Flash Cache: Reducing Bias in Radiance Cache Based Inverse Rendering Supplement

Benjamin Attal[1], Dor Verbin[2], Ben Mildenhall[2], Peter Hedman[2], Jonathan T. Barron[2]Matthew O'Toole[1], and Pratul P. Srinivasan[2]

[1] Carnegie Mellon University
[2] Google Research

1 Overview

In Section 2, we provide additional implementation details regarding the cost of the fast cache, the cost of the NeRF radiance cache, and the cost and implementation of the physically-based model. We also provide additional details about training, evaluation, and the losses that we use. In Section 3 we prove the unbiasedness of our control variate scheme and our estimator for volume rendering quadrature. In Section 4 we clarify some details regarding our ablations, and provide additional ablations for different values of K for our volume rendering quadrature estimator. Finally, in Section 5, we provide additional result figures and per-scene quantitative metrics. Please see our supplemental website for more results and videos.

2 Additional Implementation Details

2.1 Fast Cache

Rendering a single ray with our fast cache requires:

1. Querying the distance NGP grid once.
2. Querying the feature NGP grid 8 times.
3. Querying the deferred shading MLP once.
4. Querying the environment color once.

We additionally predict the sum of rendering weights for an incoming ray, which is supervised to match the sum of the rendering weights from the NeRF cache. This prevents us from having to query the NeRF cache before blending the fast cache color with the environment color. The full cost of casting a secondary ray (steps 1-4 above) is 0.128 sec for the fast cache (per million rays).

2.2 NeRF Radiance Cache

The cost of rendering a ray from the NeRF radiance cache requires:

1. Querying the first proposal NGP 64 times.

2. Querying the second proposal NGP 64 times.
3. Querying the density NGP 32 times.
4. Querying diffuse color $\mathbf{c}_d$ and specular color $\mathbf{c}_s$ at each of the final 32 sample points, where the specular color is produced using the same architecture as the fast cache.
5. Querying environment color once.

The full cost of casting a secondary ray (steps 1-5 above) is 1.574 sec for the NeRF cache (per million rays).

2.3 Physically-Based Model

For ablations that *do not* use the fast cache, we use 16 samples from the NeRF cache. For models and ablations that use the fast cache, when rendering from the physically-based model, we use 64 samples to estimate incoming $\hat{L}_{\mathrm{o}}^{fast}$ from the fast cache, and 16 samples for both the fast cache and NeRF cache to estimate $\Delta\hat{L}_{\mathrm{o}}$.

2.4 Training Details

Cache training: In the first stage of training, we optimize the Zip-NeRF-based radiance cache with a batch size of 16384 rays for 100,000 iterations using the Charbonnier loss as in [1]. We use a learning rate schedule that warms up from 0 to 0.01 over the first 10,000 iterations, and then decreases linearly to 0.001 for the remaining 90,000 iterations. Training for 100,000 iterations takes 5.5 hours on a single A100.

Joint training: In the second stage of training, we optimize the physically-based model, environment map, fast cache, NeRF-cache, and occlusion-aware importance sampler. We use a batch size of 1024 rays for 40,000 iterations for the TensoIR-synthetic dataset and 100,000 iterations for the Open Illumination dataset, with 64 secondary rays for the fast cache, and 16 rays for the NeRF-cache. We use a learning rate of 1.5625×10^{-4} for the fast cache and occlusion aware importance sampler. For everything else, we use a learning rate of 3.125×10^{-5}. Training for 100,000 iterations takes 6.5 hours on a single A100.

Photometric loss: For the physically-based model, instead of the Charbonnier loss we use a variant of the RawNeRF [7] loss. For a given ray $(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$, predicted color $L_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)$ and ground truth $I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$, we *would like to* minimize:

$$\mathcal{L}_{photometric} = \frac{\left\| I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) - \mathbb{E}\left[\hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\right] \right\|^2}{\operatorname{stopgrad}\left(L_{\mathrm{i}}^{cache}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\right)} \tag{1}$$

where $\mathbb{E}\left[\hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\right]$ is the expected value of the estimator for radiance $\hat{L}_{\mathrm{i}}$. However, we cannot minimize this loss directly, since we do not have access to an analytic expression for the expected value. Instead, we apply the gradient trick [3], which gives correct gradients through this loss in expectation for the physically-based model. More concretely, we minimize:

$$\mathcal{L}_{photometric} = \frac{2\Big(I(\mathbf{o},\boldsymbol{\omega}_{\mathrm{o}}) - \hat{L}_{\mathrm{i}}(\mathbf{o},\boldsymbol{\omega}_{\mathrm{o}};\Phi)\Big)\,\mathrm{stopgrad}\Big(I(\mathbf{o},\boldsymbol{\omega}_{\mathrm{o}}) - \hat{L}_{\mathrm{i}}(\mathbf{o},\boldsymbol{\omega}_{\mathrm{o}};\Phi)\Big)}{\mathrm{stopgrad}(\hat{L}_{\mathrm{i}}^{cache}(\mathbf{o},\boldsymbol{\omega}_{\mathrm{o}};\Phi))} \tag{2}$$

where the stopgrad$(\cdot)$ operator treats its argument as a constant in the compute graph, and the first and second differences in the numerator are estimated using independent secondary samples.

2.5 Evaluation Details

During evaluation we use 64 secondary rays, but render the predicted color image 32 times and average the results in order to reduce Monte Carlo noise. We noticed that TensoIR renderings still contain some noise, and acknowledge that different sample counts and different importance sampling schemes could impact relative PSNR, but note that we outperform TensoIR in terms of albedo metrics, which is not affected by noise, and relighting by a fairly large margin.

2.6 BRDF Model

As we discuss in the main text, we use the Disney-GGX BRDF [2], which is comprised of three terms: albedo $\mathbf{a}(\mathbf{x})$, metalness $m(\mathbf{x})$, and roughness $r(\mathbf{x})$. This BRDF can be written as:

$$f(\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}},\mathbf{x}) = f_{diffuse}(\mathbf{x}) + f_{specular}(\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}},\mathbf{x}) \tag{3}$$

$$f_{diffuse}(\mathbf{x}) = \frac{(1-m(\mathbf{x}))\mathbf{a}(\mathbf{x})}{\pi} \tag{4}$$

$$f_{specular}(\boldsymbol{\omega}_{\mathrm{i}},\boldsymbol{\omega}_{\mathrm{o}},\mathbf{x}) = \frac{DFG}{4(\mathbf{n}\cdot\boldsymbol{\omega}_{\mathrm{i}})(\mathbf{n}\cdot\boldsymbol{\omega}_{\mathrm{o}})} \tag{5}$$

We refer readers to Burley [2] and Liu *et al*. [6] for definitions of (D,F,G) — the normal distribution function (NDF), Fresnel, and geometry terms. We use the Trowbridge-Reitz distribution function [8] for the NDF D.

2.7 Importance Sampling

For both the fast and NeRF caches, we split secondary samples evenly between the diffuse color (due to the diffuse BRDF component $f_{diffuse}$) and the specular color (due to the specular BRDF component $f_{specular}$). For the diffuse color, we leverage multiple importance sampling [8] with half of the samples coming from the occlusion-aware importance sampler and half of the samples coming from cosine-weighted hemisphere sampling. For the specular color, we importance sample according to the distribution function D [8].

2.8 Normal Loss

As discussed in the paper, we emit predicted normals from the density NGP. Similar to Ref-NeRF [9] and TensoIR [4], we constrain our predicted normals to match the negative gradient of the density field with an L2 loss:

$$\mathcal{L}_{normals} = C_{normals} \sum_k w_k \left\| \mathbf{n}_k^{pred} - \mathbf{n}_k^{derived} \right\|^2 \tag{6}$$

where w_k are the render weights for a given ray, and

$$\mathbf{n}_k^{derived} = -\frac{\nabla \sigma(\mathbf{x}_k)}{\|\nabla \sigma(\mathbf{x}_k)\|} \tag{7}$$

The loss weight $C_{normals}$ varies per-dataset. For the Open Illumination dataset, we use $C_{normals} = 1.0$. For the TensoIR-synthetic dataset, we linearly interpolate $C_{normals}$ from 0.0001 to 1.0 from iteration 20,000 to iteration 40,000.

2.9 Cache Consistency Loss

To allow the physically-based model to constrain the appearance of the NeRF-cache, we supervise the diffuse and specular colors from the cache $\mathbf{c}_d^{cache}(\mathbf{x})$ and $\mathbf{c}_s^{cache}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$ to match the diffuse and specular colors from the physically-based model $\mathbf{c}_d^{phys}(\mathbf{x})$ and $\mathbf{c}_s^{phys}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$. We further predict an additional output from the cache $\mathbf{c}_{irradiance}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$, which is supervised to match the irradiance from the physically-based model (computed by setting the BRDF to be perfectly Lambertian with $\mathbf{a}(\mathbf{x}) = \mathbf{1}$). For all of the above, we use the same Raw-NeRF loss as in Equation 2.

2.10 Smoothness Loss

To enforce BRDF smoothness we use a variant of TensoIR's smoothness loss:

$$\mathcal{L}_{BRDF} = C_{BRDF} \sum_k w_k \left| \frac{\beta(\mathbf{x}_k) - \beta(\mathbf{x}_k + \xi)}{\max(\beta(\mathbf{x}_k), \beta(\mathbf{x}_k + \xi))} \right| \lambda(\mathbf{x}, \mathbf{x} + \xi) \tag{8}$$

$$\lambda(\mathbf{x}, \mathbf{x} + \xi) = |\mathbf{a}_{pseudo}(\mathbf{x}_k) - \mathbf{a}_{pseudo}(\mathbf{x}_k + \xi)| \qquad \xi \sim \mathcal{N}(\mathbf{0}, \epsilon \mathrm{I}) \tag{9}$$

Where $\mathbf{a}_{pseudo}(\mathbf{x}_k)$ is the "pseudo-albedo" at point $\mathbf{x}_k$:

$$\mathbf{a}_{pseudo}(\mathbf{x}_k) = \frac{\mathbf{c}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{\mathbf{c}_{irradiance}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})} \tag{10}$$

For the TensoIR dataset, we set $\epsilon = 0.01$ with $C_{BRDF} = 0.05$, and for other datasets we set $\epsilon = 0.005$ with $C_{BRDF} = 0.001$.

3 Proofs

3.1 Control Variates

Here we show that Equation 9 in the main paper is an unbiased estimator of the rendering equation (provided that incoming illumination from the NeRF cache is correct):

$$\begin{aligned}
\mathbb{E}\left[\hat{L}_{\mathrm{o}}\right] &= \mathbb{E}\left[\hat{L}_{\mathrm{o}}^{fast}\right] + \mathbb{E}\left[\Delta\hat{L}_{\mathrm{o}}\right] && \text{(Eq. 9)} \\
&= \int_{\Omega} f(\mathbf{x}(t), \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) \hat{L}_{\mathrm{i}}^{NeRF}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})(\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}})\, d\boldsymbol{\omega}_{\mathrm{i}} && \text{(unbiasedness of Eq. 5)} \\
&\quad + \int_{\Omega} f(\mathbf{x}(t), \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) \left(\hat{L}_{\mathrm{i}}^{NeRF} - \hat{L}_{\mathrm{i}}^{fast}\right)(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})(\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}})\, d\boldsymbol{\omega}_{\mathrm{i}} \\
&= \int_{\Omega} f(\mathbf{x}(t), \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) \hat{L}_{\mathrm{i}}^{NeRF}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})(\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}})\, d\boldsymbol{\omega}_{\mathrm{i}} && \square
\end{aligned}$$

3.2 Volume Rendering

Here we show that Equation 11 is an unbiased estimator for volume rendering quadrature (*e.g.* its expected value is equal to Equation 3):

$$\begin{aligned}
\mathbb{E}\left[\hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})\right] &= \mathbb{E}\left[\frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} L_{\mathrm{o}}(\mathbf{x}(t_{j_k}), \boldsymbol{\omega}_{\mathrm{o}})\right] && \text{(Eq. 11)} \\
&= \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} \mathbb{E}[L_{\mathrm{o}}(\mathbf{x}(t_{j_k}), \boldsymbol{\omega}_{\mathrm{o}})] && (j_k \sim \mathrm{Cat}(w_1, \ldots, w_{\mathrm{N}})) \\
&= \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} \sum_{j=1}^{\mathrm{N}} w_j L_{\mathrm{o}}(\mathbf{x}(t_j), \boldsymbol{\omega}_{\mathrm{o}}) \\
&= \sum_{j=1}^{\mathrm{N}} w_j L_{\mathrm{o}}(\mathbf{x}(t_j), \boldsymbol{\omega}_{\mathrm{o}}) && \square
\end{aligned}$$

where $(w_1, \ldots, w_{\mathrm{N}})$ are render weights for the ray $(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$.

4 Additional Ablations

All ablations are evaluated in the single-light setting of the TensoIR-synthetic dataset.

Table 1: Sample Number Ablations

Method	NVS PSNR↑	Albedo PSNR↑	MAE↓
(a) K = 1 (Ours)	**34.915**	**30.345**	3.355
(b) K = 2	34.913	30.208	**3.354**
(c) K = 4	34.850	30.275	3.356
(d) K = 8	34.778	30.059	3.355

4.1 Secondary Sample Ablations

We report aggregate novel view synthesis PSNR, albedo PSNR, and normal MAE for three additional ablations in Table 1 as we vary the value of K in Equation 11. In practice, we find that $K = 1$ provides the best results.

We expect that $K = 1$ will do worse on complex datasets with "more volumetric" geometry or partial transparencies, although we note that for any K, Equation 11 is still an unbiased estimator for volume rendering quadrature.

5 Additional Results

5.1 Per-Scene Results for TensoIR

We provide per-scene results for both the TensoIR-Synthetic dataset in Table 2. We additionally provide results for the multi-light TensoIR setting.

5.2 Per-Scene Results for Open Illumination

We provide per-scene results for both the Open Illumination dataset in Table 3. We label each scene as *diffuse* or *specular*. The dataset provides both single-light data (with the object illuminated under a single lighting condition) and multi-light data (with the object illuminated under many different lighting conditions). We perform evaluation in the single-light setting for novel view synthesis, and in the multi-light setting for relighting, as no relighting metrics are provided in the original Open Illumination paper for the single-light setting.

5.3 Qualitative Results

Find additional qualitative results in Figures 1, 3, an 4as well as our videos on our supplemental website.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. ICCV (2023)
2. Burley, B., Studios, W.D.A.: Physically-Based Shading at Disney. ACM Trans. Graph. (2012)

Table 2: Per-Scene TensoIR-Synthetic Dataset [4] Results.

	Method	Normal MAE↓	Albedo PSNR↑	Albedo SSIM↑	Albedo LPIPS↓	Novel View Synthesis PSNR↑	Novel View Synthesis SSIM↑	Novel View Synthesis LPIPS↓	Relighting PSNR↑	Relighting SSIM↑	Relighting LPIPS↓
Armadillo	NeRFactor	3.467	28.001	0.946	0.096	26.479	0.947	0.095	26.887	0.944	0.102
	InvRender	1.723	35.573	0.959	0.076	31.116	0.968	0.057	27.814	0.949	0.069
	TensoIR	1.950	34.360	0.989	0.059	39.050	0.986	0.039	34.504	0.975	0.045
	Ours	1.564	34.832	0.960	0.082	39.313	0.977	0.043	34.809	0.959	0.058
	TensoIR multi-light	1.550	34.270	0.989	0.057	38.230	0.984	0.043	34.941	0.977	0.043
	Ours multi-light	1.427	35.921	0.961	0.079	39.097	0.978	0.042	35.937	0.965	0.052
Ficus	NeRFactor	6.442	22.402	0.928	0.085	21.664	0.919	0.095	20.684	0.907	0.107
	InvRender	4.884	25.335	0.942	0.072	22.131	0.934	0.057	20.330	0.895	0.073
	TensoIR	4.420	27.130	0.964	0.044	29.780	0.973	0.041	24.296	0.947	0.068
	Ours	2.709	28.337	0.972	0.048	30.380	0.976	0.036	26.286	0.960	0.052
	TensoIR multi-light	4.060	26.220	0.952	0.054	28.640	0.967	0.050	24.622	0.949	0.068
	Ours multi-light	2.689	28.137	0.971	0.046	30.175	0.976	0.037	26.730	0.964	0.049
Hotdog	NeRFactor	5.579	24.654	0.950	0.142	24.498	0.940	0.141	22.713	0.914	0.159
	InvRender	3.708	27.028	0.950	0.094	31.832	0.952	0.089	27.630	0.928	0.089
	TensoIR	4.050	30.370	0.947	0.093	36.820	0.976	0.045	27.927	0.933	0.115
	Ours	2.882	30.832	0.966	0.073	36.966	0.961	0.095	29.241	0.941	0.104
	TensoIR multi-light	3.220	31.240	0.958	0.080	35.670	0.973	0.048	28.952	0.939	0.110
	Ours multi-light	2.741	31.180	0.968	0.077	36.036	0.958	0.097	29.050	0.942	0.103
Lego	NeRFactor	9.767	25.444	0.937	0.112	26.076	0.881	0.151	23.246	0.865	0.156
	InvRender	9.980	21.435	0.882	0.160	24.391	0.883	0.151	20.117	0.832	0.171
	TensoIR	5.980	25.240	0.900	0.145	34.700	0.968	0.037	27.596	0.922	0.095
	Ours	6.265	27.097	0.922	0.137	32.973	0.945	0.081	28.560	0.917	0.105
	TensoIR multi-light	5.370	25.560	0.905	0.146	34.350	0.967	0.038	27.517	0.922	0.091
	Ours multi-light	5.713	27.735	0.917	0.143	31.942	0.942	0.083	28.286	0.918	0.102

3. Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., Levin, A.: Inverse volume rendering with material dictionaries. ACM Trans. Graph. (2013)
4. Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: TensoIR: Tensorial Inverse Rendering. CVPR (2023)
5. Liu, I., Chen, L., Fu, Z., Wu, L., Jin, H., Li, Z., Wong, C.M.R., Xu, Y., Ramamoorthi, R., Xu, Z., et al.: OpenIllumination: A Multi-Illumination Dataset for Inverse Rendering Evaluation on Real Objects. NeurIPS (2024)
6. Liu, Y., Wang, P., Lin, C., Long, X., Wang, J., Liu, L., Komura, T., Wang, W.: NeRO: Neural Geometry and BRDF Reconstruction of Reflective Objects from Multiview Images. ACM Trans. Graph. (2023)
7. Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T.: NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images. CVPR (2022)
8. Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation. MIT Press (2023)
9. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. CVPR (2022)

Table 3: Per-Scene Open Illumination dataset [5] Results.

	Scene	Method	PSNR ↑ NVS	PSNR ↑ Relit.
Diffuse	*Egg*	TensoIR	**34.88**	**31.99**
		Ours	34.48	30.76
	Stone	TensoIR	29.96	**31.07**
		Ours	**30.52**	30.55
	Pumpkin	TensoIR	**28.20**	**27.16**
		Ours	27.64	26.58
	Hat	TensoIR	**31.96**	**32.38**
		Ours	31.02	30.41
	Sponge	TensoIR	**32.49**	**30.86**
		Ours	32.14	28.87
	Banana	TensoIR	34.77	**32.13**
		Ours	**34.89**	30.90
Specular	*Bird*	TensoIR	**30.21**	30.16
		Ours	29.92	**30.19**
	Box	TensoIR	**26.80**	27.57
		Ours	26.40	**28.93**
	Cup	TensoIR	**22.13**	22.96
		Ours	21.84	**23.35**
	Bucket	TensoIR	29.32	27.13
		Ours	**30.55**	**28.77**

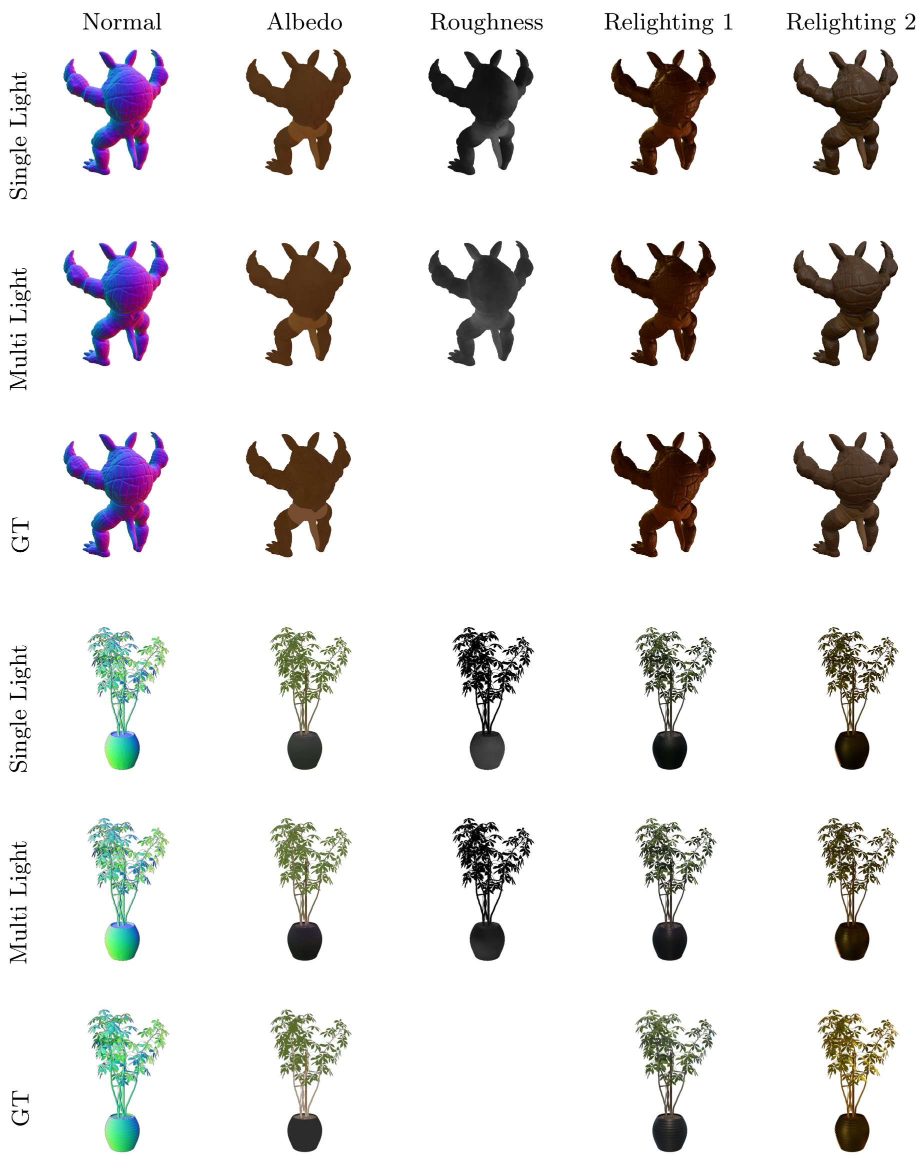

Fig. 1: Additional TensoIR-Synthetic Results [4] Results.

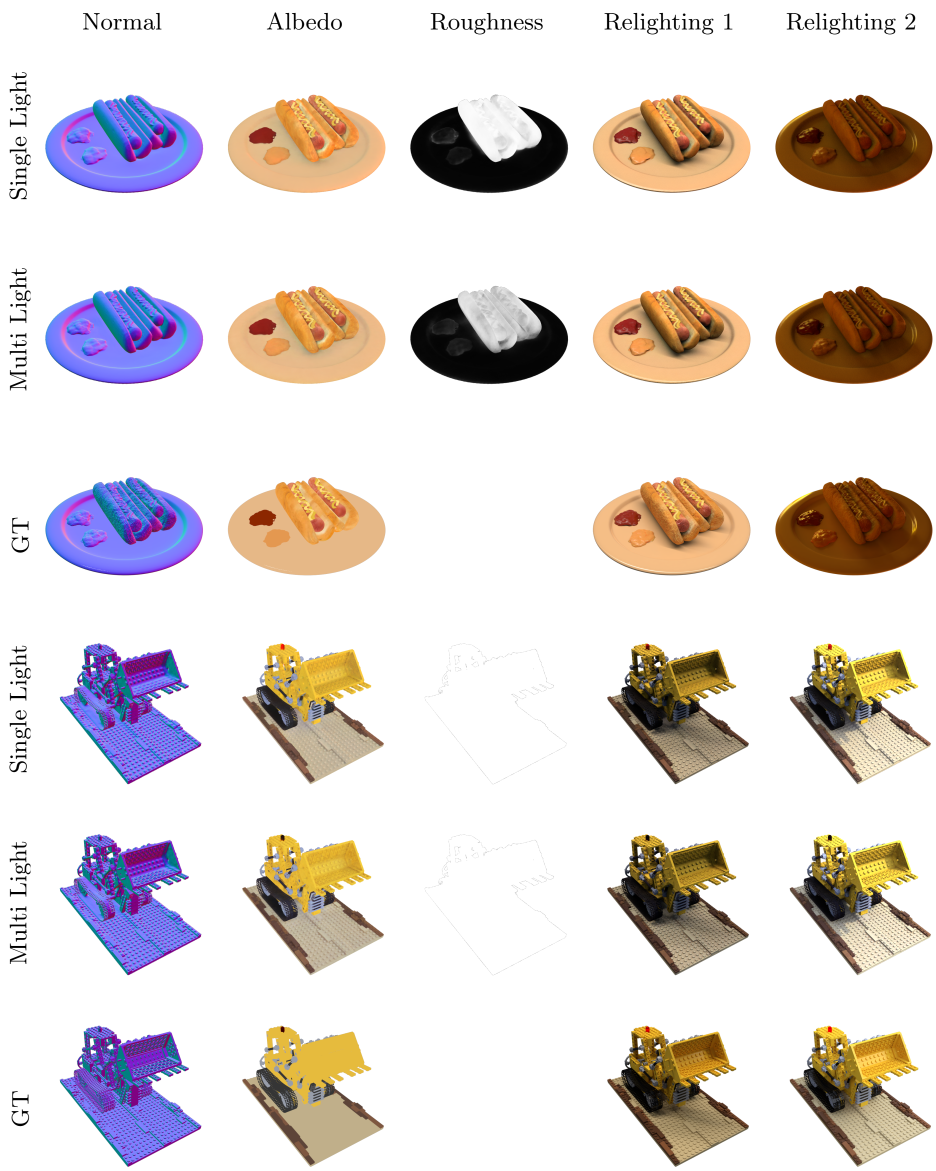

Fig. 2: Additional TensoIR-Synthetic Results [4] Results.

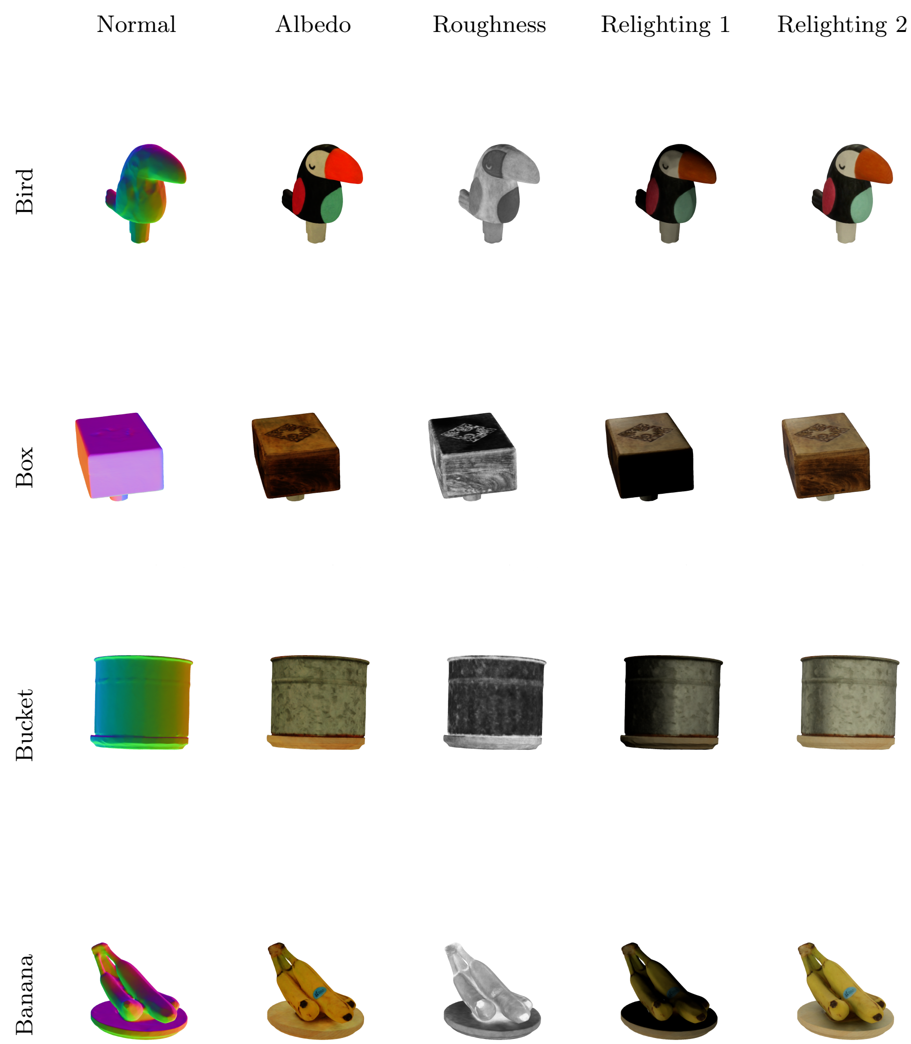

Fig. 3: Additional Open Illumination [5] Results.

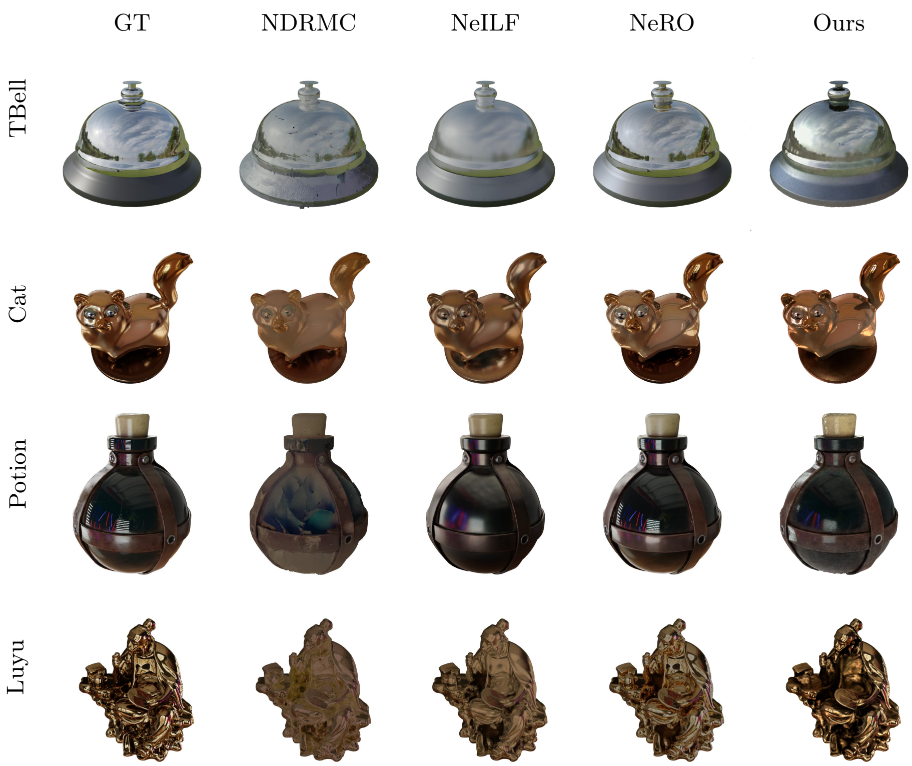

Fig. 4: Glossy Synthetic [6] Results. Note that our results are relit with *direct light only* (our codebase only supports direct relighting), while the other methods are relit using blender with exported meshes.