GS-IR: 3D Gaussian Splatting for Inverse Rendering - Supplementary Material

Zhihao Liang[1,*], Qi Zhang[2,*], Ying Feng[2], Ying Shan[2], Kui Jia[3,†]
[1]South China University of Technology, [2] Tencent AI Lab,
[3] School of Data Science, The Chinese University of Hong Kong, Shenzhen
eezhihaoliang@mail.scut.edu.cn, nwpuqzhang@gmail.com,
vonyfeng@gmail.com, yingsshan@tencent.com, kuijia@cuhk.edu.cn

1. Implementation Details

We implement GS-IR in PyTorch framework [6] with CUDA extensions, and customized the baking-based method for GS-IR.

Representation. In the vanilla GS [3], each 3D Gaussian utilizes learnable $\mathcal{T} = \{\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{q}\}$ and $\mathcal{A} = \{\alpha, \boldsymbol{f}_c\}$ to describe its geometric properties and volumetric appearance respectively, where $\boldsymbol{p}$ denotes the position vector, $\boldsymbol{s}$ denotes the scaling vector, $\boldsymbol{q}$ denotes the unit quaternion for rotation, α denotes the opacity and $\boldsymbol{f}_c$ denotes spherical harmonics (SH) coefficients for view-dependent color. In GS-IR, we use $\boldsymbol{n}$ to present the normal vector of 3D Gaussian and extend the geometric properties as $\mathcal{T} = \{\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{q}, \boldsymbol{n}\}$. In addition, we introduce $\mathcal{M} = \{\boldsymbol{a}, \rho, m\}$ to describe the material of 3D Gaussian.

Training Details. We use the Adam optimizer [4] for training, and the training process includes the initial stage (*cf*. Sec. 4.1) and decomposition stage (*cf*. Sec. 4.3). In the initial stage, we minimize color reconstruction loss $\mathcal{L}_c$ and normal loss $\mathcal{L}_n$ (*cf*. Eq. (9)) to optimize $\mathcal{T}, \mathcal{A}$ for 30K iterations. In the decomposition stage, we fix $\mathcal{T}, \mathcal{A}$ and minimize the proposed decomposition loss $\mathcal{L}_d$ (*cf*. Eq. (16)) to merely optimize $\mathcal{M}$ for 10K iterations. The total optimization is running on a single V100 GPU.

Loss Definition. In the initial stage, the supervision loss $\mathcal{L}_{\text{init}}$ consists of the $L1$ color reconstruction loss $\mathcal{L}_c$ and our proposed normal loss $\mathcal{L}_n$:

$$\begin{aligned} \mathcal{L}_{\text{init}} &= \mathcal{L}_c + \mathcal{L}_n \\ \mathcal{L}_n &= \mathcal{L}_{n\text{-}p} + \lambda_{n\text{-}TV}\, TV_{\text{normal}} \end{aligned} \tag{1}$$

the smoothing term TV_{normal} in our proposed normal loss $\mathcal{L}_n$ is a total variation (TV) loss conditioned by the predicted normal map $\hat{\boldsymbol{N}}$ and the given reference image $\boldsymbol{I}$:

$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{N}}} &= \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i-1,j}|\right)(\hat{\boldsymbol{N}}_{i,j} - \hat{\boldsymbol{N}}_{i-1,j})^2 + \\ &\quad \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i,j-1}|\right)(\hat{\boldsymbol{N}}_{i,j} - \hat{\boldsymbol{N}}_{i,j-1})^2, \\ TV_{\text{normal}} &= \frac{1}{|\hat{\boldsymbol{N}}|}\sum_{i,j} \triangle_{ij}^{\hat{\boldsymbol{N}}}. \end{aligned} \tag{2}$$

In the decomposition stage, the supervision loss $\mathcal{L}_d$ includes $\mathcal{L}_{\text{shade}}, \mathcal{L}_{\text{material}}, \mathcal{L}_{\text{light}}$:

$$\mathcal{L}_d = \underbrace{\left\| \boldsymbol{I} - \hat{\boldsymbol{I}}^{\text{shade}}(\hat{\boldsymbol{M}}, \hat{\boldsymbol{E}}, \mathcal{V}^{\text{illu}}) \right\|}_{\mathcal{L}_{\text{shade}}} + \underbrace{\lambda_{\boldsymbol{M}}\ TV_{\text{mat}}}_{\mathcal{L}_{\text{material}}} + \underbrace{\lambda_{\boldsymbol{E}}\ TV_{\text{light}}}_{\mathcal{L}_{\text{light}}}, \tag{3}$$

the smoothing term TV_{mat} in Eq. (3) is a TV loss similar to TV_{normal} in Eq. (2):

$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{M}}} &= \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i-1,j}|\right)(\hat{\boldsymbol{M}}_{i,j} - \hat{\boldsymbol{M}}_{i-1,j})^2 + \\ &\quad \exp\left(-|\boldsymbol{I}_{i,j} - \boldsymbol{I}_{i,j-1}|\right)(\hat{\boldsymbol{M}}_{i,j} - \hat{\boldsymbol{M}}_{i,j-1})^2, \\ TV_{\text{mat}} &= \frac{1}{|\hat{\boldsymbol{M}}|}\sum_{i,j} \triangle_{ij}^{\hat{\boldsymbol{M}}}, \end{aligned} \tag{4}$$

where $\hat{\boldsymbol{M}}$ is the predicted material map. Unlike the above two smoothing terms, TV_{light} is defined as:

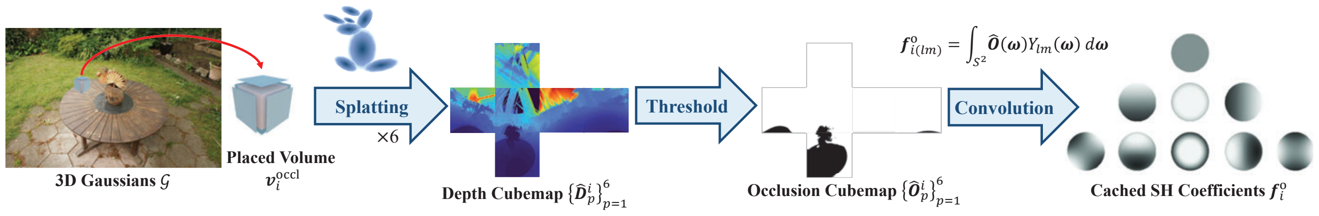
$$\begin{aligned} \triangle_{ij}^{\hat{\boldsymbol{E}}} &= (\hat{\boldsymbol{E}}_{i,j} - \hat{\boldsymbol{E}}_{i-1,j})^2 + (\hat{\boldsymbol{E}}_{i,j} - \hat{\boldsymbol{E}}_{i,j-1})^2, \\ TV_{\text{light}} &= \frac{1}{|\hat{\boldsymbol{E}}|}\sum_{i,j} \triangle_{ij}^{\hat{\boldsymbol{E}}}. \end{aligned} \tag{5}$$

During training, we set $\lambda_{n\text{-}TV}, \lambda_{\boldsymbol{M}}, \lambda_{\boldsymbol{E}}$ to $5.0, 1.0, 0.01$. And we study the efficacy of these smoothing terms in Sec. 5.

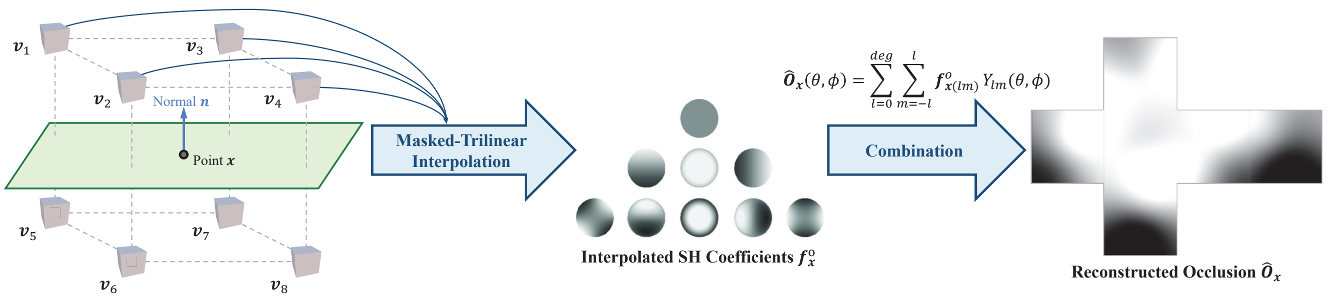
2. Occlusion Caching and Recovery

In the baking stage (*cf*. Sec. 4.2), we introduce SH architectures and cache occlusion into occlusion volumes $\mathcal{V}^{\text{occl}}$ as illustrated in Fig. 1a. For each volume $\boldsymbol{v}_i^{\text{occl}} \subset \mathcal{V}^{\text{occl}}$, we set six cameras with FoV of 90° and non-overlapping each other, and perform six render passes to obtain the depth cubemap $\{\hat{\boldsymbol{D}}_p^i\}_{p=1}^6$. Then we convert $\{\hat{\boldsymbol{D}}_p^i\}_{p=1}^6$ into the occlusion cubemap $\{\hat{\boldsymbol{O}}_p^i\}_{p=1}^6$ and store the principal components of occlusion into SH coefficients $\boldsymbol{f}_{\boldsymbol{x}}^o$.

In the decomposition stage, we recover the ambient occlusion (AO) for each surface point $\boldsymbol{x}$ from occlusion volumes $\mathcal{V}^{\text{occl}}$. The first step is to get the coefficients $\boldsymbol{f}_{\boldsymbol{x}}^o$ of the point $\boldsymbol{x}$. Considering that AO of the point $\boldsymbol{x}$ only calculates the occlusion integral of the upper hemisphere Ω of the normal $\boldsymbol{n}$, we thus conduct masked-trilinear interpolation to get the correct coefficients. As illustrated in Fig. 1b,


3D Gaussians $\mathcal{G}$
Placed Volume v_i^{occl}
Splatting
×6
Depth Cubemap $\{\hat{D}_p^i\}_{p=1}^6$
Threshold
Occlusion Cubemap $\{\hat{O}_p^i\}_{p=1}^6$
$f_{i(lm)}^{o} = \int_{S^2} \hat{O}(\omega) Y_{lm}(\omega)\, d\omega$
Convolution
Cached SH Coefficients f_i^o

(a) Occlusion caching from the pretrained 3D Gaussians $\mathcal{G}$ in the baking stage.


v_1
v_3
v_2
Normal n
v_4
Point x
v_5
v_7
v_6
v_8
Masked-Trilinear Interpolation
Interpolated SH Coefficients f_x^o
$\hat{O}_x(\theta,\phi) = \sum_{l=0}^{deg} \sum_{m=-l}^{l} f_{x(lm)}^{o} Y_{lm}(\theta,\phi)$
Combination
Reconstructed Occlusion $\hat{O}_x$

(b) Occlusion recovery from occlusion volumes $\mathcal{V}^{\text{occl}}$ in the decomposition stage.

Figure 1. Occlusion caching and recovery in GS-IR.

Scene	Method	Normal	Novel View Synthesis			Albedo			Relight		
		MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Lego	NeRFactor	9.767	26.076	0.881	0.151	25.444	0.937	0.112	23.246	0.865	0.156
	InvRender	9.980	24.391	0.883	0.151	21.435	0.882	0.160	20.117	0.832	0.171
	NVDiffrec	12.486	30.056	0.945	0.059	21.353	0.849	0.166	20.088	0.844	0.114
	TensoIR	5.980	34.700	0.968	0.037	25.240	0.900	0.145	28.581	0.944	0.081
	Ours	8.078	34.379	0.968	0.036	24.958	0.889	0.143	23.256	0.842	0.117
Hotdog	NeRFactor	5.579	24.498	0.940	0.141	24.654	0.950	0.142	22.713	0.914	0.159
	InvRender	3.708	31.832	0.952	0.089	27.028	0.950	0.094	27.630	0.928	0.089
	NVDiffrec	5.068	34.903	0.972	0.054	26.057	0.920	0.116	19.075	0.885	0.118
	TensoIR	4.050	36.820	0.976	0.045	30.370	0.947	0.093	27.927	0.933	0.115
	Ours	4.771	34.116	0.972	0.049	26.745	0.941	0.088	21.572	0.888	0.140
Armadillo	NeRFactor	3.467	26.479	0.947	0.095	28.001	0.946	0.096	26.887	0.944	0.102
	InvRender	1.723	31.116	0.968	0.057	35.573	0.959	0.076	27.814	0.949	0.069
	NVDiffrec	2.190	33.664	0.983	0.031	38.844	0.969	0.076	23.099	0.921	0.063
	TensoIR	1.950	39.050	0.986	0.039	34.360	0.989	0.059	34.504	0.975	0.045
	Ours	2.176	39.287	0.980	0.039	38.572	0.986	0.051	27.737	0.918	0.091
Ficus	NeRFactor	6.442	21.664	0.919	0.095	22.402	0.928	0.085	20.684	0.907	0.107
	InvRender	4.884	22.131	0.934	0.057	25.335	0.942	0.072	20.330	0.895	0.073
	NVDiffrec	4.567	22.131	0.946	0.064	30.443	0.894	0.101	17.260	0.865	0.073
	TensoIR	4.420	29.780	0.973	0.041	27.130	0.964	0.044	24.296	0.947	0.068
	Ours	4.762	33.551	0.976	0.031	30.867	0.948	0.053	24.932	0.893	0.081

Table 1. Per-scene results on TensoIR Synthetic dataset. For albedo reconstruction results, we follow NeRFactor [8] and scale each RGB channel by a global scalar.

for the given point $\boldsymbol{x}$ with normal $\boldsymbol{n}$, we firstly find the eight nearest volumes $\{\boldsymbol{v}_k\}_{k=1}^{8}$. In this case, each volume has position vector $\boldsymbol{p}_k$ and SH coefficients $\boldsymbol{f}_k^o$. Given the trilinear interpolation weights $\{w_k\}_{k=1}^{8}$ [1] defined in vanilla trilinear interpolation, we get the coefficients $\boldsymbol{f}_{\boldsymbol{x}}^{\text{o}}$:

$$\begin{aligned} \tilde{w}_k &= \begin{cases} 0, & (\boldsymbol{p}_k - \boldsymbol{x}) \cdot \boldsymbol{n} \le 0 \\ w_k, & (\boldsymbol{p}_k - \boldsymbol{x}) \cdot \boldsymbol{n} > 0 \end{cases}, \\ \hat{w}_k &= \frac{\tilde{w}_k}{\sum_{k=1}^{8} \tilde{w}_k}, \\ \boldsymbol{f}_{\boldsymbol{x}(lm)}^{\text{o}} &= \sum_{k=1}^{8} \hat{w}_k \boldsymbol{f}_{k(lm)}^{\text{o}}. \end{aligned} \tag{6}$$

[1]The weights in trilinear interpolation satisfy $\sum_{k=1}^{8} w_k = 1$

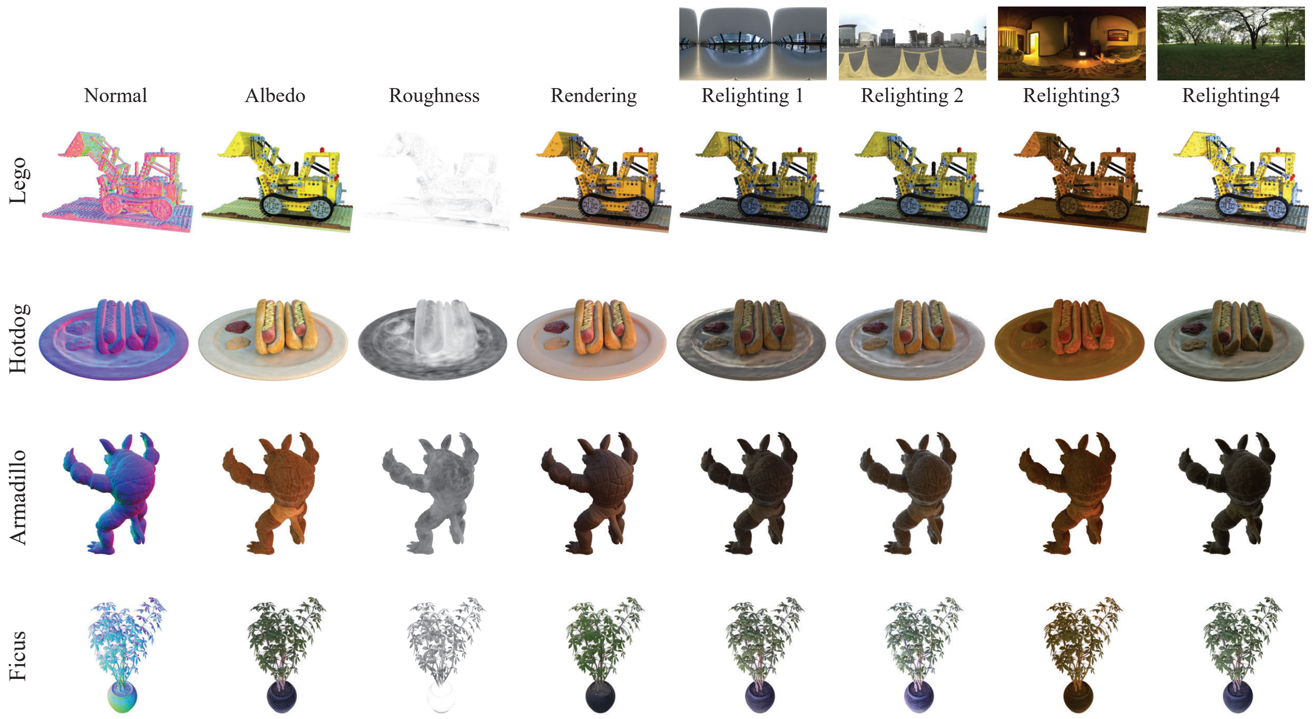


Figure 2. Visualization of our inverse rendering and relighting results on TensoIR Synthetic dataset.

After performing masked-trilinear interpolation, the occlusion $\hat{\boldsymbol{O}}_{\boldsymbol{x}}$ is written as:

$$\hat{\boldsymbol{O}}_{\boldsymbol{x}}(\theta, \phi) = \sum_{l=0}^{deg} \sum_{m=-l}^{l} \boldsymbol{f}^{o}_{\boldsymbol{x}(lm)} Y_{lm}(\theta, \phi). \tag{7}$$

For indirect illumination $I_{\mathrm{d}}^{\mathrm{indir}}$ in Eq. (13), we recover it from the volumes $\mathcal{V}^{\mathrm{illu}}$ via vanilla trilinear interpolation.

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	22.64	20.31	24.32	24.34	22.20	28.87	26.38	27.80	29.15
Plenoxels	21.91	20.10	23.49	20.66	22.25	27.59	23.62	23.42	24.67
INGP-Base	22.19	20.35	24.60	23.63	22.36	29.27	26.44	28.55	30.34
INGP-Big	22.17	20.65	25.07	23.47	22.37	29.69	26.69	29.48	30.69
Mip-NeRF 360	24.40	21.64	26.94	26.36	22.81	29.69	26.69	29.48	30.69
3DGS	25.25	21.52	27.41	26.55	22.49	30.63	28.70	30.32	31.98
Ours	23.80	20.57	25.72	25.37	21.79	28.79	26.22	27.99	28.18

Table 2. PSNR scores for Mip-NeRF360 scenes.

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	0.526	0.453	0.635	0.594	0.530	0.530	0.802	0.816	0.876
Plenoxels	0.496	0.431	0.606	0.523	0.509	0.842	0.759	0.648	0.814
INGP-Base	0.491	0.450	0.649	0.574	0.518	0.855	0.798	0.818	0.890
INGP-Big	0.512	0.486	0.701	0.594	0.542	0.871	0.817	0.858	0.906
Mip-NeRF 360	0.693	0.583	0.816	0.746	0.632	0.913	0.895	0.920	0.939
3DGS	0.771	0.605	0.868	0.775	0.638	0.914	0.905	0.922	0.938
Ours	0.706	0.543	0.804	0.716	0.586	0.867	0.839	0.867	0.883

Table 3. SSIM scores for Mip-NeRF360 scenes.

3. Results on TensoIR Synthetic Dataset

Tab. 1 provides the results on normal estimation, novel view synthesis, albedo reconstruction, and relighting for all four

Method	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
NeRF++	0.455	0.466	0.331	0.416	0.466	0.335	0.351	0.260	0.291
Plenoxels	0.506	0.521	0.386	0.503	0.540	0.419	0.441	0.447	0.398
INGP-Base	0.487	0.481	0.312	0.450	0.489	0.301	0.342	0.254	0.227
INGP-Big	0.446	0.441	0.257	0.421	0.450	0.261	0.306	0.195	0.205
Mip-NeRF 360	0.289	0.345	0.164	0.254	0.338	0.211	0.203	0.126	0.177
3DGS	0.205	0.336	0.103	0.210	0.317	0.220	0.204	0.129	0.205
Ours	0.259	0.371	0.158	0.258	0.372	0.279	0.260	0.188	0.264

Table 4. LPIPS scores for Mip-NeRF360 scenes.

scenes. We also visualize the inverse rendering and relighting results of GS-IR in Fig. 2.

4. Results on Mip-NeRF 360

For Mip-NeRF 360 [1], a dataset captured from the real world, we list the results on novel view synthesis (*i.e.* PSNR, SSIM, and LPIPS) of GS-IR and some NeRF variants [2, 5, 7] in Tabs. 2 to 4. In addition, we provide the normal estimation, novel view synthesis, and relighting results of all seven publicly available scenes in Fig. 3.

5. Ablation on Loss

The loss in GS-IR consists of contrast terms and smoothing terms. For contrast terms, we set the weights of color reconstruction loss $\mathcal{L}_c$, normal penalty loss $\mathcal{L}_{n\text{-}p}$, and shade loss $\mathcal{L}_{\text{shade}}$ to 1, which is intuitive. And the smoothing terms include TV_{normal}, TV_{mat}, and TV_{light}, we evaluate their efficacy by adjusting their weights (*i.e.* $\lambda_{n\text{-}TV}$, $\lambda_{\boldsymbol{E}}$, and $\lambda_{\boldsymbol{M}}$), and the ablation results are shown in Tab. 5.

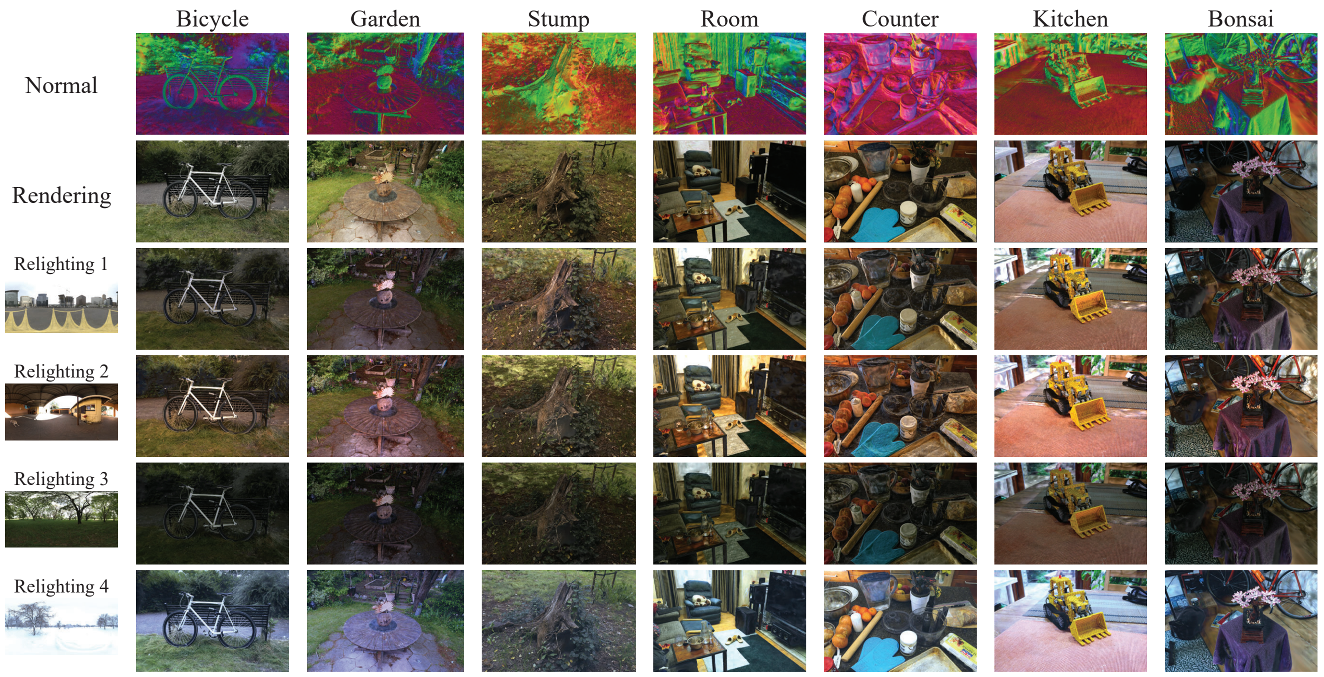


Figure 3. Visualization of our inverse rendering and relighting results on the Mip-NeRF 360 dataset.

			Normal	Novel View Synthesis			Albedo		
$\lambda_{n\text{-}TV}$	$\lambda_{\boldsymbol{E}}$	$\lambda_{\boldsymbol{M}}$	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
			5.030	35.170	0.970	0.042	30.083	0.938	0.090
✓			**4.948**	35.330	**0.974**	0.039	30.216	0.940	0.088
✓	✓		**4.948**	35.230	0.972	0.040	30.236	0.940	0.087
✓		✓	**4.948**	35.314	0.973	**0.038**	30.275	**0.941**	0.085
✓	✓	✓	**4.948**	**35.333**	**0.974**	0.039	**30.286**	**0.941**	**0.084**

Table 5. **Analysis of the impact of different loss terms on the TensoIR dataset.** ✓indicates setting the smoothing term to be valid.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 3

[2] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4): 1–14, 2023. 1

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 3

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[7] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 3

[8] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, 40(6):1–18, 2021. 2