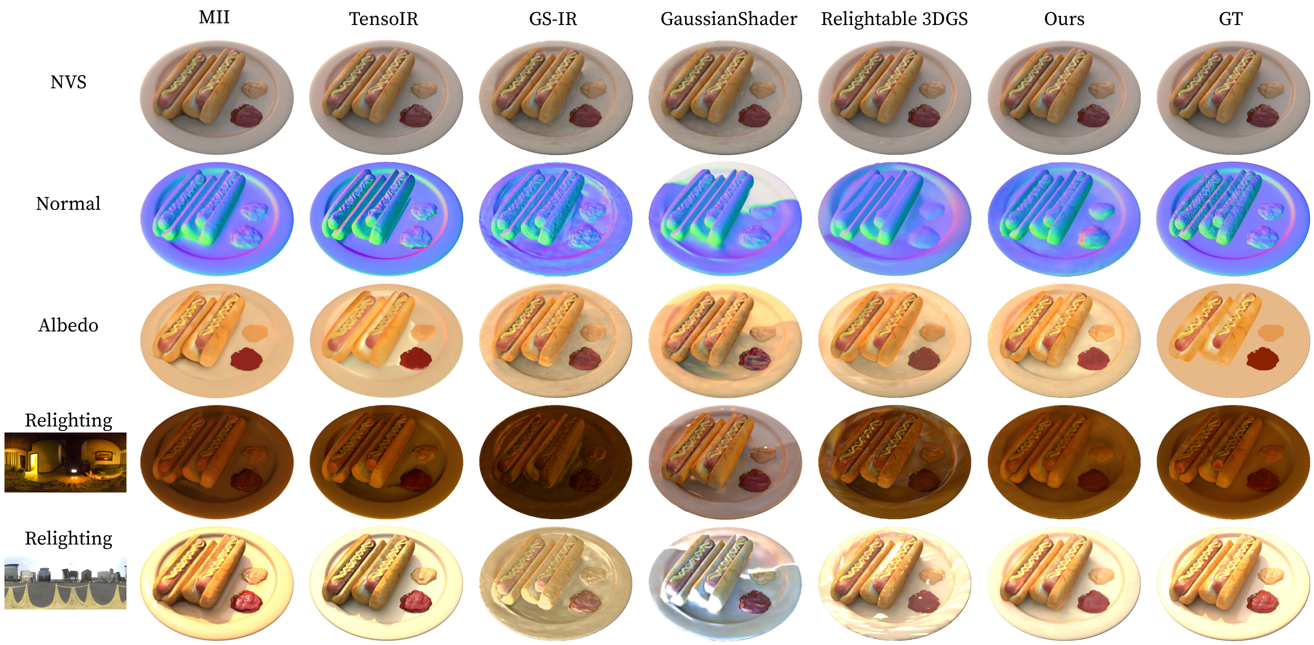


Figure 14. Qualitative comparison of NVS, normal, albedo and relighting on **hotdog** of TensoIR Synthetic datasets.

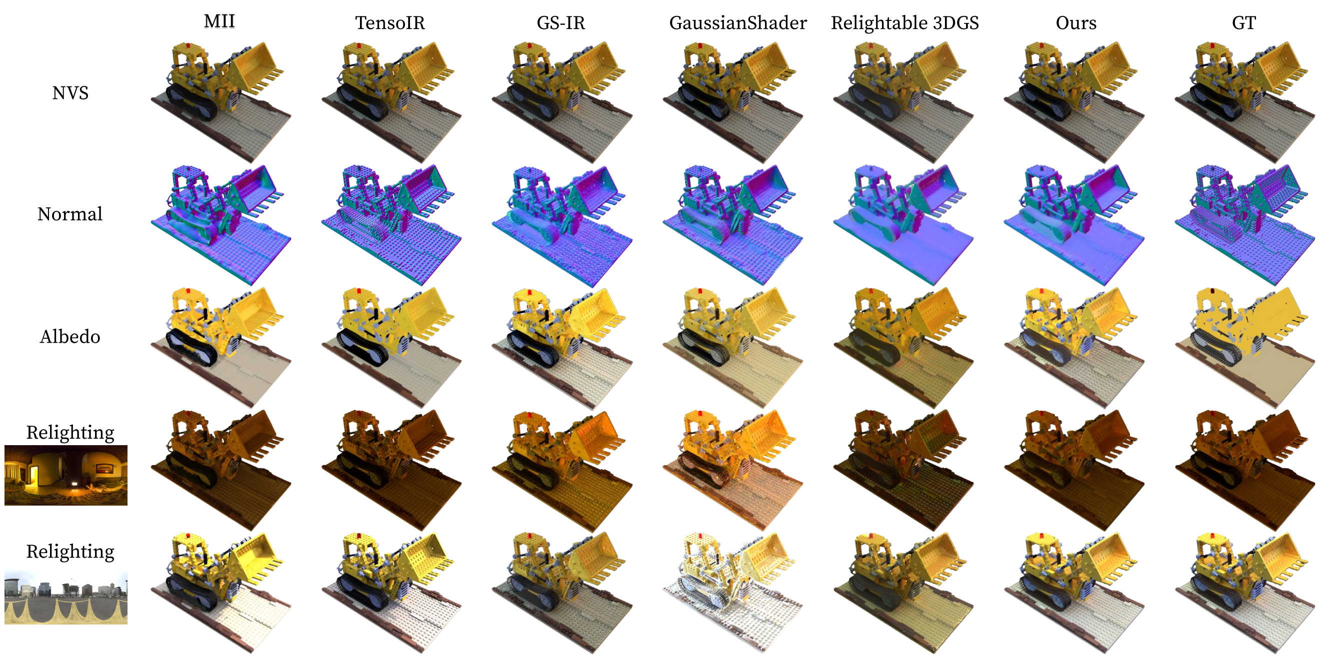


Figure 15. Qualitative comparison of NVS, normal, albedo and relighting on **lego** of TensoIR Synthetic dataset.

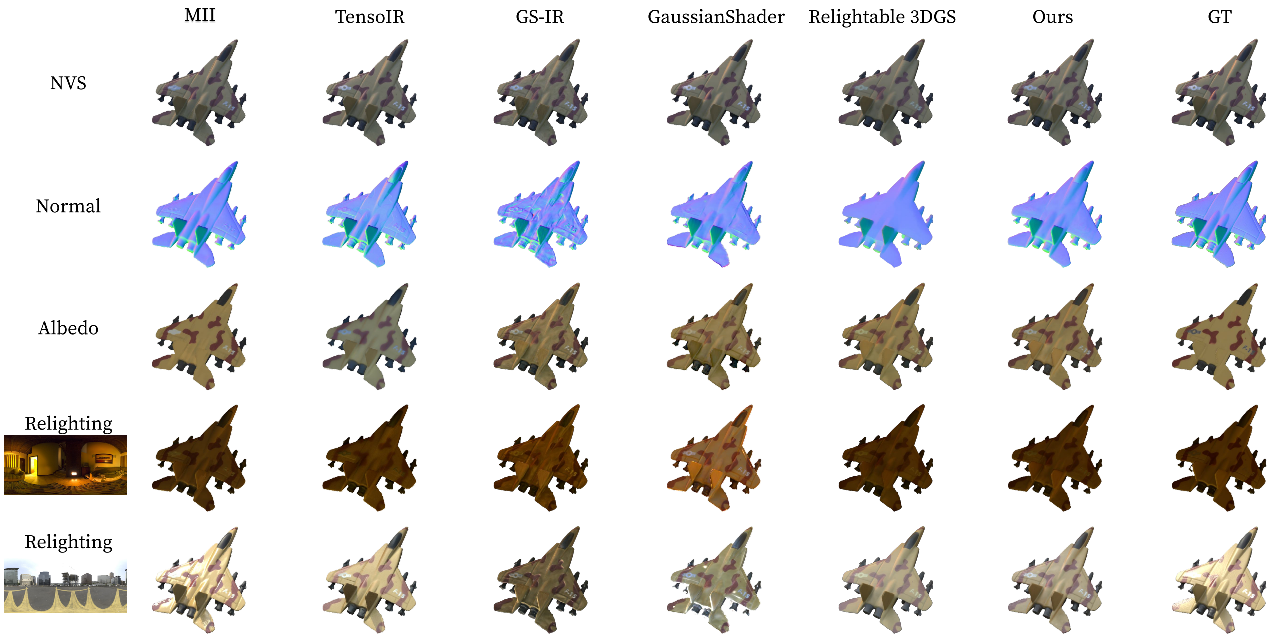


Figure 16. Qualitative comparison of NVS, normal, albedo and relighting on **airsplane** of ADT dataset.

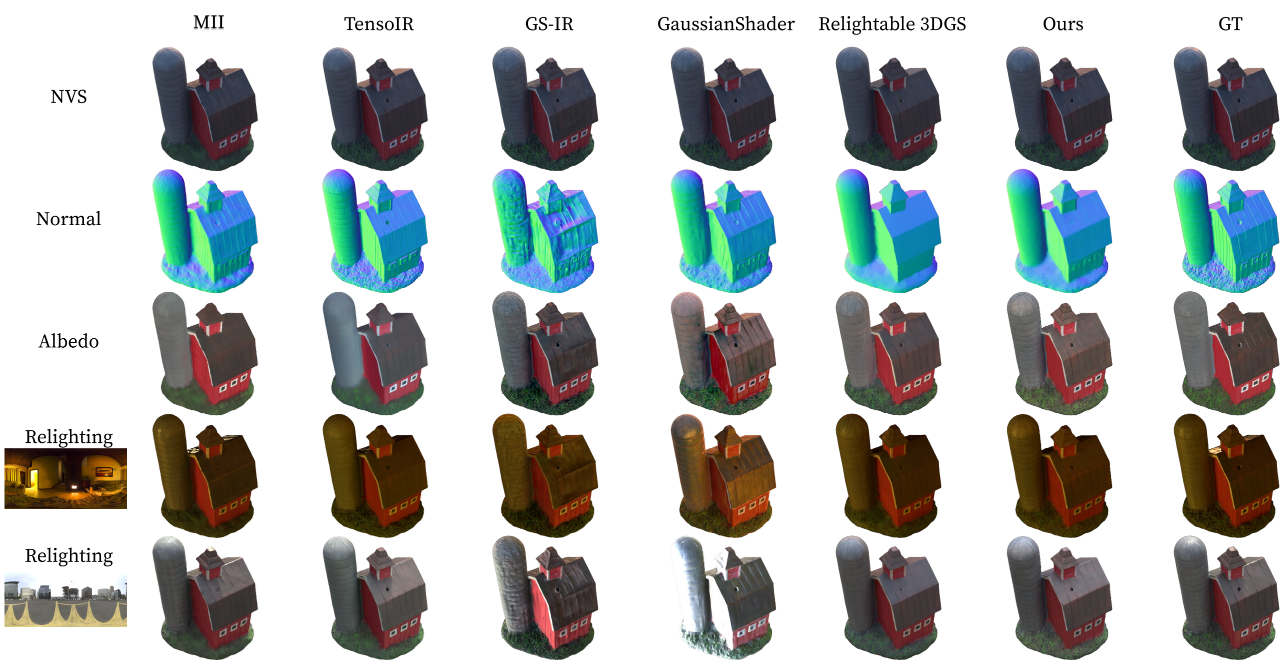


Figure 17. Qualitative comparison of NVS, normal, albedo and relighting on **birdhouse** of ADT dataset.

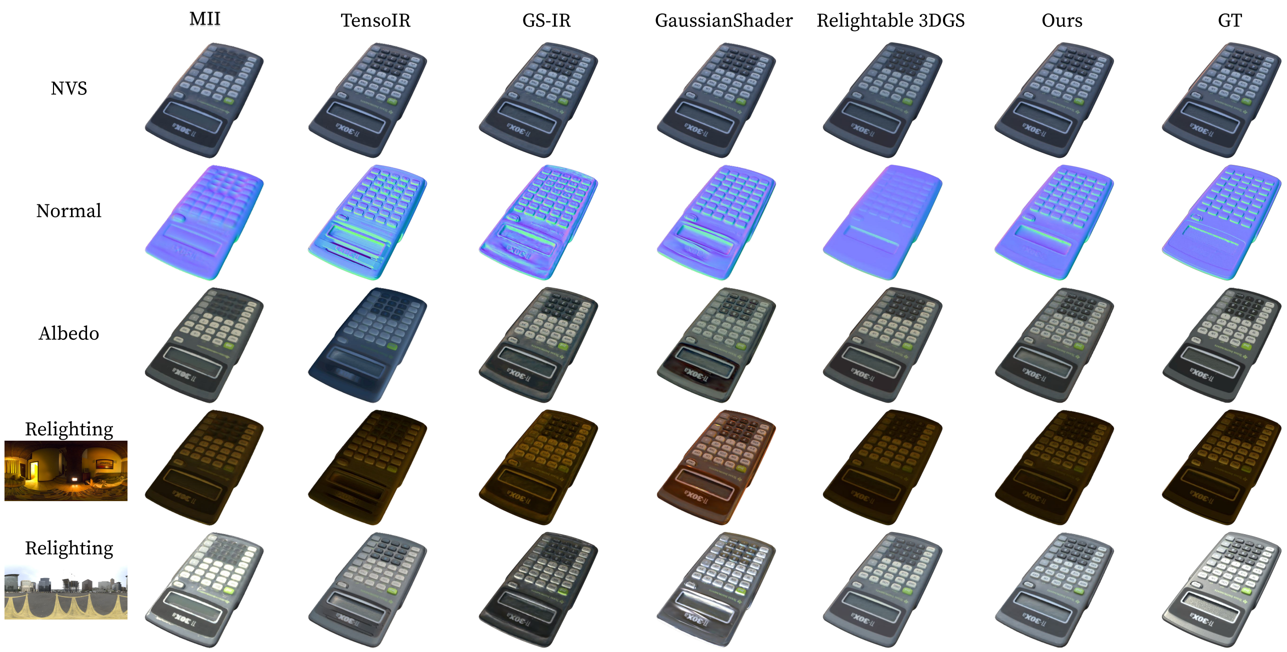


Figure 18. Qualitative comparison of NVS, normal, albedo and relighting on **calculator** of ADT dataset.

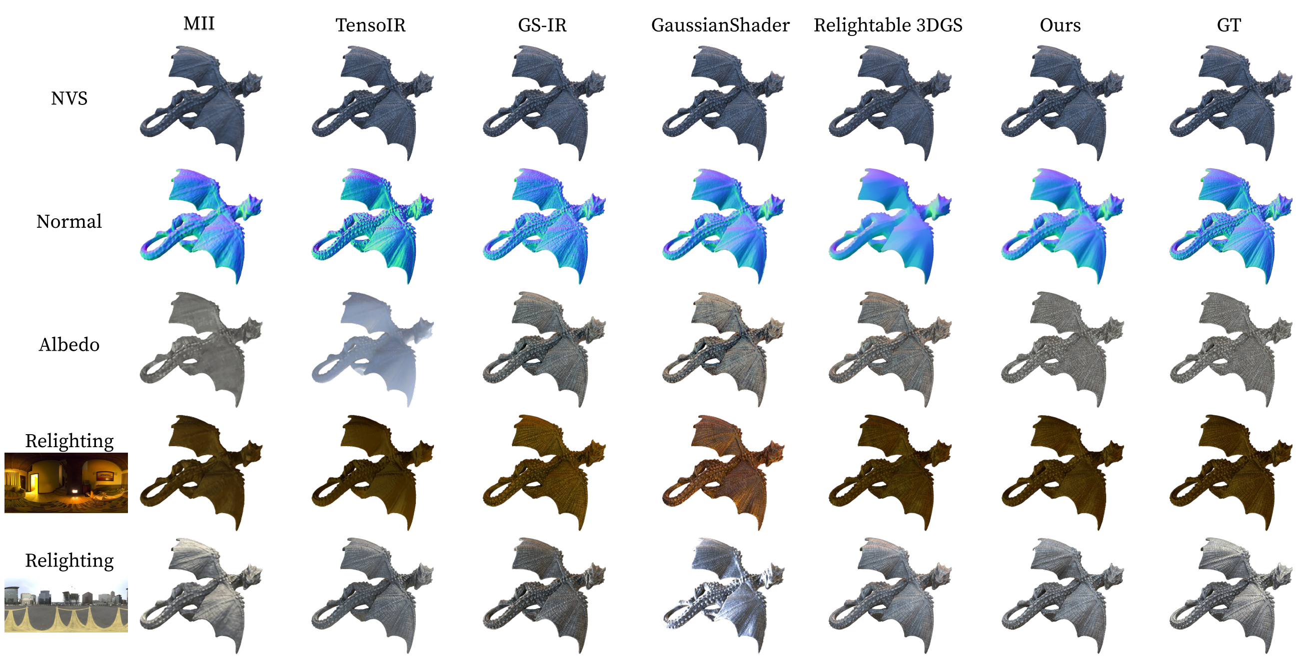


Figure 19. Qualitative comparison of NVS, normal, albedo and relighting on **Gargoyle** of ADT dataset.

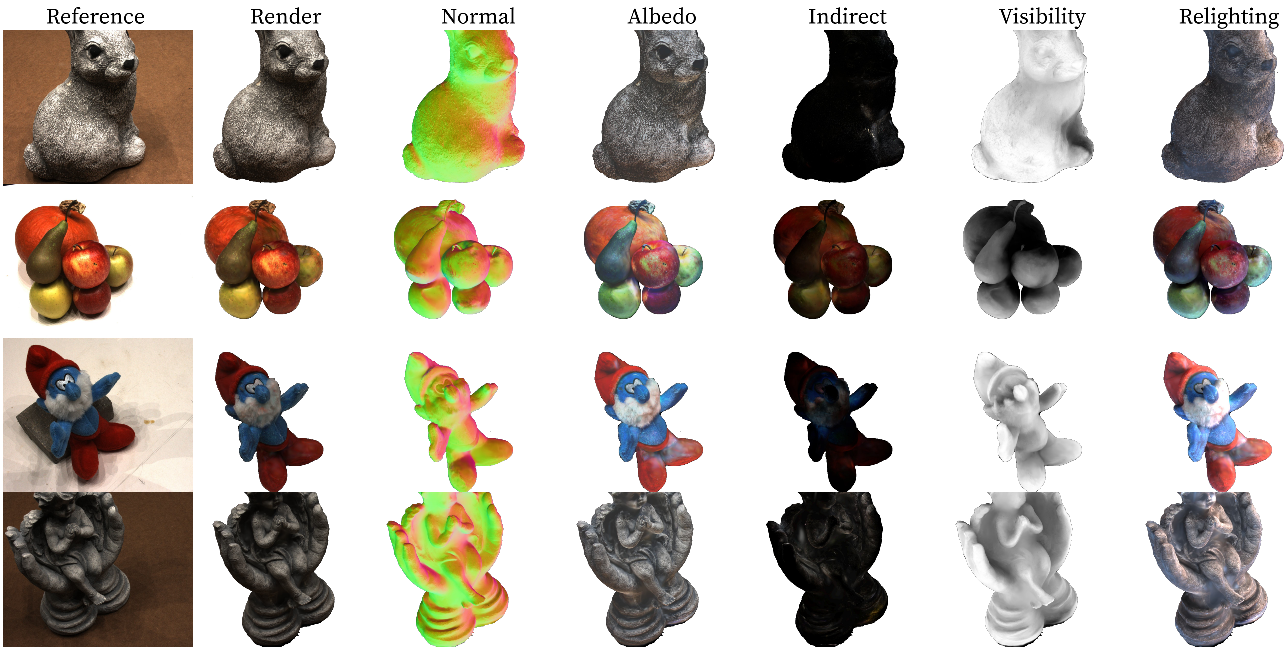


Figure 20. Inverse rendering and relighting results on DTU dataset.

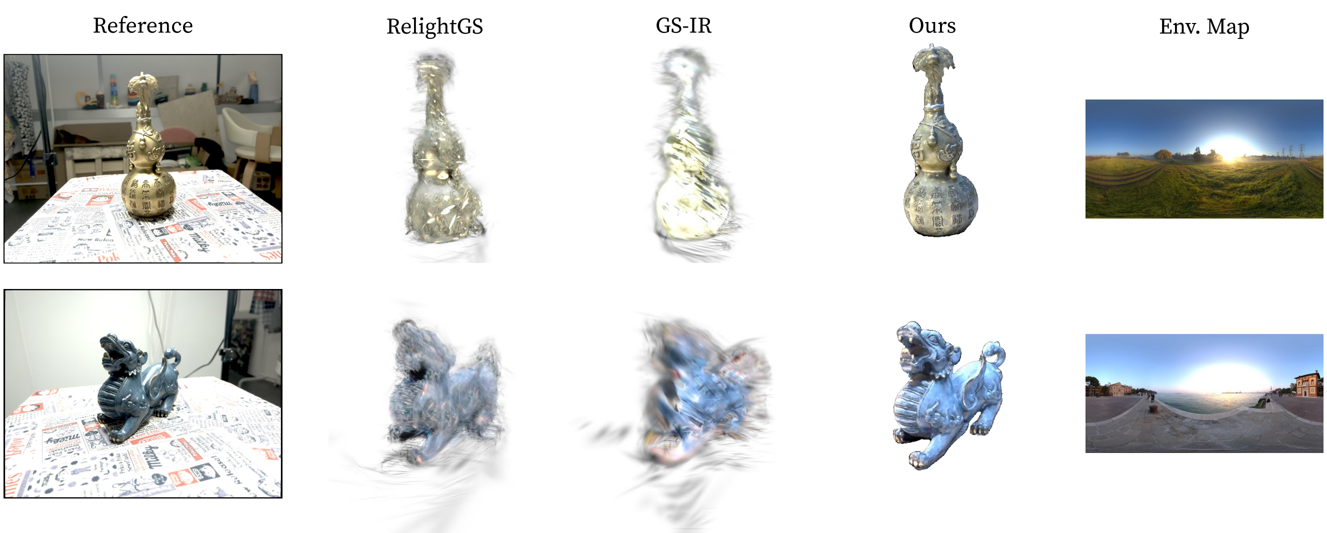


Figure 21. Relighting Qualitative comparison on NeILF++ dataset.

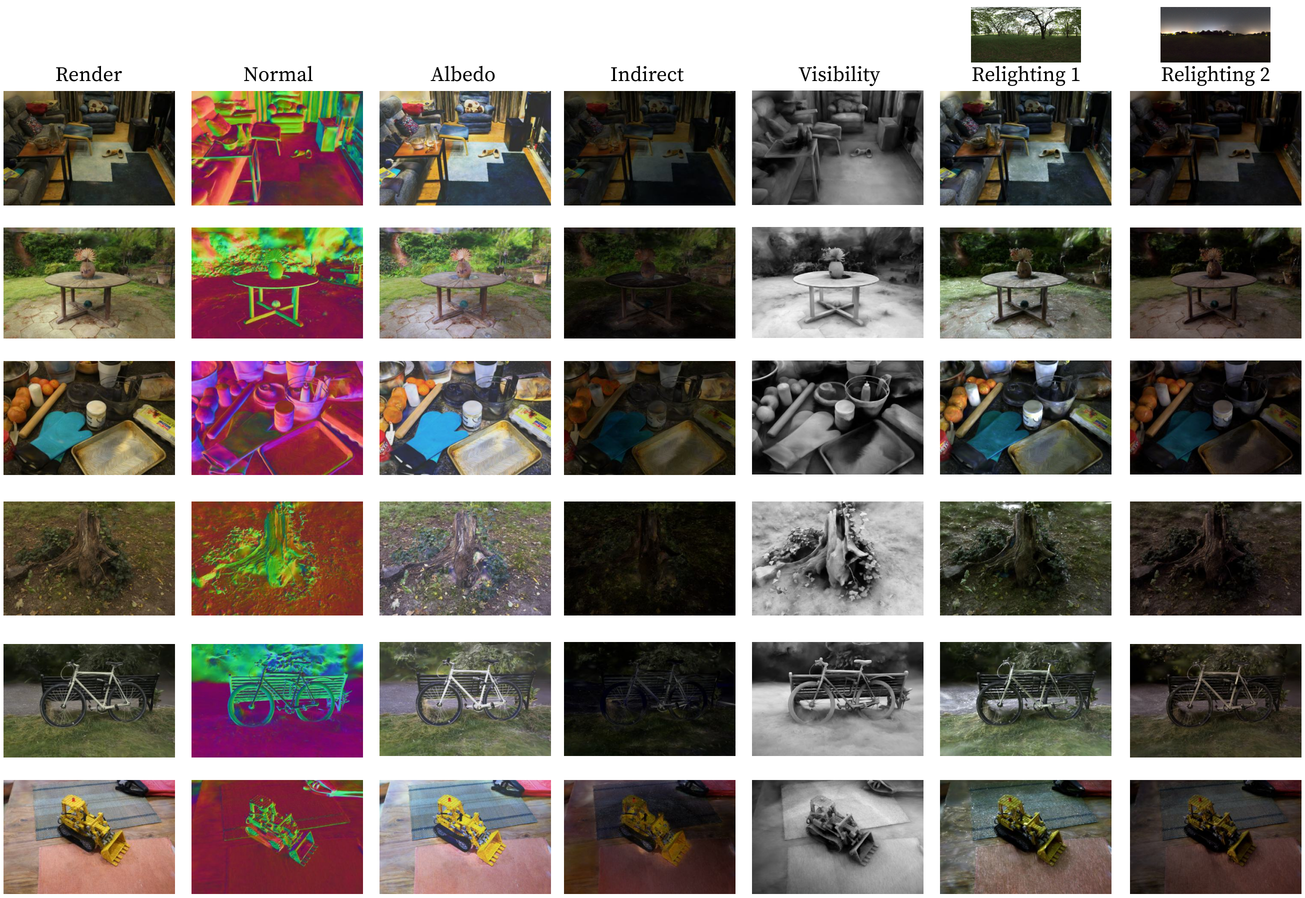


Figure 22. Inverse rendering and relighting results on MipNeRF360 dataset.