Flash Cache: Reducing Bias in Radiance Cache Based Inverse Rendering

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

Abstract. State-of-the-art techniques for 3D reconstruction are largely based on volumetric scene representations, which require sampling multiple points to compute the color arriving along a ray. Using these representations for more general inverse rendering — reconstructing geometry, materials, and lighting from observed images — is challenging because recursively path-tracing such volumetric representations is expensive. Recent works alleviate this issue through the use of radiance caches: data structures that store the steady-state, infinite-bounce radiance arriving at any point from any direction. However, these solutions rely on approximations that introduce bias into the renderings and, more importantly, into the gradients used for optimization. We present a method that avoids these approximations while remaining computationally efficient. In particular, we leverage two techniques to reduce variance for unbiased estimators of the rendering equation: (1) an occlusion-aware importance sampler for incoming illumination and (2) a fast cache architecture that can be used as a control variate for the radiance from a high-quality, but more expensive, volumetric cache. We show that by removing these biases our approach improves the generality of radiance cache based inverse rendering, as well as increasing quality in the presence of challenging light transport effects such as specular interreflections.

1 Introduction

Volume rendering has been remarkably effective for novel view synthesis and 3D reconstruction, as demonstrated by approaches such as Neural Radiance Fields (NeRFs) [29] and Gaussian Splatting [18]. However, it is difficult to use volume rendering for the more general inverse rendering problem of recovering materials and lighting in addition to geometry. The issue is largely computational: volume rendering techniques generally query the appearance and density of the volume at many sample points to compute the light arriving along each ray. Combining this with physically-accurate rendering models that simulate global illumination (instead of simple shading models that only consider direct lighting) dramatically increases computational complexity, because evaluating the rendering integral requires recursively casting additional scattered rays from each sample location.

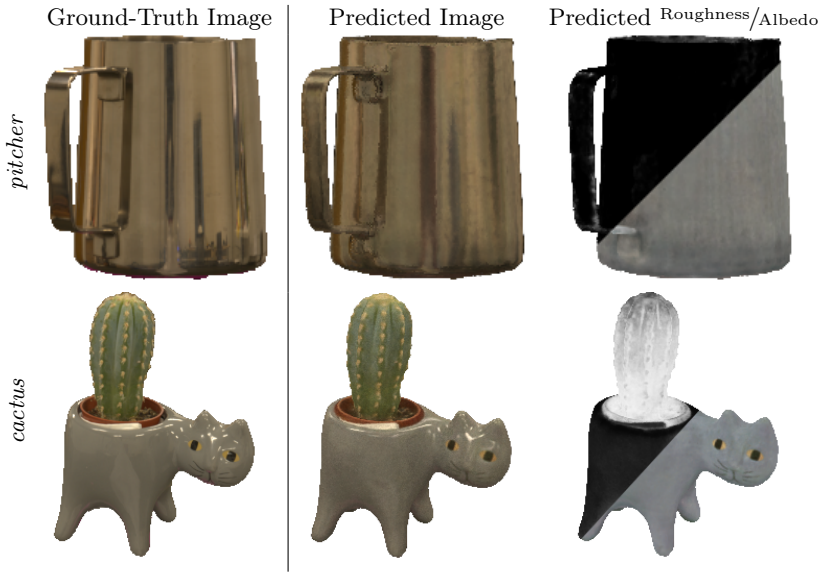

Fig. 1: Results on two captured scenes from the Stanford ORB dataset [21]. Our model synthesizes realistic novel views of real-world objects that exhibit complicated specular materials, and recovers accurate material roughnesses (black represents a perfect mirror, white is perfectly rough) and albedo maps.

Recent methods [15, 49] have leveraged radiance caches to simulate global illumination tractably within an inverse rendering optimization pipeline. Instead of integrating over the domain of n-bounce light paths, radiance caches store a representation of the steady-state light arriving at any point in the volume from any direction. This is a powerful paradigm that removes the need for recursively evaluating the rendering integral by considering a single reflection of the incoming steady-state (as opposed to direct) light at any point on the scene geometry towards the direction of the viewer.

Existing works that combine radiance caches with volumetric inverse rendering fall into two categories. The first uses trained NeRF models as radiance caches, and volume renders the NeRF along rays from points within the scene to estimate incoming light. This type of radiance cache has high quality but tends to be computationally expensive to evaluate; methods that use NeRF as a radiance cache, such as TensoIR [15], only compute the reflection integral at a single point (the expected ray termination) along each camera ray. The second category relies on cheap radiance caches such as a small MLPs, as in InvRender [49] and NeRO [25], to evaluate incoming lighting with a single ray query. Though fast, these caches fail to capture high-frequency and near-field illumination. We observe that both of these existing approaches introduce bias into the pipeline: the first category renders a NeRF model at only a single evaluation along each camera ray, which results in a biased estimation of the volume rendering integral, and the second category approximates incoming light with a low-capacity model, which results in an inaccurate and biased estimation of the reflection integral.

We present a system for volumetric inverse rendering based on the basic goal of avoiding introducing bias into the rendering procedure as much as possible, while leveraging variance reduction techniques to keep the rendering model computationally tractable. This is made possible by the following key technical contributions:

- A strategy for efficiently estimating the light reflected by volumetric geometry along each camera ray by using a neural field to parameterize a spatially-varying mixture of von-Mises Fisher distributions for light importance sampling. This captures spatially-varying illumination effects such as light occlusion and interreflections and lets our system efficiently estimate the rendering equation with fewer queries of the radiance cache.
- An architecture for a lightweight high-capacity radiance cache that can query the incoming illumination from any direction at any point within the volume, along with a strategy for using this cache as a control variate to avoid introducing rendering bias. This lets us efficiently model high-frequency and near-field illumination without requiring the full volume rendering necessary when using NeRF as a radiance cache.

We demonstrate that our system recovers improved geometry and materials over prior work, and in particular enables more accurate recovery of material parameters in regions affected by near-field lighting effects such as specular interreflections. Furthermore, since our system does not use a simplified model of volumetric geometry, it preserves the advantages of NeRF-like volumetric representations for capturing thin structures. We validate the benefits of our system with evaluation on existing benchmarks and rendered scenes that exhibit challenging lighting effects.

2 Related Work

Inverse rendering. Inverse rendering aims to recover scene attributes like materials, lighting, and geometry, from a set of images [1,8,36,46]. A common approach is to use a physically-based model of light transport [16,35,41] and differentiable rendering so that it is possible to leverage gradient-based optimization for decomposing the scene's appearance into its constituent parts [7,14,22]. Recent advances in view synthesis demonstrate the benefit of volume rendering neural field representations for the task of reproducing realistic images of a captured scene from unseen views [2, 29, 42]. Neural field representations also improve inverse rendering methods, combining ideas from volume rendering to represent geometry and physically-based rendering to represent appearance [3, 12, 26, 32, 40].

Variance-reduction for inverse rendering. Among the different paradigms for inverse rendering using neural fields, cache-based inverse rendering is the most closely related to ours. These methods rely on using a radiance cache for storing the incoming radiance at every point and along every direction. These radiance caches capture global illumination effects, replacing a prohibitively expensive recursive path tracing procedure with a much cheaper cache look-up. This technique has been used extensively in the past for computing indirect illumination

and global illumination effects [20, 43], and consequently used for real-time rendering applications [31, 38, 39].

In the context of inverse rendering, radiance caches are used for storing the hemisphere of incoming radiance at every point. This hemisphere is then integrated against the local BRDF to yield outgoing radiance, which is optimized to match the observed image pixels. NeRFactor [48] only considers direct illumination effects for decomposing lights and materials, but caches the visibility of each secondary ray into an MLP to model self occlusions. Neural Incident Light Field (NeILF) [44] and Neural Reflective Objects (NeRO) [25] and InvRender [49] parameterize the radiance cache using an MLP modeling the incident light field. However, they only supervise the outgoing radiance, which does not enforce consistency between the radiance cache and incoming radiance. An improved version of NeILF [47] solves this by penalizing differences between the radiance cache and the corresponding outgoing radiance. These methods all use inexpensive caches which tend to fail to capture high-frequency and near-field illumination. Most recently, concurrent work to ours from Ling *et al.* [23] uses a NeRF-based cache, along with an optimizable mixture of Gaussians model for importance sampling the 3D distribution of light sources in the scene. Our work also uses importance sampling to reduce Monte Carlo noise, but our distributions account for occlusions — a key part of variance reduction [41]. Along with importance sampling, we further reduce noise using control variates, two techniques that have been extensively explored in the context of forward rendering [4, 30, 34].

3 Preliminaries

In this work, we consider the problem of inverse rendering of geometry, materials, and lighting from a set of input images. We leverage a combination of volumetric rendering and physically-based Monte Carlo rendering in order to accomplish this task. In particular, we optimize the parameters Φ of a scene representation using a regularization term $\mathcal{L}_{reg}$ and a loss term such that the color value $L_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)$ incident at a pixel corresponding to a camera ray $(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$ — where $\mathbf{o}$ is its origin and $\boldsymbol{\omega}_{\mathrm{o}}$ its direction — matches with the corresponding measured image pixel $I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$:

$$\min_{\Phi} \sum_{\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}} \| I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) - L_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi) \|^2 + \mathcal{L}_{reg}(\Phi) \tag{1}$$

3.1 Volume Rendering and Neural Radiance Fields

In the volume rendering setting, the incident radiance L_{i} of a ray $(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}})$ passing through a neural volume, is computed by integrating the differential outgoing radiance L_{o} at every point along that ray:

$$L_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) = \int_0^{\infty} L_{\mathrm{o}}(\mathbf{x}(t), \boldsymbol{\omega}_{\mathrm{o}}) T(\mathbf{o}, \mathbf{x}(t)) \sigma(\mathbf{x}(t)) dt, \tag{2}$$

where $T(\mathbf{o}, \mathbf{x}(t)) = \exp(-\int_0^t \sigma(\mathbf{x}(s))\, ds)$ denotes the transmittance from $\mathbf{o}$ to $\mathbf{x}(t)$, and $\mathbf{x}(t) = \mathbf{o} - t\boldsymbol{\omega}_\mathrm{o}$ is a point along the ray parameterized by t. This integral is generally intractable to evaluate analytically, but it can be approximated via quadrature [27], by sampling density and radiance at a set of points $\{\mathbf{x}(t_k)\}_{k=1}^{\mathrm{N}}$:

$$\hat{L}_\mathrm{i}(\mathbf{o}, \boldsymbol{\omega}_\mathrm{o}) = \sum_{k=1}^{\mathrm{N}} w_k\, L_\mathrm{o}(\mathbf{x}(t_k), \boldsymbol{\omega}_\mathrm{o}), \tag{3}$$

$$w_k = \left(1 - e^{-\sigma(\mathbf{x}(t_k))(t_{k+1} - t_k)}\right) \exp\left(-\sum_{j=1}^{k-1} \sigma(\mathbf{x}(t_j))(t_{j+1} - t_j)\right).$$

In NeRF [29], the volume density σ and outgoing radiance L_o are parameterized using a multi-layer perceptron (MLP). Follow-up work [6, 9, 33] relies on optimizable grids or hash tables along with smaller neural networks. In both cases, a function maps every position $\mathbf{x}$ and direction $\boldsymbol{\omega}_\mathrm{o}$ into the volume density $\sigma(\mathbf{x})$ at that location and the outgoing radiance from it in the specified direction $L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$. Together these give the radiance along the ray using Equation 3.

3.2 Physics-Based Monte Carlo Light Transport

We use ideas from physically-based rendering for combining incoming radiance L_i with a BRDF f to produce outgoing radiance L_o using the rendering equation:

$$L_\mathrm{o}(\mathbf{x}(t), \boldsymbol{\omega}_\mathrm{o}) = \int_\Omega f(\mathbf{x}(t), \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i})\, d\boldsymbol{\omega}_\mathrm{i}\,, \tag{4}$$

where $\Omega = \{\boldsymbol{\omega}_\mathrm{i} \colon \mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i} > 0\}$ is the positive hemisphere with respect to normal $\mathbf{n}$.

Like the volume rendering integral in Equation 2, evaluating the rendering equation analytically is, in general, intractable. Most approaches therefore use Monte Carlo integration in order to approximate Equation 4. In particular, if we randomly sample M incoming directions from some distribution p over incoming directions, $\boldsymbol{\omega}_\mathrm{i}^{(1)}, \ldots, \boldsymbol{\omega}_\mathrm{i}^{(\mathrm{M})} \sim p(\boldsymbol{\omega}_\mathrm{i})$, then we can construct an unbiased estimator:

$$\hat{L}_\mathrm{o}(\mathbf{x}(t), \boldsymbol{\omega}_\mathrm{o}) = \frac{1}{\mathrm{M}} \sum_{j=1}^{\mathrm{M}} f\left(\mathbf{x}(t), \boldsymbol{\omega}_\mathrm{i}^{(j)}, \boldsymbol{\omega}_\mathrm{o}\right) L_\mathrm{i}\left(\mathbf{x}(t), \boldsymbol{\omega}_\mathrm{i}^{(j)}\right) \left(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}^{(j)}\right) \Big/ p\left(\boldsymbol{\omega}_\mathrm{i}^{(j)}\right), \tag{5}$$

where performing optimization using the estimator in Equation 5 is, in expectation, equivalent to optimizing using Equation 4 [19].

We can now plug the Monte Carlo-based estimator in Equation 5 into the volume rendering estimator of Equation 3 to arrive at the combined estimator:

$$\hat{L}_\mathrm{i}(\mathbf{o}, \boldsymbol{\omega}_\mathrm{o}) = \sum_{k=1}^{\mathrm{N}} \frac{w_k}{\mathrm{M}} \sum_{j=1}^{\mathrm{M}} f\left(\mathbf{x}(t_k), \boldsymbol{\omega}_\mathrm{i}^{(j,k)}, \boldsymbol{\omega}_\mathrm{o}\right) \hat{L}_\mathrm{i}\left(\mathbf{x}(t_k), \boldsymbol{\omega}_\mathrm{i}^{(j,k)}\right) \left(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}^{(j,k)}\right) \Big/ p\left(\boldsymbol{\omega}_\mathrm{i}^{(j,k)}\right) \tag{6}$$

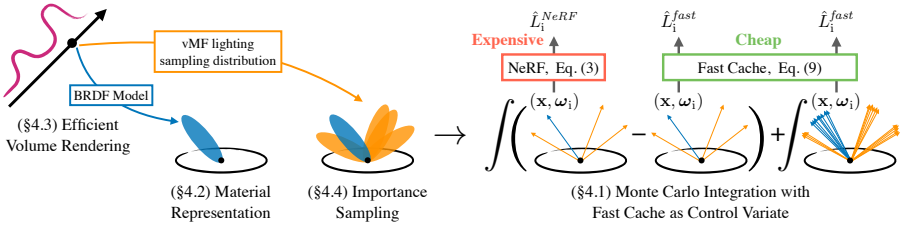

Fig. 2: Our pipeline for physically-based inverse rendering combines volume rendering with physically-based Monte Carlo rendering to recover geometry, materials, and lighting. We first use an estimator (Equation 11) to approximate the volume rendering integral (Equation 2) with only a single sample point per ray. Given this sample point, we use a spatially-varying mixture-of-vMFs (Section 4.4) to importance-sample directions based on the distribution of incoming radiance. We then produce a low-variance estimate of the rendering equation (Equation 4) using a NeRF as a radiance cache, and a fast cache approximation of the NeRF (Equation 9) that is far less expensive to evaluate.

Unfortunately, Equation 6 requires recursive evaluation of both the volume rendering integral and the rendering equation, since $\hat{L}_\mathrm{i}$ is now defined with respect to itself. If one is not careful, the recursive nature of the rendering equation can quickly lead to a rapid increase in computational cost.

To deal with this issue, we return to the discussion in Section 3.1. It is possible to define a neural volume that *directly* predicts outgoing radiance, without requiring recursive evaluation. In other words, we can replace the incoming radiance $\hat{L}_\mathrm{i}$ with the output of a neural radiance field $\hat{L}_\mathrm{i}^{cache}$, which relates the incoming and outgoing radiance according to Equation 3. It is worth remarking that for this to be useful, evaluating the radiance cache $\hat{L}_\mathrm{i}^{cache}$ for a given ray should be cheaper than performing recursive path tracing.

4 Method

Our inverse rendering method is based on a volumetric rendering approach to physically-based rendering, as described by Equation 6. However, naïvely using the rendering model of Equation 6 is extremely expensive for two main reasons. First, it requires volumetrically rendering the incoming radiance at every point along the ray. Second, for every such point we need to use Monte Carlo integration, which typically requires many samples of incoming radiance. Our work aims to resolve both issues without introducing bias in the process, which may adversely affect the recovery of materials and lighting.

In this section, we present the design of our system. We begin by describing the incoming radiance cache in Section 4.1, along with a novel "fast cache" architecture that is cheap to evaluate. Designing this cache as a control variate for incoming illumination allows us to use many incoming rays, which increases the accuracy of our method without requiring significant compute. In Section 4.2, we

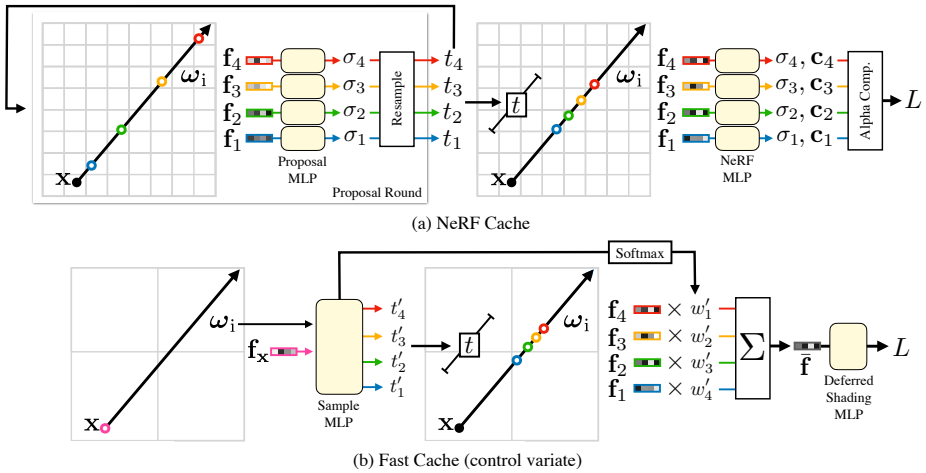

Fig. 3: An illustration of our two radiance cache architectures. (a) The NeRF-based cache, which is accurate yet expensive to evaluate, uses proposal MLPs to predict sample point distances $\{t_k\}$ through an iterative resampling procedure. These distances are then used to generate sample points at which we query density and outgoing radiance. These quantities are combined to approximate the volume rendering integral via quadrature 3. (b) Rather than predict sample point distances via proposal sampling, our fast cache directly outputs distances $\{t'_k\}$ as well as weights $\{w'_k\}$ given incoming ray parameters $(\mathbf{x}, \boldsymbol{\omega}_\text{i})$. The distances are used to generate sample points at which we query features from an NGP grid. The resulting features are averaged together, and then fed through a deferred shading MLP which predicts the final color for the incoming ray. Though our illustration depicts four samples in both cases, the NeRF-based cache (a) uses 64 samples for the first two resampling rounds and 32 samples for the final one, while our fast cache only uses 8 samples overall.

discuss our physically-based material representation. Finally, in Sections 4.3 and 4.4, we present our efficient sampling schemes based on a simple unbiased estimator for the volumetric rendering integral, and a novel optimizable occlusion-aware spatially-varying importance sampling scheme.

4.1 Incoming Radiance Cache

Our incoming radiance cache is based on the architecture of Zip-NeRF [2], and our appearance model is based on Ref-NeRF [42]. In particular, we borrow Zip-NeRF's usage of a multi-resolution hash grid as a neural graphics primitive (NGP) [33]. See Figure 3(a) for an illustration of the NeRF-based radiance cache, and see a complete description of it in the supplement.

In order to increase the efficiency of our method, we wish to use an accelerated cache for querying incoming radiance instead of relying on an expensive NeRF-based cache which requires evaluating many samples per incoming ray. We therefore design a "fast cache" and supervise it to match the incoming radi-

ance of the expensive, NeRF-based cache. See Figure 3 for a diagram comparing the two caches.

Our fast cache works by first passing the incident ray's origin and direction ($\mathbf{x}$ and $\boldsymbol{\omega}_\mathrm{i}$) through an optimizable function g_sample, implemented as a low-resolution hash-encoding followed by a small sample MLP. This sampling function returns a set of sample point distances along the ray and their corresponding weights:

$$\{t'_k, w'_k\}_{k=1}^{\mathrm{S}} = g_\mathrm{sample}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}). \tag{7}$$

Each distance t'_k corresponds to a new sample point $\mathbf{x}'_k = \mathbf{x} + t'_k\boldsymbol{\omega}_\mathrm{i}$ along the incident ray. Each of these samples is passed through a separate low-resolution NGP to yield a set of corresponding features $\{\mathbf{f}_k\}_{k=1}^{\mathrm{S}}$, which are combined with their corresponding normalized (via a softmax) weights w'_k to yield a ray feature vector which encodes the appearance of that ray. This feature is then decoded into the incident radiance by using a small MLP, similar to the "deferred shading" approach of Hedman *et al.* [13]:

$$\hat{L}_\mathrm{i}^{fast} = \mathrm{MLP}_\mathrm{color}\left(\sum_{k=1}^{\mathrm{S}} w'_k\mathbf{f}_k\right). \tag{8}$$

Even for a small number of samples (we use $\mathrm{S} = 8$ for all experiments), we find that this design serves as an effective cache for incoming radiance that captures high-frequency near-field illumination. See Figure 3 for an overview of the NeRF-based and fast caches, as well as figure 5 for an example of the output of the fast cache. Our supplement contains more details about both models.

Instead of simply replacing the NeRF-based cache with our fast cache, we combine the accuracy of the NeRF-based cache with the efficiency of the fast cache by treating the fast cache as a control variate [17,34] for incoming radiance. In order to do this, we sample two sets of incident directions using an importance sampling distribution p (to be introduced in Section 4.4): M' samples $\{\boldsymbol{\omega}_\mathrm{i}^{\prime(j)}\}_{j=1}^{\mathrm{M}'}$, and $\mathrm{M} \ll \mathrm{M}'$ additional independent samples $\{\boldsymbol{\omega}_\mathrm{i}^{(j)}\}_{j=1}^{\mathrm{M}}$. We then use the M' first samples to evaluate the fast cache $\hat{L}_\mathrm{i}^{fast}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$ via Equation 8, and the other M samples to evaluate the error between the NeRF-based cache $\hat{L}_\mathrm{i}^{NeRF}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$ and the fast cache. We then add the two estimates to arrive at an unbiased estimator for the outgoing radiance:

$$\begin{aligned}
\hat{L}_\mathrm{o}^{fast} &= \frac{1}{\mathrm{M}'}\sum_{j=1}^{\mathrm{M}'} f\left(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^{\prime(j)}, \boldsymbol{\omega}_\mathrm{o}\right)\hat{L}_\mathrm{i}^{fast}\left(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^{\prime(j)}\right)\left(\mathbf{n}\cdot\boldsymbol{\omega}_\mathrm{i}^{\prime(j)}\right)\Big/ p\left(\boldsymbol{\omega}_\mathrm{i}^{\prime(j)}\right), \\
\Delta\hat{L}_\mathrm{o} &= \frac{1}{\mathrm{M}}\sum_{j=1}^{\mathrm{M}} f\left(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^{(j)}, \boldsymbol{\omega}_\mathrm{o}\right)\left(\hat{L}_\mathrm{i}^{NeRF}\left(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^{(j)}\right) - \hat{L}_\mathrm{i}^{fast}\left(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}^{(j)}\right)\right)\left(\mathbf{n}\cdot\boldsymbol{\omega}_\mathrm{i}^{(j)}\right)\Big/ p\left(\boldsymbol{\omega}_\mathrm{i}^{(j)}\right), \\
\hat{L}_\mathrm{o} &= \hat{L}_\mathrm{o}^{fast} + \Delta\hat{L}_\mathrm{o}\,.
\end{aligned} \tag{9}$$

Note that this requires M evaluations of the NeRF-based cache and $\mathrm{M} + \mathrm{M}'$ evaluations of the fast cache. Given sufficiently accurate cache (see Section 5 for

an example) that is indeed cheap to evaluate, this process results in the quality of using M' incoming rays from the accurate NeRF-based cache, by only requiring a fraction of the computation.

On top of the two caches, we also have an explicit model for far-field illumination based on an MLP, $\hat{L}_{\mathrm{i}}^{env}(\boldsymbol{\omega}_{\mathrm{i}}) = \mathrm{MLP}_{\mathrm{env}}(\boldsymbol{\omega}_{\mathrm{i}})$, which is multiplied by the transparency of the ray $1 - \sum_{k=1}^{\mathrm{N}} w_k$ and added to the radiance of the cache.

4.2 Material Representation

Like Ref-NeRF [42], we extract analytic normals using the negative gradient of the density field, and leverage "predicted normals" output by the NGP, which are tied to the analytic normals using an L2 loss. Our material model is based on the Disney-GGX BRDF [5], which is combined with the predicted normals to obtain the radiance value as in Equation 4. The spatially-varying parameters of the BRDF model are output by a separate NGP, g_{material}:

$$\Big(m(\mathbf{x}), r(\mathbf{x}), \mathbf{a}(\mathbf{x})\Big) = g_{\mathrm{material}}(\mathbf{x})\,, \tag{10}$$

where m is metalness, r is roughness, and $\mathbf{a}$ is albedo.

4.3 Efficient Gradients for Volume Rendering

Evaluating the volume rendering integral using the quadrature technique in Equation 3 means querying the appearance and geometry of the volume at N sample point locations per ray, each of which requires Monte Carlo integration. However, this can be accelerated considerably by replacing the sum with a Monte Carlo estimate along the ray using $\mathrm{K} \ll \mathrm{N}$ samples [11]. We draw K indices from a categorical distribution $j_1, \ldots, j_{\mathrm{K}} \sim \mathrm{Cat}(w_1, \ldots, w_{\mathrm{N}})$, and replace Equation 3 with:

$$\hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} L_{\mathrm{o}}(\mathbf{x}(t_{j_k}), \boldsymbol{\omega}_{\mathrm{o}})\,. \tag{11}$$

The estimator of Equation 11 is unbiased, *i.e.*, its expected value equals the quadrature value of Equation 3. While this still requires querying volume density for N points, we only need to evaluate the costly physically-based appearance at K points. Furthermore, if the render weight distribution has a single peak (often the case in practice), then this estimator has low variance, even for $\mathrm{K} = 1$.

4.4 Low-Variance Gradients for Monte Carlo Rendering

The unbiased estimator in Equation 11 reduces the number of secondary ray samples from $\mathrm{N} \cdot \mathrm{M}$ to $\mathrm{K} \cdot \mathrm{M}$. However, a large number of secondary rays $\mathrm{K} \cdot \mathrm{M}$ may be necessary for convergence, but may be prohibitively expensive. We reduce the number of samples by leveraging occlusion-aware importance sampler.

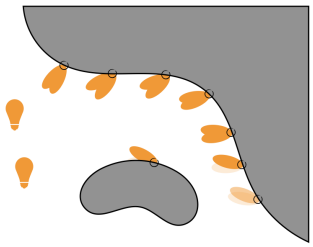

Fig. 4: An illustration of our occlusion-aware importance sampler. We use an NGP to predict the parameters of a mixture of vMFs distribution for each point in space (Equation 12), which can produce different lobes for different light sources. Because the parameters of these distributions are *spatially-varying*, if a light source is occluded from the perspective of one surface point, then the model can "turn off" the lobe for that light source.

Importance sampling for incoming radiance is common technique in physically-based forward rendering and inverse rendering. Here, we aim to build a lightweight importance sampler that is capable of approximating the true distribution of incoming radiance, including occlusions, from both direct and indirect light sources. For this purpose, we use an NGP to map a point $\mathbf{x}$ into a set of 5L scalars representing the parameters $\{\boldsymbol{\mu}'_\ell, \kappa_\ell, \lambda_\ell\}_{\ell=1}^{\mathrm{L}}$ which parameterize a mixture of von Mises-Fisher (vMF) distributions as:

$$q(\boldsymbol{\omega}; \mathbf{x}) = \frac{1}{Z} \sum_{\ell=1}^{\mathrm{L}} \lambda_\ell(\mathbf{x}) \, \mathrm{vMF}(\boldsymbol{\omega}; \boldsymbol{\mu}_\ell(\mathbf{x}), \kappa_\ell(\mathbf{x})), \text{ with } \boldsymbol{\mu}_\ell(\mathbf{x}) = \frac{\boldsymbol{\mu}'_\ell(\mathbf{x}) - \mathbf{x}}{\|\boldsymbol{\mu}'_\ell(\mathbf{x}) - \mathbf{x}\|}, \quad (12)$$

where $Z = \sum_{\ell=1}^{\mathrm{L}} \lambda_\ell(\mathbf{x})$ is the partition function used for normalizing q, and the coefficients $\{\lambda_\ell\}_{\ell=1}^{\mathrm{L}}$ and $\{\kappa_\ell\}_{\ell=1}^{\mathrm{L}}$ are nonnegative numbers representing the mixture weights and vMF concentration parameters respectively.

The parameterization of the means $\{\boldsymbol{\mu}'_\ell\}_{\ell=1}^{\mathrm{L}}$ in Equation 12 is designed so that a single vMF lobe can be assigned to the location of single (direct or indirect) light source in 3D, which is then projected onto the sphere.

We optimize the parameters describing the distribution q at every location to match the true distribution of incoming radiance, taking occlusions into account. Concretely, we use the same importance sampling scheme used for evaluating the rendering integral in Equation 5, to minimize the loss:

$$\mathcal{L}_{\mathrm{vMF}} = \frac{1}{\mathrm{M}} \sum_{j=1}^{\mathrm{M}} \left\| Z \cdot q\left(\boldsymbol{\omega}_{\mathrm{i}}^{(j)}; \mathbf{x}\right) - \bar{L}\left(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}}^{(j)}\right) \right\|^2 \Big/ p\left(\boldsymbol{\omega}_{\mathrm{i}}^{(j)}\right), \quad (13)$$

where p is a distribution used to generate the samples $\{\boldsymbol{\omega}_{\mathrm{i}}^{(j)}\}_{j=1}^{\mathrm{M}}$ (as well as the M' cheap cache samples used in Section 4.1), which is based the mixture distribution q and a standard material-based importance sampling distribution (see supplement for more details), and $\bar{L}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}}) = \|\hat{L}_{\mathrm{i}}^{NeRF}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})\|_2$ is the L2 norm of the incident radiance, taken across channels.

5 Experiments

Below, we provide implementation details of our approach, and describe how we validate it on both synthetic and real inverse rendering benchmarks.

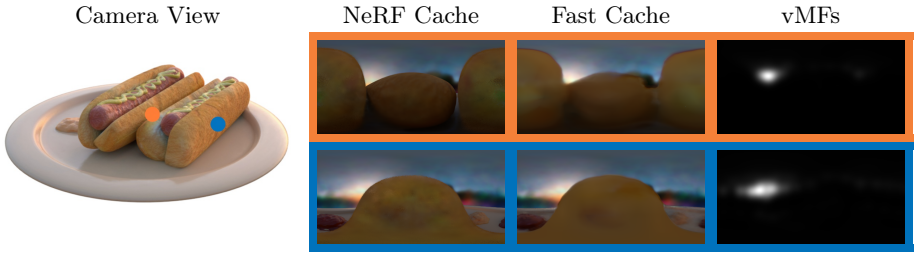

Fig. 5: *(left)* We choose two points on the hotdog for which to visualize distributions of incoming radiance. *(right)* We show (a) incoming radiance from the NeRF cache, which is high quality but slow to evaluate, (b) incoming radiance from the fast cache, which is slightly lower quality but fast to evaluate, and (c) the probability density of the spatially-varying mixture of vMFs, which is able to capture occlusions in the incoming radiance distribution.

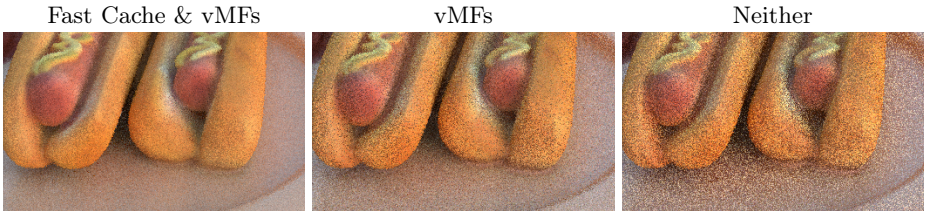

Fig. 6: We demonstrate the impact of both of our variance reduction strategies by showcasing a rendering using 16spp from the *Hotdog* scene in the TensoIR-synthetic dataset. (Left) We use both the fast cache and vMF importance sampler, (Middle) we use only vMF importance sampler, (Right) we use neither strategy.

5.1 Implementation Details

Model Parameters. Our NGP parameters for volumetric density closely follow those of Zip-NeRF [2]. We do proposal resampling with two additional NGPs with maximum resolutions of 1024 and 512, both of which, as well as the final density NGP, use of 2-layer 64-hidden-unit MLPs before emitting density. The material NGP as well as another NGP used for diffuse appearance in the cache (see supplement for more details) each has a maximum grid resolution of 2048 and a 2-layer 64-hidden-unit MLP. For the fast cache discussed in Section 4.1, we use an NGP of maximum resolution 256 with a 4-layer 128-hidden-unit MLP to predict $S = 8$ sample-point distances, and a maximum NGP of resolution 256 to predict features. For the incoming radiance importance sampler in Section 4.4, we use 128 vMF lobes, NGP with a maximum resolution of 2048, and a 2-layer 64-hidden-unit MLP. The hash maps of all NGPs have size 2^{19}.

Regularization. We implement a smoothness regularizer $\mathcal{L}_{BRDF}$ for BRDF parameters. In particular, we use a variant of TensoIR's smoothness regularizer, with an $L1$ rather than an $L2$ loss (see the supplement for more details). We also enforce a loss $\mathcal{L}_{consistency}$ that encourages the specular color and diffuse color from the NeRF cache to match the specular and diffuse lobes from physically-

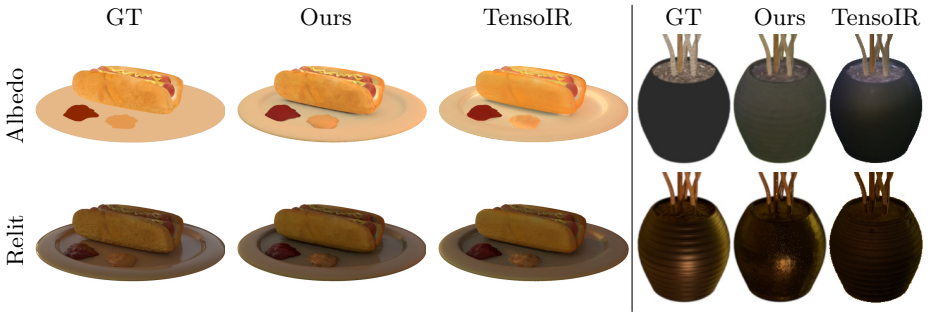

Fig. 7: A visualization of ground-truth albedo and relighting for the *hotdog* (left) and *ficus* (right, cropped) scenes in the Blender dataset [29], alongside our and TensoIR's results. Our method better decouples strong indirect illumination from the albedo (see the lip of the plate) while also reproducing more accurate highlights.

based rendering. This *indirectly* provides gradients through the physically-based model for volume rendering. We further apply the anti-aliased inter-level loss $\mathcal{L}_{interlevel}$ from Zip-NeRF to the proposal grids, and $L1$ regularization to the parameters of all density grids with a loss $\mathcal{L}_{density}$.

Additional Details. We perform two-stage optimization by (1) first optimizing the NeRF cache and an initial estimate of geometry to match a scene's input color images, and (2) optimizing geometry, materials, fast cache, and importance sampler through Equation 1. For the second stage, we use an $L2$ loss with the gradient trick [10], in order to ensure that our gradients for inverse rendering are unbiased. We leverage the same secondary rays that are used in rendering the physically-based model to supervise the fast cache and importance sampler. We *do not* apply full stop-gradients to any part of the pipeline, so that in the second stage geometry optimization can benefit from reflection and shading cues using the more accurate physically-based model of appearance. Altogether, we have the following:

$$\mathcal{L}_{reg} = \mathcal{L}_{normals} + \mathcal{L}_{BRDF} + \mathcal{L}_{consistency} + \mathcal{L}_{interlevel} + \mathcal{L}_{density} \tag{14}$$

All training and evaluation is carried out using a single NVIDIA A100 GPU.

5.2 Results on Synthetic Data

We first evaluate our approach on the TensoIR-synthetic dataset, which contains four scenes: *armadillo*, *ficus*, *hotdog*, and *lego*. Each scene consists of a set of 100 training views and 200 test views containing ground truth normals and albedo, as well as ground truth color under several different lighting conditions. We evaluate (a) the quality of our recovered normals, (b) the quality of our recovered albedo maps, (c) novel view synthesis quality for the *physically-based model*, and (d) relighting quality. Our method outperforms baselines in terms of albedo quality and relighting, while performing comparably to the best method for novel view synthesis. See Table 5, Figure 7, and the supplement for visualizations.

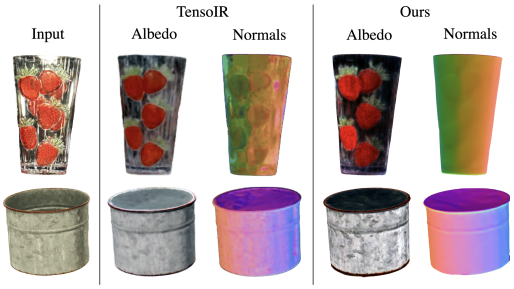

Fig. 8: Real results from Open Illumination [24]. Our normals are more accurate than those from TensoIR [15], and our albedo correctly removes specularities. Because of scale and color ambiguities, albedo maps are all visualized after using Photoshop's auto-tone, auto-contrast, and auto-color features.

Method	Normal	Albedo			Novel View Synthesis			Relighting		
	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRFactor	6.314	25.125	0.940	0.109	24.679	0.922	0.120	23.383	0.908	0.131
InvRender	5.074	27.341	0.933	0.100	27.367	0.934	0.089	23.973	0.901	0.101
TensoIR	4.100	29.275	0.950	0.085	**35.088**	**0.976**	**0.040**	28.580	**0.944**	0.081
Ours	**3.355**	**30.274**	**0.955**	**0.085**	34.908	0.965	0.064	**29.724**	**0.944**	**0.080**

Table 1: TensoIR-Synthetic Dataset [15] Results. Above, we present results of our method and 3 baselines on the TensoIR-synthetic dataset. We outperform all methods in terms of normal MAE, albedo quality, and relighting quality, while performing comparably to TensoIR in terms of novel view synthesis quality.

5.3 Results on Real Data

We further evaluate on 10 real scenes from the Open Illumination dataset. Each scene has 38 training views and 10 test views under 13 different lighting conditions. We split this dataset into two parts: (a) one that contains mostly diffuse objects, and (b) one that contains mostly glossy/specular objects. We evaluate novel view synthesis quality for our physically-based model, as well as relighting quality. We tend to outperform TensoIR on scenes with specular materials (visualized in Figure 8), while TensoIR performs better for diffuse materials.

5.4 Ablations

We ablate various design decisions and parts of our pipeline on the TensoIR synthetic dataset. In particular, we show that all of the following improve result quality: (a) volume rendering vs. median depth sampling, (b) vMF-importance sampling, and (c) our fast cache as a control variate for incoming illumination.

6 Conclusion

In this paper, we have outlined a system for volumetric inverse rendering of geometry, materials, and lighting, where we reduce bias in two ways: (a) by approximating the volume rendering integral (Equation 2), and (b) by using Monte Carlo sampling to evaluate the rendering integral (Equation 5). In particular,

Table 2: A comparison to TensoIR [15] on the Open Illumination dataset [24] split into *diffuse* and *specular* scenes.

Method	PSNR↑ (*diffuse*)		PSNR↑ (*specular*)	
	NVS	Relighting	NVS	Relighting
TensoIR	**32.043**	**30.932**	27.115	26.955
Ours	31.781	29.679	**27.180**	**27.810**

Table 3: Ablations We progressively ablate (a) our control variate scheme (Equation 9), (b) our vMF importance sampler (Section 4.4), and (c) our estimator (Section 11), replacing it with median depth sampling.

Method	NVS PSNR↑	Albedo PSNR↑	MAE↓
Ours	**34.915**	**30.345**	3.355
(a) Ours, no variate	34.629	30.329	**3.352**
(b) Ours, no variate, no sample	34.653	30.237	**3.352**
(c) Ours, no variate, no sample, no estimator	34.249	29.838	3.491

we accelerate volume rendering while providing correct *expected* gradients for optimization, and we accelerate the evaluation of the rendering equation by combining the high-quality NeRF-based radiance cache with an occlusion-aware importance sampler and a fast-cache-based control variate. This is in contrast to existing approaches which either use a single fixed intersected surface per ray, or use low-capacity and blurry representations of incoming radiance; both of which introduce bias. We show that our system achieves better quality material reconstruction and relighting than baselines on standard benchmarks containing both real and synthetic data, and justify our design decisions with ablations.

Limitations & Future Work: Despite our method's reduced bias, the quadrature approximation to the volume rendering integral (Equation 3) is still a source of bias. Our proposed approach can potentially be combined with unbiased volume rendering methods such as differential ratio tracking [Nimier-David et al. 2022].

Our fast cache sometimes struggles to capture particularly thin structural details in near-field incoming illumination. While our control variate scheme prevents this from introducing bias into the optimization process, it does increase gradient variance. In principle, any implementation of a fast cache may be used, such as a high-quality baked representation of a neural radiance field [13, 37, 45]. We leave investigation of alternative fast cache architectures up to future work.

Note that we do not claim that our method is more efficient than existing approaches; rather, that we provide efficient tools to accelerate radiance cache queries and reduce rendering variance, allowing our less-biased & more physically accurate system to run tractably on commodity GPUs. Our method could potentially be combined with other tools (e.g. denoisers) to further reduce secondary ray sample counts, accelerate inference, and increase convergence speed.

Acknowledgements This work was carried out while Benjamin was an intern at Google Research. Authors thank Rick Szeliski, Aleksander Holynski, and Janne Kontkanen for fruitful discussions, as well as Isabella Liu for help with TensoIR comparisons. Benjamin Attal is supported by a Meta Research PhD Fellowship. Matthew O'Toole acknowledges support from NSF CAREER 2238485 and a Google gift.

References

1. Barron, J.T., Malik, J.: Intrinsic Scene Properties from a Single RGB-D Image. CVPR (2013)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. ICCV (2023)
3. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition (2020), arXiv:2008.03824
4. Bitterli, B., Wyman, C., Pharr, M., Shirley, P., Lefohn, A., Jarosz, W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. ACM Trans. Graph. (2020)
5. Burley, B., Studios, W.D.A.: Physically-Based Shading at Disney. ACM Trans. Graph. (2012)
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: TensoRF: Tensorial radiance fields. ECCV (2022)
7. Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. NeurIPS (2019)
8. Debevec, P.: Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. ACM Trans. Graph. (1998)
9. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. CVPR (2022)
10. Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., Levin, A.: Inverse volume rendering with material dictionaries. ACM Trans. Graph. (2013)
11. Gupta, K., Hasan, M., Xu, Z., Luan, F., Sunkavalli, K., Sun, X., Chandraker, M., Bi, S.: MCNeRF: Monte Carlo rendering and denoising for real-time NeRFs. ACM Trans. Graph. (2023)
12. Hasselgren, J., Hofmann, N., Munkberg, J.: Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. NeurIPS (2022)
13. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. ICCV (2021)
14. Jakob, W., Speierer, S., Roussel, N., Vicini, D.: Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. ACM Trans. Graph. (2022)
15. Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: TensoIR: Tensorial Inverse Rendering. CVPR (2023)
16. Kajiya, J.T.: The rendering equation. ACM Trans. Graph. (1986)
17. Kalos, M.H., Whitlock, P.A.: Monte Carlo Methods. John Wiley & Sons (2009)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics (2023)

19. Kloek, T., Van Dijk, H.K.: Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. Econometrica: Journal of the Econometric Society (1978)
20. Kriv´anek, J., Gautron, P.: Practical global illumination with irradiance caching. Springer Nature (2022)
21. Kuang, Z., Zhang, Y., Yu, H.X., Agarwala, S., Wu, E., Wu, J., et al.: Stanford-ORB: A Real-World 3D Object Inverse Rendering Benchmark. NeurIPS Datasets & Benchmarks Track (2023)
22. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable Monte Carlo Ray Tracing through Edge Sampling. ACM Trans. Graph. (2018)
23. Ling, J., Yu, R., Xu, F., Du, C., Zhao, S.: NeRF as Non-Distant Environment Emitter in Physics-based Inverse Rendering. arXiv:2402.04829 (2024)
24. 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)
25. 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)
26. Mai, A., Verbin, D., Kuester, F., Fridovich-Keil, S.: Neural microfacet fields for inverse rendering. ICCV (2023)
27. Max, N.: Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics (1995)
28. 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)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. ECCV (2020)
30. Müller, T., McWilliams, B., Rousselle, F., Gross, M., Novák, J.: Neural importance sampling. ACM Trans. Graph. (2019)
31. Müller, T., Rousselle, F., Novák, J., Keller, A.: Real-time neural radiance caching for path tracing. ACM Trans. Graph. (2021)
32. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting Triangular 3D Models, Materials, and Lighting From Images. CVPR (2022)
33. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. (2022)
34. Nicolet, B., Rousselle, F., Novak, J., Keller, A., Jakob, W., Müller, T.: Recursive control variates for inverse rendering. ACM Trans. Graph. (2023)
35. Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation. MIT Press (2023)
36. Ramamoorthi, R., Hanrahan, P.: A signal-processing framework for inverse rendering. ACM Trans. Graph. (2001)
37. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes. ACM Trans. Graph. (2023)
38. Scherzer, D., Nguyen, C.H., Ritschel, T., Seidel, H.P.: Pre-convolved radiance caching. Computer Graphics Forum (2012)
39. Silvennoinen, A., Lehtinen, J.: Real-time global illumination by precomputed local reconstruction from sparse radiance probes. ACM Transactions on Graphics (TOG) (2017)

40. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: NeRV: Neural reflectance and visibility fields for relighting and view synthesis. CVPR (2021)
41. Veach, E.: Robust Monte Carlo methods for light transport simulation. Stanford University (1998)
42. 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)
43. Ward, G.J., Rubinstein, F.M., Clear, R.D.: A ray tracing solution for diffuse interreflection. ACM Trans. Graph. (1988)
44. Yao, Y., Zhang, J., Liu, J., Qu, Y., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: NeiLF: Neural incident light field for physically-based material estimation. European Conference on Computer Vision (2022)
45. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. SIGGRAPH (2023)
46. Yu, Y., Debevec, P., Malik, J., Hawkins, T.: Inverse global illumination: Recovering reflectance models of real scenes from photographs. SIGGRAPH (1999)
47. Zhang, J., Yao, Y., Li, S., Liu, J., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: NeILF++: Inter-reflectable light fields for geometry and material estimation. ICCV (2023)
48. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: NeRFactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Trans. Graph. (2021)
49. Zhang, Y., Sun, J., He, X., Fu, H., Jia, R., Zhou, X.: Modeling indirect illumination for inverse rendering. CVPR (2022)

A Overview

In Section B, 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 C we prove the unbiasedness of our control variate scheme and our estimator for volume rendering quadrature. In Section D 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 E, we provide additional result figures and per-scene quantitative metrics. Please see our supplemental website for more results and videos.

B Additional Implementation Details

B.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).

B.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).

B.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 $\Delta\hat{L}_{\mathrm{o}}$.

B.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 [2]. 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 [28] 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:

$$\mathcal{L}_{photometric} = \frac{\left\| I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) - \mathbb{E}\left[\hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\right] \right\|^2}{\mathrm{stopgrad}\big(L_{\mathrm{i}}^{cache}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\big)} \tag{15}$$

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 [10], which gives correct gradients through this loss in expectation for the physically-based model. More concretely, we minimize:

$$\mathcal{L}_{photometric} = \frac{2\Big(I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) - \hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\Big)\,\mathrm{stopgrad}\Big(I(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}) - \hat{L}_{\mathrm{i}}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi)\Big)}{\mathrm{stopgrad}(\hat{L}_{\mathrm{i}}^{cache}(\mathbf{o}, \boldsymbol{\omega}_{\mathrm{o}}; \Phi))} \tag{16}$$

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.

B.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.

B.6 BRDF Model

As we discuss in the main text, we use the Disney-GGX BRDF [5], 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:

$$f(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \mathbf{x}) = f_{diffuse}(\mathbf{x}) + f_{specular}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \mathbf{x}) \tag{17}$$

$$f_{diffuse}(\mathbf{x}) = \frac{(1 - m(\mathbf{x}))\mathbf{a}(\mathbf{x})}{\pi} \tag{18}$$

$$f_{specular}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \mathbf{x}) = \frac{DFG}{4(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i})(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{o})} \tag{19}$$

We refer readers to Burley [5] and Liu *et al.* [25] for definitions of (D, F, G) — the normal distribution function (NDF), Fresnel, and geometry terms. We use the Trowbridge-Reitz distribution function [35] for the NDF D.

B.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 [35] 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 [35].

B.8 Normal Loss

As discussed in the paper, we emit predicted normals from the density NGP. Similar to Ref-NeRF [42] and TensoIR [15], we constrain our predicted normals to match the negative gradient of the density field with an L2 loss:

$$\mathcal{L}_{normals} = C_{normals} \sum_k w_k \left\| \mathbf{n}_k^{pred} - \mathbf{n}_k^{derived} \right\|^2 \tag{20}$$

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

$$\mathbf{n}_k^{derived} = -\frac{\nabla \sigma(\mathbf{x}_k)}{\|\nabla \sigma(\mathbf{x}_k)\|} \tag{21}$$

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.

B.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 16.

B.10 Smoothness Loss

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

$$\mathcal{L}_{BRDF} = C_{BRDF} \sum_k w_k \left| \frac{\beta(\mathbf{x}_k) - \beta(\mathbf{x}_k + \xi)}{\max(\beta(\mathbf{x}_k), \beta(\mathbf{x}_k + \xi))} \right| \lambda(\mathbf{x}, \mathbf{x} + \xi) \tag{22}$$

$$\lambda(\mathbf{x}, \mathbf{x} + \xi) = |\mathbf{a}_{pseudo}(\mathbf{x}_k) - \mathbf{a}_{pseudo}(\mathbf{x}_k + \xi)| \qquad \xi \sim \mathcal{N}(\mathbf{0}, \epsilon \mathrm{I}) \tag{23}$$

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

$$\mathbf{a}_{pseudo}(\mathbf{x}_k) = \frac{\mathbf{c}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{\mathbf{c}_{irradiance}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})} \tag{24}$$

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

C Proofs

C.1 Control Variates

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

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

C.2 Volume Rendering

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

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

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

D Additional Ablations

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

D.1 Secondary Sample Ablations

We report aggregate novel view synthesis PSNR, albedo PSNR, and normal MAE for three additional ablations in Table 4 as we vary the value of K in Equation 11. In practice, we find that $\mathrm{K} = 1$ provides the best results.

Table 4: Sample Number Ablations

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

We expect that $\mathrm{K} = 1$ 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.

E Additional Results

E.1 Per-Scene Results for TensoIR

We provide per-scene results for both the TensoIR-Synthetic dataset in Table 5. We additionally provide results for the multi-light TensoIR setting.

Table 5: Per-Scene TensoIR-Synthetic Dataset [15] Results.

	Method	Normal	Albedo			Novel View Synthesis			Relighting		
		MAE↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	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

E.2 Per-Scene Results for Open Illumination

We provide per-scene results for both the Open Illumination dataset in Table 6. 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.

E.3 Qualitative Results

Find additional qualitative results in Figures 9, 11, an 12as well as our videos on our supplemental website.

Table 6: Per-Scene Open Illumination dataset [24] 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. 9: Additional TensoIR-Synthetic Results [15] Results.

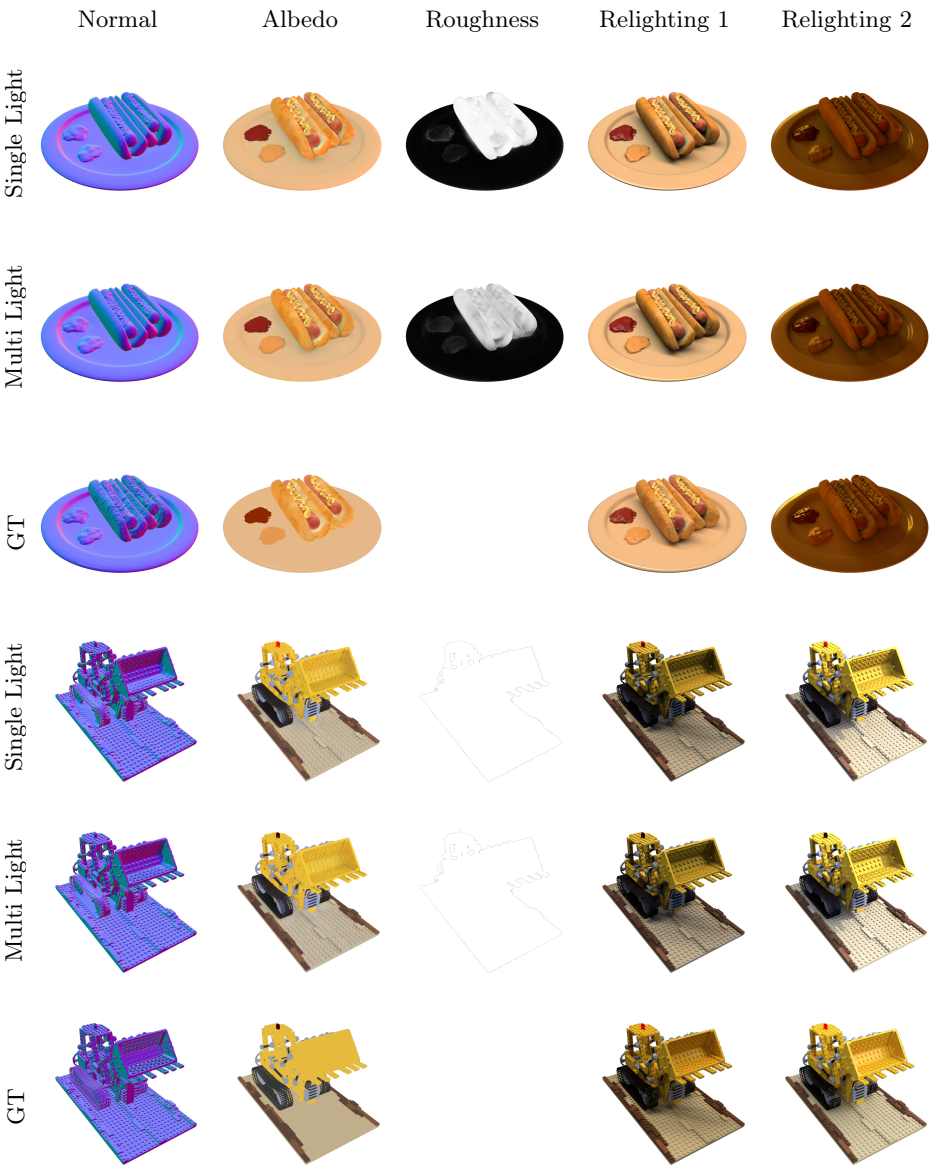

Fig. 10: Additional TensoIR-Synthetic Results [15] Results.

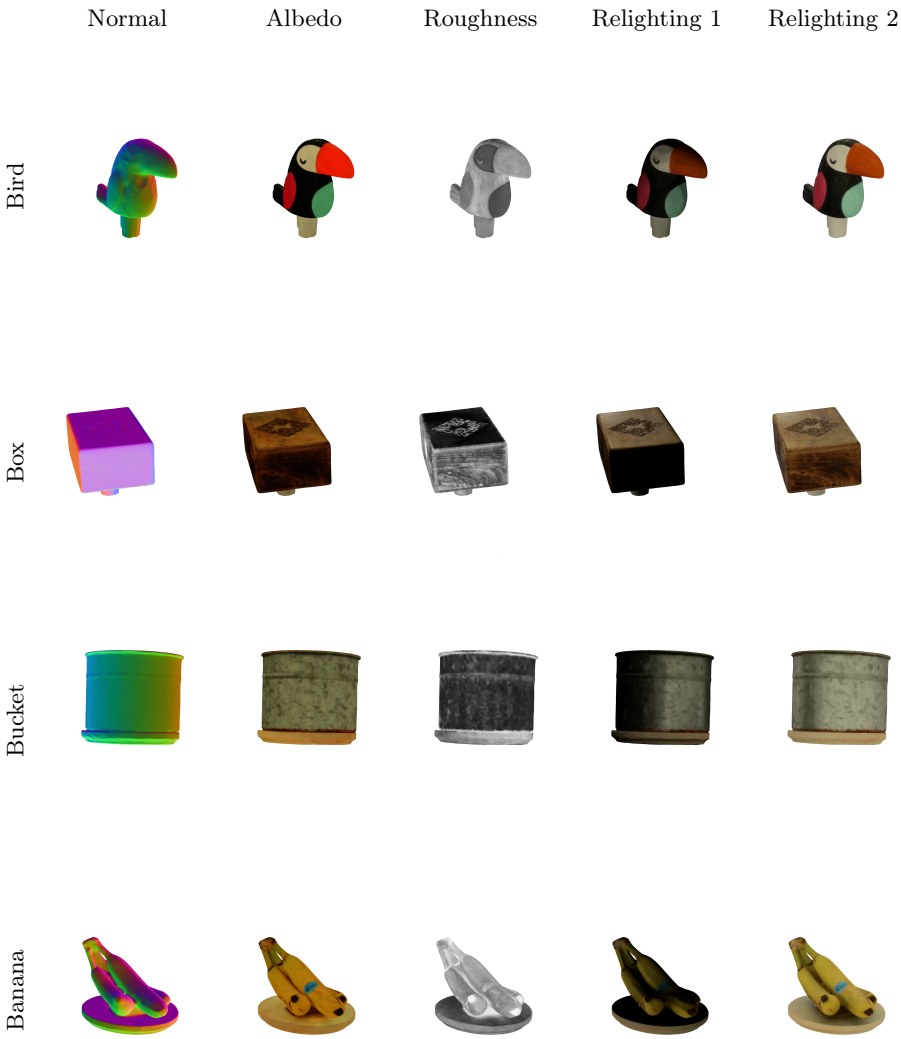

Fig. 11: Additional Open Illumination [24] Results.

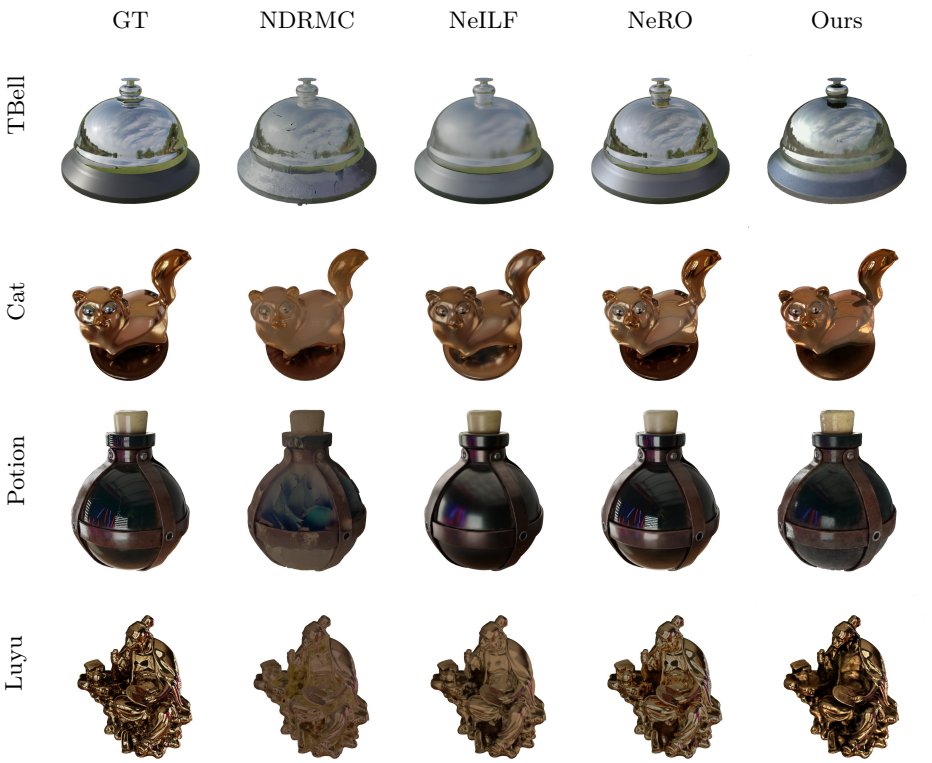

Fig. 12: Glossy Synthetic [25] Results. Note that our results are relit with *direct light only* (our codebase only supports direct relighting), while the other methods are relit using blender with exported meshes.