NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination

XIUMING ZHANG, MIT CSAIL
PRATUL P. SRINIVASAN, Google Research
BOYANG DENG, Google Research
PAUL DEBEVEC, Google Research
WILLIAM T. FREEMAN, MIT CSAIL & Google Research
JONATHAN T. BARRON, Google Research

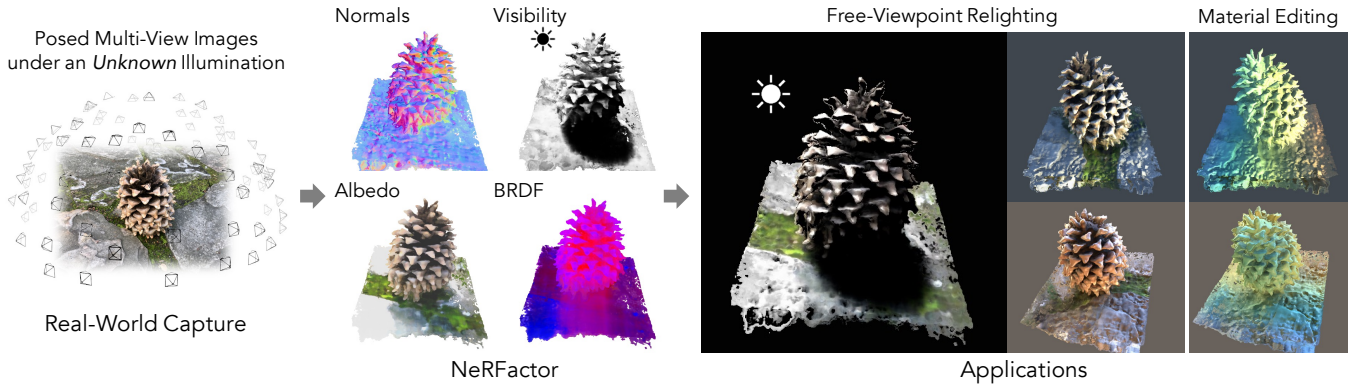


Fig. 1. Given a set of posed images of an object captured from multiple views under just one unknown illumination condition (left), Neural Radiance Factorization (NeRFactor) is able to factorize the scene into 3D neural fields of surface normals, light visibility, albedo, and material (center), which enables applications such as free-viewpoint relighting and material editing (right).

We address the problem of recovering the shape and spatially-varying reflectance of an object from posed multi-view images of the object illuminated by one unknown lighting condition. This enables the rendering of novel views of the object under arbitrary environment lighting and editing of the object's material properties. The key to our approach, which we call Neural Radiance Factorization (NeRFactor), is to distill the volumetric geometry of a Neural Radiance Field (NeRF) [Mildenhall et al. 2020] representation of the object into a surface representation and then jointly refine the geometry while solving for the spatially-varying reflectance and the environment lighting. Specifically, NeRFactor recovers 3D neural fields of surface normals, light visibility, albedo, and Bidirectional Reflectance Distribution Functions (BRDFs) without any supervision, using only a re-rendering loss, simple smoothness priors, and a data-driven BRDF prior learned from real-world BRDF measurements. By explicitly modeling light visibility, NeRFactor is able to separate shadows from albedo and synthesize realistic soft or hard shadows under arbitrary lighting conditions. NeRFactor is able to recover convincing 3D models for free-viewpoint relighting in this challenging and underconstrained capture setup for both synthetic and real scenes. Qualitative and quantitative experiments show that NeRFactor outperforms classic and deep learning-based state of the art across various tasks. Our code and data are available at people.csail.mit.edu/xiuming/projects/nerfactor/.

Authors' addresses: Xiuming Zhang, xiuming@csail.mit.edu, MIT CSAIL; Pratul P. Srinivasan, Google Research; Boyang Deng, Google Research; Paul Debevec, Google Research; William T. Freeman, MIT CSAIL & Google Research; Jonathan T. Barron, Google Research.

1 INTRODUCTION

Recovering an object's geometry and material properties from captured images, such that it can be rendered from arbitrary viewpoints under novel lighting conditions, is a longstanding problem within computer vision and graphics. The difficulty of this problem stems from its fundamentally underconstrained nature, and prior work has typically addressed this either by using additional observations such as scanned geometry, known lighting conditions, or images of the object under multiple different lighting conditions, or by making restrictive assumptions such as assuming a single material for the entire object or ignoring self-shadowing. In this work, we demonstrate that it is possible to recover convincing relightable representations from images of an object captured under *one unknown* natural illumination condition, as shown in Figure 1. Our key insight is that we can first optimize a Neural Radiance Field (NeRF) [Mildenhall et al. 2020] from the input images to initialize our model's surface normals and light visibility, and then jointly optimize these initial estimates along with the spatially-varying reflectance and the lighting condition to best explain the observed images. The use of NeRF to produce a high-quality geometry estimation for initialization helps break the inherent ambiguities among shape, reflectance, and lighting, thereby allowing us to recover a full 3D model for convincing view synthesis and relighting using just a re-rendering loss, simple spatial smoothness priors for each of these components, and a novel data-driven Bidirectional Reflectance

Distribution Function (BRDF) prior. Because NeRFactor models light visibility explicitly and efficiently, it is capable of removing shadows from albedo estimation and synthesizing realistic soft or hard shadows under arbitrary novel lighting conditions.

Although the geometry estimated by NeRF is effective for view synthesis, it has two limitations that prevent it from being easily used for relighting. First, NeRF models shape as a volumetric field, and as such it is computationally expensive to compute shading and visibility at each point along a camera ray for a full hemisphere of lighting. Second, the geometry estimated by NeRF contains extraneous high-frequency content that, while unnoticeable in view synthesis results, introduces high-frequency artifacts into the surface normals and light visibility computed from NeRF's geometry. We address the first issue by using a "hard surface" approximation of the NeRF geometry, where we only perform shading calculations at a single point along each ray, corresponding to the expected termination depth of the volume. We address the second issue by representing the surface normal and light visibility at any 3D location on this surface as continuous functions parameterized by Multi-Layer Perceptrons (MLPs), and encourage these functions to be close to the values derived from the pre-trained NeRF and be spatially smooth. Thus, our model, which we call Neural Radiance Factorization (NeRFactor), factors the observed images into estimated environment lighting as well as a 3D surface representation of the object with surface normals, light visibility, albedo, and spatially-varying BRDFs. This enables us to render novel views of the object under arbitrary novel environment lighting.

In summary, our main technical contributions are:

- A method for factorizing images of an object under an unknown lighting condition into shape, reflectance, and illumination, thereby supporting free-viewpoint relighting with shadows and material editing;
- A strategy to distill NeRF-estimated volume density into surface geometry (with normals and light visibility) to use as an initialization when improving the geometry and recovering reflectance; and
- A novel data-driven BRDF prior based on training a latent code model on real measured BRDFs.

Input & output. The input to NeRFactor is a set of multi-view images of an object illuminated under *one unknown* environment lighting condition and the camera poses of these images. NeRFactor jointly estimates a *plausible* collection of surface normals, light visibility, albedo, spatially-varying BRDFs, and the environment lighting that together explain the observed views. We then use the recovered geometry and reflectance to synthesize images of the object from novel viewpoints under arbitrary lighting. Modeling visibility explicitly, NeRFactor is able to remove shadows from albedo and synthesize soft or hard shadows under arbitrary lighting.

Assumptions. NeRFactor considers objects to be composed of hard surfaces with a single intersection point per ray, so volumetric light transport effects such as scattering, transparency, and translucency are not modeled. In addition, we only model direct illumination to simplify computation. Finally, our reflectance models consider materials with achromatic specular reflectance (dielectrics), so we do not model metallic materials (though one can easily extend our model for metallic materials by additionally predicting a specular color for each surface point).

2 RELATED WORK

Inverse rendering [Marschner 1998; Sato et al. 1997; Yu et al. 1999; Ramamoorthi and Hanrahan 2001], the task of factoring the appearance of an object in observed images into the underlying geometry, material properties, and lighting conditions, is a longstanding problem in computer vision and graphics. Since the full general inverse rendering problem is well-known to be severely underconstrained, most prior approaches have addressed this problem by assuming no shadow, learning priors on shape, illumination, and reflectance, or requiring additional observations, such as scanned geometry, measured lighting conditions, or additional images of the object under multiple (known) lighting conditions.

Methods for single-image inverse rendering [Barron and Malik 2015; Li et al. 2020, 2018; Lichy et al. 2021; Sang and Chandraker 2020; Sengupta et al. 2019; Wei et al. 2020; Yu and Smith 2019] largely rely on strong priors on geometry, reflectance, and illumination learned from large datasets. Recent methods can effectively infer plausible settings of these factors from a single image, but do not recover full 3D representations that can be viewed from arbitrary viewpoints.

Most methods that recover factorized full 3D models for relighting and view synthesis rely on additional observations instead of strong priors. A common strategy is to use 3D geometry obtained from active scanning [Guo et al. 2019; Lensch et al. 2003; Park et al. 2020; Schmitt et al. 2020; Zhang et al. 2020], proxy models [Chen et al. 2020; Dong et al. 2014; Gao et al. 2020; Georgoulis et al. 2015; Sato et al. 2003], silhouette masks [Godard et al. 2015; Oxholm and Nishino 2014; Xia et al. 2016], or multi-view stereo (followed by surface reconstruction and meshing) [Goel et al. 2020; Laffont et al. 2012; Nam et al. 2018; Philip et al. 2019] as a starting point before recovering reflectance and refined geometry. In this work, we show that starting with geometry estimated using a state-of-the-art neural volumetric representation enables us to recover a fully-factorized 3D model just using images captured under one illumination, without requiring any additional observations. Crucially, using a neural volumetric representation to estimate the initial geometry enables us to recover factored models for objects that have proven to be challenging for traditional geometry estimation methods, including objects with highly reflective surfaces and detailed geometry.

A large body of work within the computer graphics community has focused on the specific subproblem of material acquisition, where the goal is to estimate BRDF properties from images of materials with known (typically planar) geometry. These methods have traditionally utilized a signal processing reconstruction strategy and used complex controlled camera and lighting setups to adequately sample the BRDF [Foo 2015; Matusik et al. 2003; Nielsen et al. 2015], and more recent methods have enabled material acquisition from more casual smartphone setups [Aittala et al. 2015; Hui et al. 2017]. However, this line of work generally requires the geometry to be simple and fully known, while we focus on a more general problem where our only observations are images of an object with complex shape and spatially-varying reflectance.

Our work builds upon a recent trend within the computer vision and graphics communities that replaces traditional shape representations such as polygon meshes or discretized voxel grids with Multi-Layer Perceptrons (MLPs) that represent geometry as parametric functions. These MLPs are optimized to approximate continuous 3D geometry by mapping from 3D coordinates to properties of an object or scene (such as volume density, occupancy, or signed distance) at that location. This strategy has been successful for recovering continuous 3D shape representations from 3D observations [Mescheder et al. 2019; Park et al. 2019; Tancik et al. 2020] as well as from images observed under fixed lighting [Mildenhall et al. 2020; Yariv et al. 2020]. The Neural Radiance Fields [Mildenhall et al. 2020], or NeRF, technique has been particularly successful for optimizing volumetric geometry and appearance from observed images for the purpose of rendering photorealistic novel views, and has inspired subsequent approaches that extend NeRF's neural representation to enable relighting [Bi et al. 2020; Boss et al. 2020; Srinivasan et al. 2021; Zhang et al. 2021]. Our method differs from those of Bi et al. [2020] and Srinivasan et al. [2021] in that we do not require images to be captured under multiple known lighting conditions, since we recover a relightable model just from images of an object under *one unknown* lighting condition. The methods of Boss et al. [2020] and Zhang et al. [2021] address the same casual capture setup as our work, but both crucially do not consider light visibility and are thus unable to simulate any lighting occlusion or shadowing effects. Our method leverages the high-quality geometry estimated by NeRF to model accurate high-frequency shadowing and lighting occlusion when recovering a representation for relighting and view synthesis.

3 METHOD

The input to NeRFactor is assumed to be only posed multi-view images of an object lit by *one unknown* illumination condition. NeRFactor represents the shape and spatially-varying reflectance of an object as a set of 3D fields, each parameterized by MLPs whose weights are optimized so as to "explain" the set of observed input images. After optimization, NeRFactor outputs the surface normal $\boldsymbol{n}$, light visibility in any direction $v(\boldsymbol{\omega}_\text{i})$, albedo $\boldsymbol{a}$, and reflectance $\boldsymbol{z}_\text{BRDF}$ that together explain the observed appearance at any 3D location $\boldsymbol{x}$ on the object's surface*. By recovering the object's geometry and reflectance, NeRFactor enables applications such as free-viewpoint relighting with shadows and material editing.

3.1 Shape

The input to our model is the same as what is used by NeRF [Mildenhall et al. 2020], so we can apply NeRF to our input images to compute initial geometry. NeRF optimizes a neural radiance field: an MLP that maps from any 3D spatial coordinate and 2D viewing direction to the volume density at that 3D location and color emitted by particles at that location along the 2D viewing direction. NeRFactor leverages NeRF's estimated geometry by "distilling" it into a continuous surface representation that we use to initialize NeRFactor's geometry. In particular, we use the optimized NeRF to compute the expected surface location along any camera ray, the surface normal at each point on the object's surface, and the visibility of light arriving from any direction at each point on the object's surface. This subsection describes how we derive these functions from the optimized NeRF, as well as how we re-parameterize them with MLPs so that they can be fine-tuned after this initialization step to improve the full re-rendering loss (Figure 3).

Surface points. Given a camera and a trained NeRF, we compute the location at which a ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ from that camera $\boldsymbol{o}$ along direction $\boldsymbol{d}$ is expected to terminate according to NeRF's optimized volume density σ:

$$\boldsymbol{x}_\text{surf} = \boldsymbol{o} + \left(\int_0^\infty T(t)\sigma\big(\boldsymbol{r}(t)\big) t \, dt \right) \boldsymbol{d}, \tag{1}$$

where $T(t) = \exp\left(-\int_0^t \sigma\big(\boldsymbol{r}(s)\big)\, ds\right)$ is the probability that the ray travels distance t without being blocked. Instead of maintaining a full volumetric representation, we fix the geometry to lie on this surface distilled from the optimized NeRF. This enables much more efficient relighting during both training and inference, because we can compute the outgoing radiance just at each camera ray's expected termination location instead of every point along each camera ray.

Surface normals. We compute analytic surface normals $\boldsymbol{n}_\text{a}(\boldsymbol{x})$ at any 3D location as the negative normalized gradient of NeRF's σ-volume w.r.t. $\boldsymbol{x}$. Unfortunately, the normals derived from a trained NeRF tend to be noisy (Figure 3) and therefore produce "bumpy" artifacts when used for rendering (see the supplemental video). Therefore, we re-parameterize these normals using an MLP $\boldsymbol{f}_\text{n}$, which maps from any location $\boldsymbol{x}_\text{surf}$ on the surface to a "denoised" surface normal $\boldsymbol{n}$: $\boldsymbol{f}_\text{n} : \boldsymbol{x}_\text{surf} \mapsto \boldsymbol{n}$. During the joint optimization of NeRFactor's weights, we encourage the output of this MLP (1) to stay close to the normals produced from the pretrained NeRF, (2) to vary smoothly in the 3D space, and (3) to reproduce the observed appearance of the object. Specifically, the loss function reflecting (1) and (2) is:

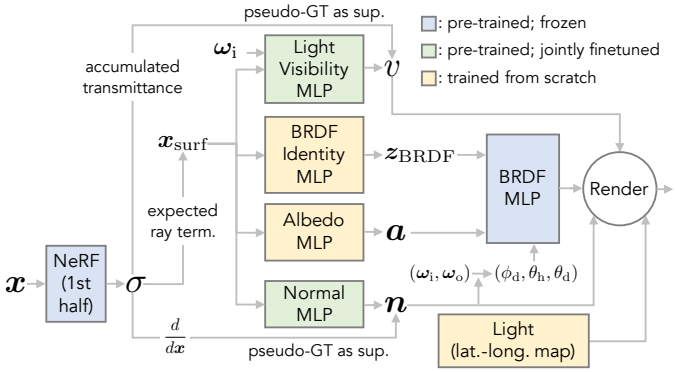
$$\ell_\text{n} = \sum_{\boldsymbol{x}_\text{surf}} \left(\frac{\lambda_1}{3} \left\| \boldsymbol{f}_\text{n}(\boldsymbol{x}_\text{surf}) - \boldsymbol{n}_\text{a}(\boldsymbol{x}_\text{surf}) \right\|_2^2 \right. \tag{2}$$

$$\left. + \frac{\lambda_2}{3} \left\| \boldsymbol{f}_\text{n}(\boldsymbol{x}_\text{surf}) - \boldsymbol{f}_\text{n}(\boldsymbol{x}_\text{surf} + \boldsymbol{\epsilon}) \right\|_1 \right), \tag{3}$$

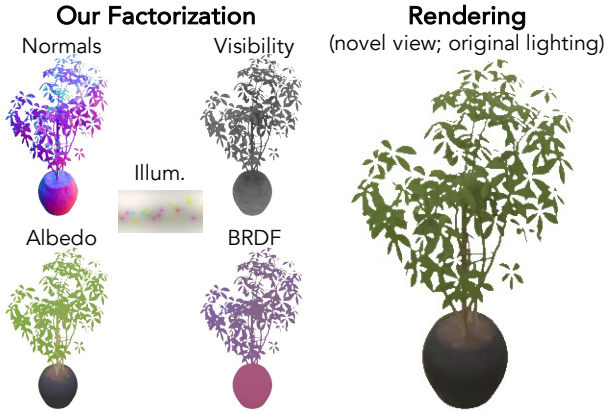
where $\boldsymbol{\epsilon}$ is a random 3D displacement from $\boldsymbol{x}_\text{surf}$ sampled from a zero-mean Gaussian with standard deviation 0.01 (0.001 for the real scenes due to the different scene scales), and the λ_1 and λ_2 are hyperparameters set to 0.1 and 0.05, respectively. A similar smoothness loss on surface normals is used in the concurrent work by Oechsle et al. [2021] for the goal of shape reconstruction. Crucially, not restricting $\boldsymbol{x}$ on the expected surface increases the robustness of the MLP by providing a "safe margin" where the output remains well-behaved even when the input is slightly displaced from the surface. As shown in Figure 3, NeRFactor's normal MLP produces normals that are significantly higher-quality than those produced by NeRF, and are smooth enough to be used for relighting (Figure 5).

Light visibility. We compute the visibility v_a to each light source from any point by marching through NeRF's σ-volume from the point to each light location, as in Bi et al. [2020]. However, similarly to the surface normal estimation described above, the visibility

*In this paper, vectors and matrices (as well as functions that return them) are in bold; scalars and scalar functions are not.



(a) **Model.** NeRFactor leverages NeRF's σ-volume as an initialization to predict surface normal $\boldsymbol{n}$, light visibility v, albedo $\boldsymbol{a}$, and BRDF latent code $\boldsymbol{z}_{\text{BRDF}}$ for each surface location $\boldsymbol{x}_{\text{surf}}$, as well as the lighting condition. $\boldsymbol{x}$ denotes 3D locations, $\boldsymbol{\omega}_{\text{i}}$ light direction, $\boldsymbol{\omega}_{\text{o}}$ viewing direction, and ϕ_{d}, θ_{h}, θ_{d} Rusinkiewicz coordinates. NeRFactor does not require supervision on any of the intermediate factors, but rather relies only on priors and a reconstruction loss. Note that NeRFactor is an all-MLP architecture that models only surface points (unlike NeRF that models the entire volume).



(b) **Example factorization.** NeRFactor jointly solves for *plausible* 3D fields of surface normals, light visibility, albedo, and BRDFs (as well as the environment lighting) that together explain the observed views. Here we visualize the average of the light visibility over all incoming directions at each location (i.e., the ambient occlusion). We visualize z_{BRDF} as an RGB image (similar colors indicate similar materials).

Fig. 2. NeRFactor is a coordinate-based MLP-only model that factorizes, in an unsupervised manner, the appearance of a scene observed under one unknown lighting condition. It tackles this severely ill-posed problem by using a reconstruction loss, simple smoothness regularization, and data-driven BRDF priors. Modeling visibility explicitly, NeRFactor is a physically-based model that supports hard and soft shadows under arbitrary lighting.

estimates derived directly from NeRF's σ-volume are too noisy to be used directly, and result in rendering artifacts (see the supplemental video). We address this by re-parameterizing the visibility function as another MLP that maps from a surface location $\boldsymbol{x}_{\text{surf}}$ and a light direction $\boldsymbol{\omega}_{\text{i}}$ to the light visibility v: $f_{\text{v}} : (\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}) \mapsto v$. We optimize the weights of f_{v} to encourage the recovered visibility field (1) to be close to the visibility traced from the NeRF, (2) to be spatially smooth, and (3) to reproduce the observed appearance. Specifically, the loss function implementing (1) and (2) is:

$$\ell_{\text{v}} = \sum_{\boldsymbol{x}_{\text{surf}}} \sum_{\boldsymbol{\omega}_{\text{i}}} \Big(\lambda_3 \big(f_{\text{v}}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}) - v_{\text{a}}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}) \big)^2 \tag{4}$$

$$+ \lambda_4 \big| f_{\text{v}}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}) - f_{\text{v}}(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}, \boldsymbol{\omega}_{\text{i}}) \big| \Big), \tag{5}$$

where $\boldsymbol{\epsilon}$ is the random displacement defined above, and λ_3 and λ_4 are hyperparameters set to 0.1 and 0.05, respectively. As the equation shows, smoothness is encouraged across spatial locations given the same $\boldsymbol{\omega}_{\text{i}}$, not the other way around. This is by design, to avoid the visibility at a certain location getting blurred over different light locations. Note that this is similar to the visibility fields in Srinivasan et al. [2021], but in our case we optimize the visibility MLP parameters to denoise the visibility derived from a pretrained NeRF and minimize the re-rendering loss. For computing the NeRF visibility, we use a fixed set of 512 light locations given a predefined illumination resolution (to be discussed later). After optimization, f_{v} produces spatially smooth and realistic estimates of light visibility, as can be seen in Figure 3 (II) and Figure 4 (C) where we visualize the average visibility over all light directions (i.e., ambient occlusion).

In practice, before the full optimization of our model, we independently pretrain the visibility and normal MLPs to just reproduce the visibility and normal values from the NeRF σ-volume without any smoothness regularization or re-rendering loss. This provides a reasonable initialization of the visibility maps, which prevents the albedo or BRDF MLP from mistakenly attempting to explain away shadows as being modeled as "painted on" reflectance variation (see "w/o geom. pretrain." in Figure 9 and Table 1).

3.2 Reflectance

Our full BRDF model $\boldsymbol{R}$ consists of a diffuse component (Lambertian) fully determined by albedo $\boldsymbol{a}$ and a specular spatially-varying BRDF $\boldsymbol{f}_{\text{r}}$ (defined for any location on the surface $\boldsymbol{x}_{\text{surf}}$ with incoming light direction $\boldsymbol{\omega}_{\text{i}}$ and outgoing direction $\boldsymbol{\omega}_{\text{o}}$) learned from real-world reflectance:

$$\boldsymbol{R}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) = \frac{\boldsymbol{a}(\boldsymbol{x}_{\text{surf}})}{\pi} + \boldsymbol{f}_{\text{r}}\left(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}\right). \tag{6}$$

Prior art in neural rendering has explored the use of parameterizing $\boldsymbol{f}_{\text{r}}$ with analytic BRDFs, such as microfacet models [Bi et al. 2020; Srinivasan et al. 2021], within a NeRF-like setting. Although these analytic models provide an effective BRDF parameterization for optimization to explore, no prior is imposed upon the parameters themselves: all materials that are expressible within a microfacet model are considered equally likely a priori. Additionally, the use of an explicit analytic model limits the set of materials that can be recovered, and this is insufficient for modelling all real-world reflectance functions.

Instead of assuming an analytic BRDF, NeRFactor starts with a learned reflectance function that is pretrained to reproduce a wide range of empirically observed real-world reflectance functions,

while also learning a latent space for those real-world reflectance functions. By doing so, we learn data-driven priors on real-world BRDFs that encourage the optimization procedure to recover *plausible* reflectance functions. The use of such priors is crucial: because all of our observed images are taken under one (unknown) illumination, our problem is highly ill-posed, so priors are necessary to disambiguate the most likely factorization of the scene from the set of all possible factorizations.

Albedo. We parameterize the albedo $\boldsymbol{a}$ at any location on the surface $\boldsymbol{x}_{\text{surf}}$ as an MLP $\boldsymbol{f}_{\text{a}} : \boldsymbol{x}_{\text{surf}} \mapsto \boldsymbol{a}$. Because there is no direct supervision on albedo, and our model is only able to observe one illumination condition, we rely on simple spatial smoothness priors (and light visibility) to disambiguate between, for example, the "white-painted surface containing a shadow" case and the "black-and-white-painted surface" case. In addition, the reconstruction loss of the observed views also drives the optimization of $\boldsymbol{f}_{\text{a}}$. The loss function that reflects this smoothness prior is:

$$\ell_{\text{a}} = \lambda_5 \sum_{\boldsymbol{x}_{\text{surf}}} \frac{1}{3} \left\| \boldsymbol{f}_{\text{a}}(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{f}_{\text{a}}(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}) \right\|_1, \tag{7}$$

where $\boldsymbol{\epsilon}$ is the same random 3D perturbation as defined above, and λ_5 is a hyperparameter set to 0.05. The output from $\boldsymbol{f}_{\text{a}}$ is used as albedo in the Lambertian reflectance, but not in the non-diffuse component, for which we assume the specular highlight color to be white. We empirically constrain the albedo prediction to $[0.03, 0.8]$ following Ward and Shakespeare [1998], by scaling the network's final sigmoid output by 0.77 and then adding a bias of 0.03.

Learning priors from real-world BRDFs. For the specular components of the BRDF, we seek to learn a latent space of real-world BRDFs and a paired "decoder" that translates each latent code in the learned space $\boldsymbol{z}_{\text{BRDF}}$ to a full 4D BRDF. To this end, we adopt the Generative Latent Optimization (GLO) approach [Bojanowski et al. 2018], which has been previously used by other coordinate-based models such as Park et al. [2019] and Martin-Brualla et al. [2021]. The $\boldsymbol{f}_{\text{r}}$ component of our model is pretrained using the the MERL dataset [Matusik et al. 2003]. Because the MERL dataset assumes isotropic materials, we parameterize the incoming and outgoing directions for $\boldsymbol{f}_{\text{r}}$ using Rusinkiewicz coordinates [Rusinkiewicz 1998] $(\phi_{\text{d}}, \theta_{\text{h}}, \theta_{\text{d}})$ (3 degrees of freedom) instead of $\boldsymbol{\omega}_{\text{i}}$ and $\boldsymbol{\omega}_{\text{o}}$ (4 degrees of freedom). Denote this coordinate conversion by $\boldsymbol{g} : (\boldsymbol{n}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) \mapsto (\phi_{\text{d}}, \theta_{\text{h}}, \theta_{\text{d}})$, where $\boldsymbol{n}$ is the surface normal at that point. We train a function $\boldsymbol{f}'_{\text{r}}$ (a re-parameterization of $\boldsymbol{f}_{\text{r}}$) that maps from a concatenation of a latent code $\boldsymbol{z}_{\text{BRDF}}$ (which represents a BRDF identity) and a set of Rusinkiewicz coordinates $(\phi_{\text{d}}, \theta_{\text{h}}, \theta_{\text{d}})$ to an achromatic reflectance $\boldsymbol{r}$:

$$\boldsymbol{f}'_{\text{r}} : \left(\boldsymbol{z}_{\text{BRDF}}, (\phi_{\text{d}}, \theta_{\text{h}}, \theta_{\text{d}})\right) \mapsto \boldsymbol{r}. \tag{8}$$

To train this model, we optimize both the weights of the MLP as well as the set of latent codes $\boldsymbol{z}_{\text{BRDF}}$ to reproduce a set of real-world BRDFs. Simple mean squared errors are computed on the log of the High-Dynamic-Range (HDR) reflectance values to train $\boldsymbol{f}'_{\text{r}}$.

Because the color component of our reflectance model is assumed to be handled by the albedo prediction network, we discard all color information from the MERL dataset by converting its RGB reflectance values into achromatic ones[†]. The latent BRDF identity codes $\boldsymbol{z}_{\text{BRDF}}$ are parameterized as unconstrained 3D vectors, and are initialized with a zero-mean isotropic Gaussian with a standard deviation of 0.01. No sparsity or norm penalty is imposed on $\boldsymbol{z}_{\text{BRDF}}$ during training. After this pretraining, the weights of this BRDF MLP are frozen during the joint optimization of our entire model, and we predict only $\boldsymbol{z}_{\text{BRDF}}$ for each $\boldsymbol{x}_{\text{surf}}$ by training from scratch a BRDF identity MLP (Figure 2a): $\boldsymbol{f}_{\text{z}} : \boldsymbol{x}_{\text{surf}} \mapsto \boldsymbol{z}_{\text{BRDF}}$. This can be thought of as predicting spatially-varying BRDFs for all the surface points in the plausible space of real-world BRDFs. We optimize the BRDF identity MLP to minimize the re-rendering loss as well as the same spatial smoothness prior as in albedo:

$$\ell_{\text{z}} = \lambda_6 \sum_{\boldsymbol{x}_{\text{surf}}} \frac{\left\| \boldsymbol{f}_{\text{z}}(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{f}_{\text{z}}(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}) \right\|_1}{\dim(\boldsymbol{z}_{\text{BRDF}})}, \tag{9}$$

where λ_6 is a hyperparameter set to 0.01, and $\dim(\boldsymbol{z}_{\text{BRDF}})$ denotes the dimensionality of the BRDF latent code (3 in our implementation because there are only 100 materials in the MERL dataset). The final BRDF is the sum of the Lambertian component and the learned non-diffuse reflectance (subscript of $\boldsymbol{x}_{\text{surf}}$ dropped for brevity):

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) = \frac{\boldsymbol{f}_{\text{a}}(\boldsymbol{x})}{\pi} + \boldsymbol{f}'_{\text{r}}\Big(\boldsymbol{f}_{\text{z}}(\boldsymbol{x}), \boldsymbol{g}\big(\boldsymbol{f}_{\text{n}}(\boldsymbol{x}), \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}\big)\Big), \tag{10}$$

where the specular highlight color is assumed to be white.

3.3 Illumination

We adopt a simple and direct representation of lighting: an HDR light probe image [Debevec 1998] in the latitude-longitude format. In contrast to spherical harmonics or a mixture of spherical Gaussians, this representation allows our model to represent detailed high-frequency lighting and therefore to support hard cast shadows. That said, the challenges of using this representation are clear: it contains a large number of parameters, and every pixel/parameter can vary independently of all other pixels/parameters. This issue can be ameliorated by our use of the light visibility MLP, which allows us to quickly evaluate a surface point's visibility to all pixels of the light probe. Empirically, we use a 16×32 resolution for our lighting environments, as we do not expect to recover higher-frequency content in the light probe image beyond that resolution (the environment is effectively low-pass filtered by each object's BRDFs as discussed by Ramamoorthi and Hanrahan [2004], and the objects in our datasets are not mirror-like).

To encourage smoother lighting, we apply a simple ℓ^2 gradient penalty on the pixels of the light probe $\boldsymbol{L}$ along both the horizontal and vertical directions:

$$\ell_{\text{i}} = \lambda_7 \left(\left\| \begin{bmatrix} -1 & 1 \end{bmatrix} * \boldsymbol{L} \right\|_2^2 + \left\| \begin{bmatrix} -1 \\ 1 \end{bmatrix} * \boldsymbol{L} \right\|_2^2 \right), \tag{11}$$

where $*$ denotes the convolution operator, and λ_7 is a hyperparameter set to 5×10^{-6} (given that there are 512 pixels with HDR values). During the joint optimization, these probe pixels get updated directly by the final reconstruction loss and the gradient penalty.

[†]In principle, one should be able to perform diffuse-specular separation on the MERL BRDFs and then learn priors on just the specular lobes. We experimented with this idea by using the separation provided by Sun et al. [2018], but this yielded qualitatively worse results.

3.4 Rendering

Given the surface normal, visibility of all light directions, albedo, and BRDF at each point $\boldsymbol{x}_{\text{surf}}$, as well as the estimated lighting environment, the final physically-based, non-learnable renderer renders an image that is then compared against the observed image. The errors in this rendered image are backpropagated up to, but excluding, the σ-volume of the pretrained NeRF, thereby driving the joint estimation of surface normals, light visibility, albedo, BRDFs, and illumination.

Given the ill-posed nature of the problem (largely due to our only observing one unknown illumination), we expect the majority of useful information to be from direct illumination rather than global illumination, and therefore consider only single-bounce direct illumination (i.e., from the light source to the object surface, and then to the camera). This assumption also reduces the computational cost of evaluating our model. Mathematically, the rendering equation in our setup is (subscript of $\boldsymbol{x}_{\text{surf}}$ dropped again for brevity):

$$\boldsymbol{L}_{\text{o}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}}) = \int_{\Omega} \boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) \boldsymbol{L}_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}) \big(\boldsymbol{\omega}_{\text{i}} \cdot \boldsymbol{n}(\boldsymbol{x})\big) d\boldsymbol{\omega}_{\text{i}} \quad (12)$$

$$= \sum_{\boldsymbol{\omega}_{\text{i}}} \boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) \boldsymbol{L}_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}) \big(\boldsymbol{\omega}_{\text{i}} \cdot \boldsymbol{f}_{\text{n}}(\boldsymbol{x})\big) \Delta\boldsymbol{\omega}_{\text{i}} = \sum_{\boldsymbol{\omega}_{\text{i}}} \Bigg(\frac{\boldsymbol{f}_{\text{a}}(\boldsymbol{x})}{\pi} + \quad (13)$$

$$\boldsymbol{f}'_{\text{r}}\Big(\boldsymbol{f}_{\text{z}}(\boldsymbol{x}), \boldsymbol{g}\big(\boldsymbol{f}_{\text{n}}(\boldsymbol{x}), \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}\big)\Big)\Bigg) \boldsymbol{L}_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}}) \big(\boldsymbol{\omega}_{\text{i}} \cdot \boldsymbol{f}_{\text{n}}(\boldsymbol{x})\big) \Delta\boldsymbol{\omega}_{\text{i}}, \quad (14)$$

where $\boldsymbol{L}_{\text{o}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}})$ is the outgoing radiance at $\boldsymbol{x}$ as viewed from $\boldsymbol{\omega}_{\text{o}}$, $\boldsymbol{L}_{\text{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}})$ is the incoming radiance, masked by the visibility $\boldsymbol{f}_{\text{v}}(\boldsymbol{x}, \boldsymbol{\omega}_{\text{i}})$, arriving at $\boldsymbol{x}$ along $\boldsymbol{\omega}_{\text{i}}$ directly from a light probe pixel (since we consider only single-bounce direct illumination), and $\Delta\boldsymbol{\omega}_{\text{i}}$ is the solid angle corresponding to the lighting sample at $\boldsymbol{\omega}_{\text{i}}$.

The final reconstruction loss ℓ_{recon} is simply the mean squared error (with a unit weight) between the rendering and the observed image. Therefore, our full loss function is the summation of all the previously defined losses: $\ell_{\text{recon}} + \ell_{\text{n}} + \ell_{\text{v}} + \ell_{\text{a}} + \ell_{\text{z}} + \ell_{\text{i}}$.

3.5 Implementation Details

NeRFactor is implemented in TensorFlow 2 [Abadi et al. 2015]. All training uses the Adam optimizer [Kingma and Ba 2015] with the default hyperparameters.

Staged training. There are three stages in training NeRFactor. First, we optimize a NeRF using the input posed images (once per scene) and train a BRDF MLP on the MERL dataset (only once for all scenes). Both of these MLPs are frozen during the final joint optimization, since the NeRF only provides a shape initialization, and the BRDF MLP provides a latent space of real-world BRDFs for the optimization to explore. Future shape refinement happens in NeRFactor's normal and visibility MLPs, and the actual material prediction happens in NeRFactor's albedo and BRDF identity MLPs. Second, we use this trained NeRF to initialize our geometry by optimizing the normal and visibility MLPs to simply reproduce the NeRF values, without any additional smoothness loss or regularization. Finally, we jointly optimize the albedo MLP, BRDF identity MLP, and light probe pixels from scratch, along with the pretrained normal and visibility MLPs. Fine-tuning the normal and visibility MLPs along with the reflectance and lighting allows the errors in NeRF's initial geometry to be improved in order to minimize the re-rendering loss (Figure 3).

Architecture and positional encoding. We use the default architecture for NeRF [Mildenhall et al. 2020], and all other MLPs that we introduce contain four layers (with a skip connection from the input to the second layer), each with 128 hidden units. As in NeRF [Mildenhall et al. 2020], we apply positional encoding to the input coordinates of all networks with 10 encoding levels for 3D locations and 4 encoding levels for directions.

Runtime. We train NeRF for 2,000 epochs, which takes 6-8 hours when distributed over four NVIDIA TITAN RTX GPUs. Prior to the final joint optimization, computing the initial surface normals and light visibility from the trained NeRF takes 30 minutes per view on a single GPU for a 16×32 light probe (i.e., 512 light source locations). This step can be trivially parallelized because each view is processed independently. Geometry pretraining is performed for 200 epochs, which takes around 20 minutes on one TITAN RTX. The final joint optimization is performed for 100 epochs, which takes only 30 minutes on one TITAN RTX.

4 RESULTS & APPLICATIONS

In this section, we explain how the datasets are constructed for our tasks, show our results for joint shape, reflectance, and illumination estimation, and finally showcase the enabled applications including free-viewpoint relighting with a single point light or any arbitrary light probe (Figure 5 and Figure 6) and material editing (Figure 7).

4.1 Data

This work uses three sources of data: posed multi-view images of an object, real-world measured BRDFs, and captured light probes.

Synthetic renderings. We use the synthetic Blender scenes (`hotdog`, `drums`, `lego`, and `ficus`) released by Mildenhall et al. [2020], and replace the illumination used there with our own natural illuminations taken from real light probe images (we use publicly available light probes from hdrihaven.com, Stumpfel et al. [2004], and Blender). This yields significantly more natural input illumination conditions. We also disable all non-standard post-rendering effects used by Blender Cycles when rendering the images, such as "filmic" tonemapping, and retain only the standard linear-to-sRGB nonlinear tonemapping. We render all images directly to PNGs instead of EXRs to simulate real-world mobile phone captures where raw HDR pixel intensities may not be available; this indeed facilitates applying NeRFactor directly to real scenes as shown in Figure 6.

Real captures. We use mobile phone captures of two real scenes released by Mildenhall et al. [2020]: `vasedeck` and `pinecone`. These scenes are captured by inwards-facing cameras on the upper hemisphere. There are close to 100 images per scene, and the camera poses are obtained by COLMAP Structure from Motion (SfM) [Schönberger and Frahm 2016]. NeRFactor is directly applicable because it is designed to work with PNGs instead of EXRs.

Measured BRDFs. We use real measured BRDFs from the MERL dataset by Matusik et al. [2003]. The MERL dataset consists of 100

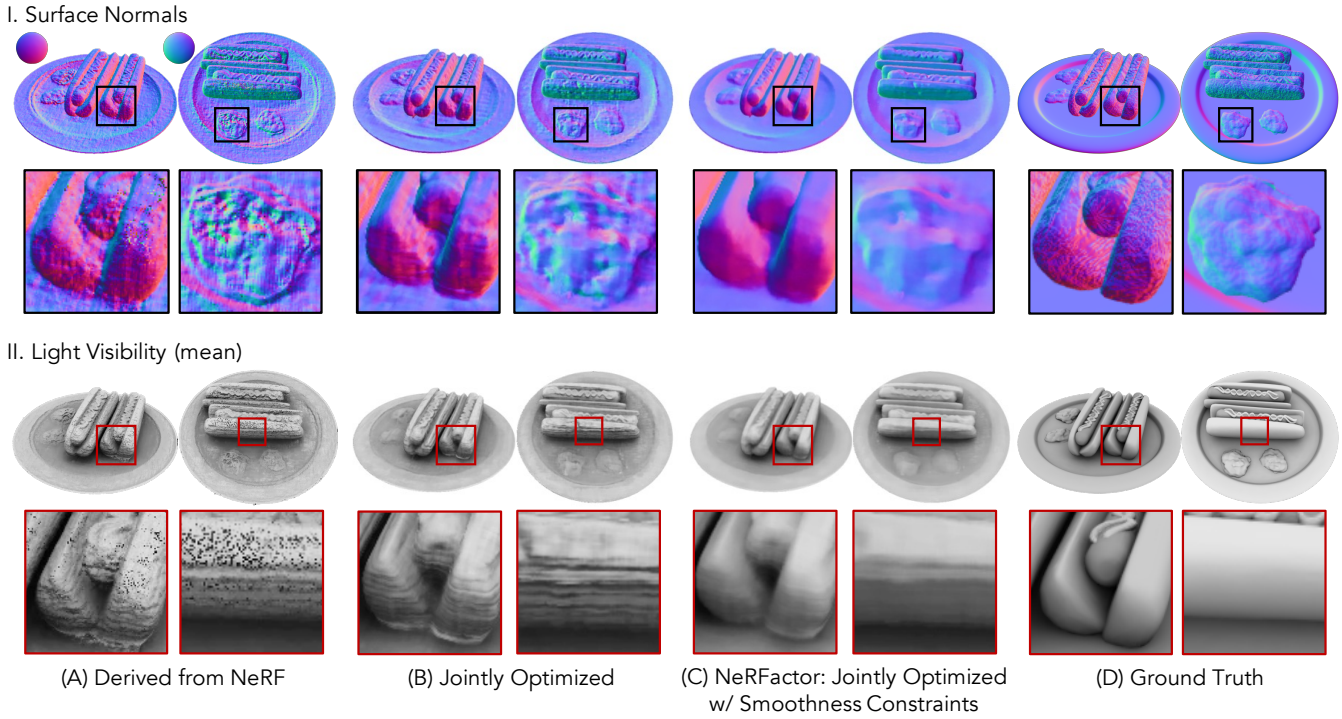


Fig. 3. **High-quality geometry recovered by NeRFactor.** (A) We directly derive the surface normals and light visibility from a trained NeRF: differentiating NeRF's σ-volume with respect to 3D location $\boldsymbol{x}$ gives surface normals, and marching through the same σ-volume to each light location gives visibility. However, geometry derived in this way directly from NeRF is too noisy to be used for relighting (see the supplemental video). (B) With the NeRF geometry as the starting point, jointly optimizing shape and reflectance improves the geometry, but there is still significant noise (e.g., the stripe artifacts in [II]). (C) Joint optimization with smoothness constraints leads to smooth surface normals and light visibility that resemble ground truth. We visualize the visibility maps as ambient occlusion maps, by taking averages over all incoming light directions.

real-world BRDFs measured by a conventional gonioreflectometer. Because the color components of BRDFs are not used by our model, we convert the RGB reflectance values to be achromatic by converting linear RGB values to relative luminance.

4.2 Shape Optimization

NeRFactor jointly estimates an object's shape in the form of surface points and their associated surface normals as well as their visibility to each light location. Figure 3 visualizes these geometric properties. To visualize light visibility, we take the mean of the 512 visibility maps corresponding to each pixel of a 16×32 light probe, and visualize that average map (i.e., ambient occlusion) as a grayscale image. See the supplemental video for movies of per-light visibility maps (i.e., shadow maps). As Figure 3 shows, our surface normals and light visibility are smooth and resemble the ground-truth normals, thanks to the joint estimation procedure that optimizes normals and visibility to minimize re-rendering errors and encourage spatial smoothness.

If we ablate the spatial smoothness constraints and rely on only the re-rendering loss, we end up with noisy geometry that is insufficient for rendering. Although these geometry-induced artifacts may not show up under low-frequency lighting, harsh lighting conditions (such as a single point light with no ambient illumination, i.e., One-Light-at-A-Time [OLAT]) reveal them as demonstrated in our supplemental video. Perhaps surprisingly, even when our smoothness constraints are disabled, the geometry estimated by NeRFactor is still significantly less noisy than the original NeRF geometry (compare [A] with [B] of Figure 3 and see [I] of Table 1) because the re-rendering loss encourages smoother geometry. See Section 5.2 for more details.

4.3 Joint Estimation of Shape, Reflectance, & Illumination

In this experiment, we demonstrate how NeRFactor factorizes appearance into shape, reflectance, and illumination for scenes with complex geometry and/or reflectance.

When visualizing albedo, we adopt the convention used by the intrinsic image literature of assuming that the absolute brightness of albedo and shading is unrecoverable [Land and McCann 1971], and furthermore we assume that the problem of color constancy (solving for a global color correction that disambiguates between the average color of the illuminant and the average color of the albedo [Buchsbaum 1980]) is also out of scope. In accordance with these two assumptions, we visualize our predicted albedo and measure its accuracy by first scaling each albedo channel by a global scalar that is identified so as to minimize squared error with respect to the ground-truth albedo, as is done in Barron and Malik [2015].

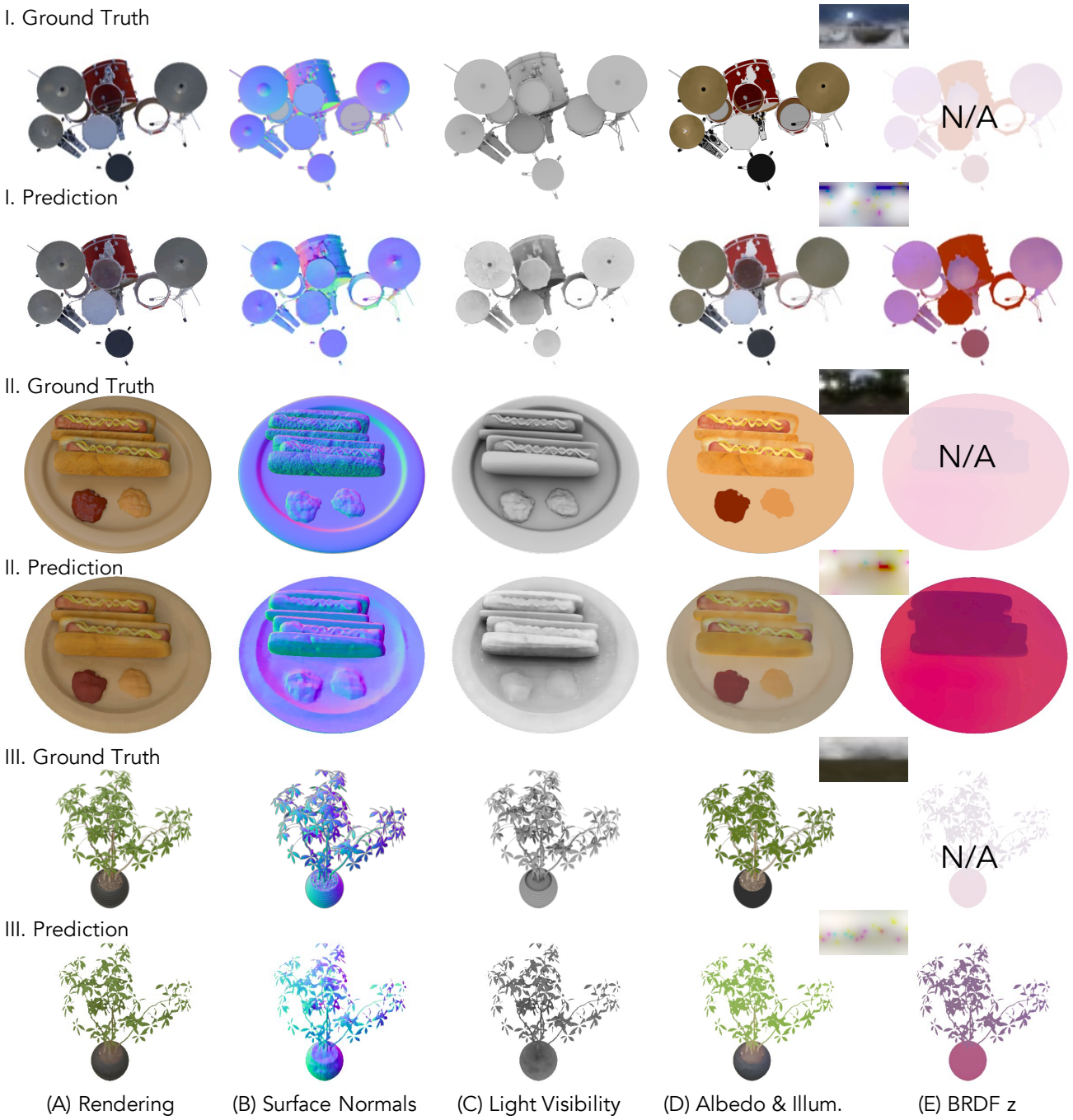


Fig. 4. **Joint optimization of shape, reflectance, and illumination.** Here we visualize factorizations produced by NeRFactor (bottom) alongside the ground truth (top) on three scenes. Although our recovered surface normals, visibility, and albedo sometimes omit some fine-grained detail, they still closely resemble the ground truth. Despite that the illuminations recovered by NeRFactor are heavily oversmoothed (due to the effective low-pass filtering induced by observing illumination only after it has been convolved by the object's BRDFs) and incorrect on the bottom half of the hemisphere (since objects are only ever observed from the top hemisphere), the dominant light sources and occluders are localized nearby their ground-truth locations in the light probes. Note that we are unable to compare against ground-truth BRDFs, as they are defined using Blender's shader node trees, while our recovered BRDFs are parameterized by our learned model.

Unless stated otherwise, all albedo predictions in this paper are corrected this way, and we apply gamma correction ($\gamma = 2.2$) to display them properly in the figures. Our estimated light probes are not scaled this way with respect to the ground truth (since illumination estimation is not the primary goal of this work) and visualized by simply scaling their maximum intensity to 1 and applying gamma correction ($\gamma = 2.2$) to show details in the dark regions.

As shown in Figure 4 (B), NeRFactor predicts high-quality and smooth surface normals that are close to the ground truth except in regions with very high-frequency details such as the bumpy surface of the `hotdog` buns. In the `drums` scene, we see that NeRFactor is able to successfully reconstruct fine details such as the screw at the center of the cymbal, and the metal rims on the sides of the drums. For `ficus`, NeRFactor is able to recover the complex leaf geometry of the potted plant. The average light visibility (ambient occlusion) maps also correctly portray the average exposure of each point in the scene to the lights. Albedo is recovered cleanly, with barely any shadowing or shading detail inaccurately attributed to albedo variation; note how the shading on the drums is absent in the albedo prediction. Moreover, the predicted light probes correctly reflect the locations of the primary light sources and the blue sky (blue pixels in [I]). In all three scenes, the predicted BRDFs are spatially-varying and correctly reflect that different parts of the scene have different materials, as indicated by different BRDF latent codes in (E).

Instead of representing illumination with a more sophisticated representation such as spherical harmonics, we opt for a straightforward representation: a latitude-longitude map whose pixels are HDR intensities. Because lighting is effectively convolved by a low-pass filter when reflected by a moderately diffuse BRDF [Ramamoorthi and Hanrahan 2004], we do not expect to recover illumination at a resolution higher than 16×32. As shown in Figure 4 (I), NeRFactor estimates a light probe that correctly captures the bright light source on the far left as well as the blue sky. Similarly, in Figure 4 (II), the dominant light source location is also correctly estimated (the bright white blob on the left).

4.4 Free-Viewpoint Relighting

NeRFactor estimates 3D fields of shape and reflectance, thus enabling simultaneous relighting and view synthesis. As such, all the relighting results shown in this paper and the supplemental video are rendered from novel viewpoints. To probe the limits of NeRFactor, we use harsh test illumination conditions that have one point light on at a time (OLAT), with no ambient illumination. These test illuminations induce hard cast shadows, which effectively expose rendering artifacts due to inaccurate geometry and materials. For visualization purposes, we composite the relit results (using NeRF's predicted opacity) onto backgrounds whose colors are the averages over upper halves of the light probes.

As shown in Figure 5 (II), NeRFactor synthesizes correct hard shadows cast by the hotdogs under the three test OLAT conditions. NeRFactor also produces realistic renderings of the ficus under the OLAT illuminations (I), especially when the ficus is back-lit by the point light in (D). Note that the ground truth in (D) appears brighter than NeRFactor's results, because NeRFactor models only direct illumination, whereas the ground-truth image was rendered with global illumination. When we relight the objects with two new light probes, realistic soft shadows are synthesized on the plate of `hotdog` (II). In `ficus`, the specularities on the vase correctly reflect the primary light sources in the both test probes. The ficus leaves also exhibit realistic specular highlights close to the ground truth in (F). In `drums` (III), the cymbals are correctly estimated to be specular and exhibit realistic reflection, though different from the ground-truth anisotropic reflection (D). This is as expected because all MERL BRDFs are isotropic [Matusik et al. 2003]. Despite being unable to explain these anisotropic reflections, NeRFactor correctly leaves them out in albedo rather than interprets them as albedo paints, since doing that would violate the albedo smoothness constraint and contradict those reflections' view dependency. In `lego`, realistic hard shadows are synthesized by NeRFactor for the OLAT test conditions (IV).

Relighting real scenes. We apply NeRFactor to the two real scenes, `vasedeck` and `pinecone`, captured by Mildenhall et al. [2020]. These captures are particularly suitable for NeRFactor: there are around 100 multi-view images of each scene lit by an unknown environment lighting. As in NeRF, we run COLMAP SfM [Schönberger and Frahm 2016] to obtain the camera intrinsics and extrinsics for each view. We then train a vanilla NeRF to obtain an initial shape estimate, which we distill into NeRFactor and jointly optimize together with reflectance and illumination. As Figure 6 (I) shows, the appearance is factorized into illumination (not pictured) and 3D fields of surface normals, light visibility, albedo, and spatially-varying BRDF latent codes that together explain the observed views. With such factorization, we relight the scenes by replacing the estimated illumination with novel arbitrary light probes (Figure 6 [II]). Because our factorization is fully 3D, all the intermediate buffers can be rendered from any viewpoint, and the relighting results shown are also from novel viewpoints. Note that the scenes are bounded to avoid faraway geometry blocking light from certain directions and casting shadows during relighting.

4.5 Material Editing

Since NeRFactor factorizes diffuse albedo and specular BRDF from appearance, one can edit the albedo, non-diffuse BRDF, or both and re-render the edited object under an arbitrary lighting condition from any viewpoint. In this subsection, we override the estimated z_{BRDF} to the learned latent code of `pearl-paint` in the MERL dataset, and the estimated albedo to colors linearly interpolated from the `turbo` colormap, spatially varying based on the surface points' x-coordinates. As Figure 7 (left) demonstrates, with the factorization by NeRFactor, we are able to realistically relight the original estimated materials with the two challenging OLAT conditions. Furthermore, the edited materials are also relit with realistic specular highlights and hard shadows by the same test OLAT conditions (Figure 7 [right]).

5 EVALUATION STUDIES

In this section, we study whether albedo estimation for the same object remains consistent across different input lighting conditions, perform ablation studies to evaluate the importance of each model component, and compare NeRFactor against both classic and deep

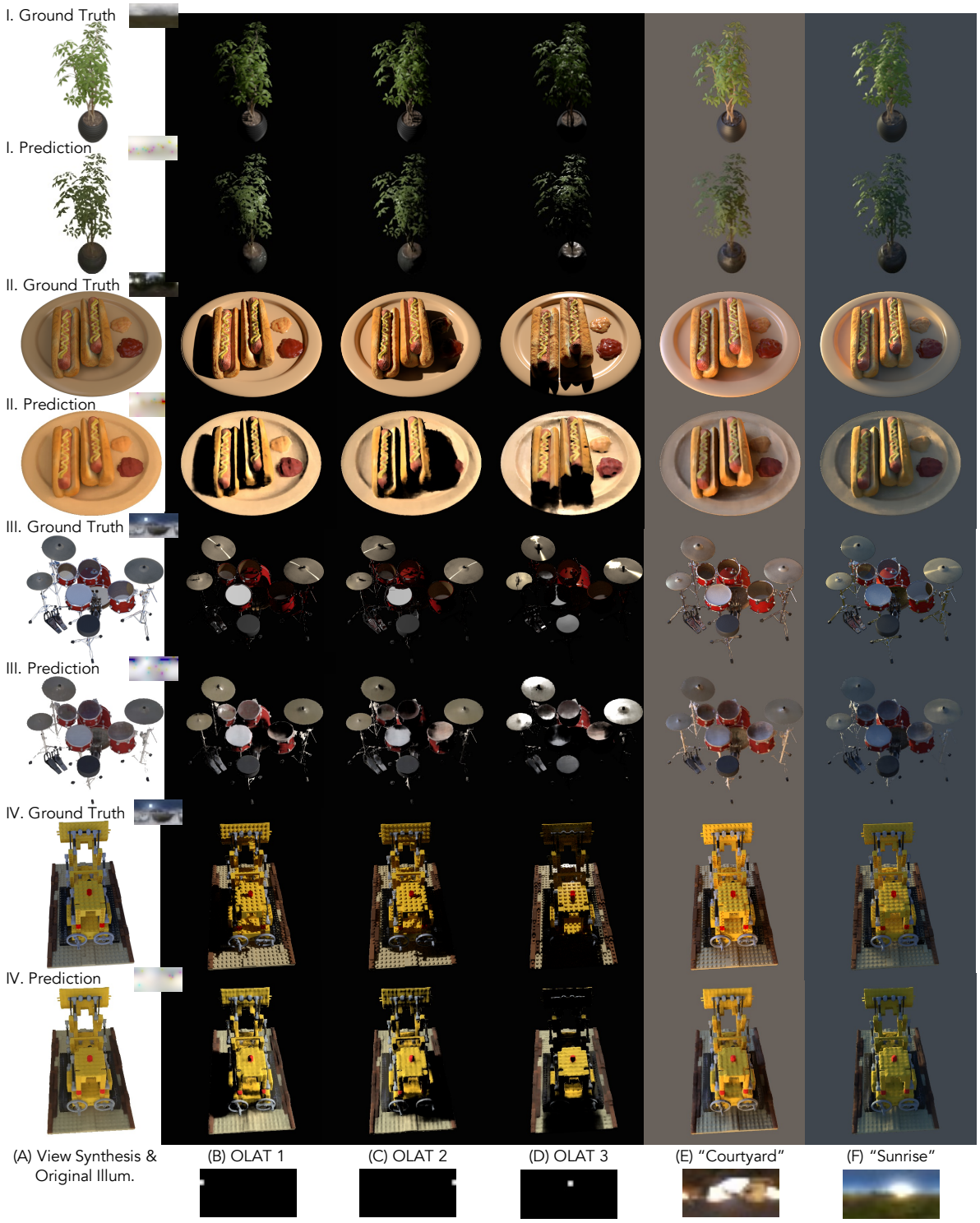


Fig. 5. **Free-viewpoint relighting.** The factorization that NeRFactor produces can be used to perform "free-viewpoint relighting": rendering a novel view of the object under arbitrary lighting conditions including the challenging One-Light-at-A-Time (OLAT) conditions. Here we relight the object using three OLAT illuminations and two real-world illuminations (light probes captured in the real world). The renderings produced by our model qualitatively resemble the ground truth and accurately exhibit challenging effects such as specularities and cast shadows (both hard and soft).

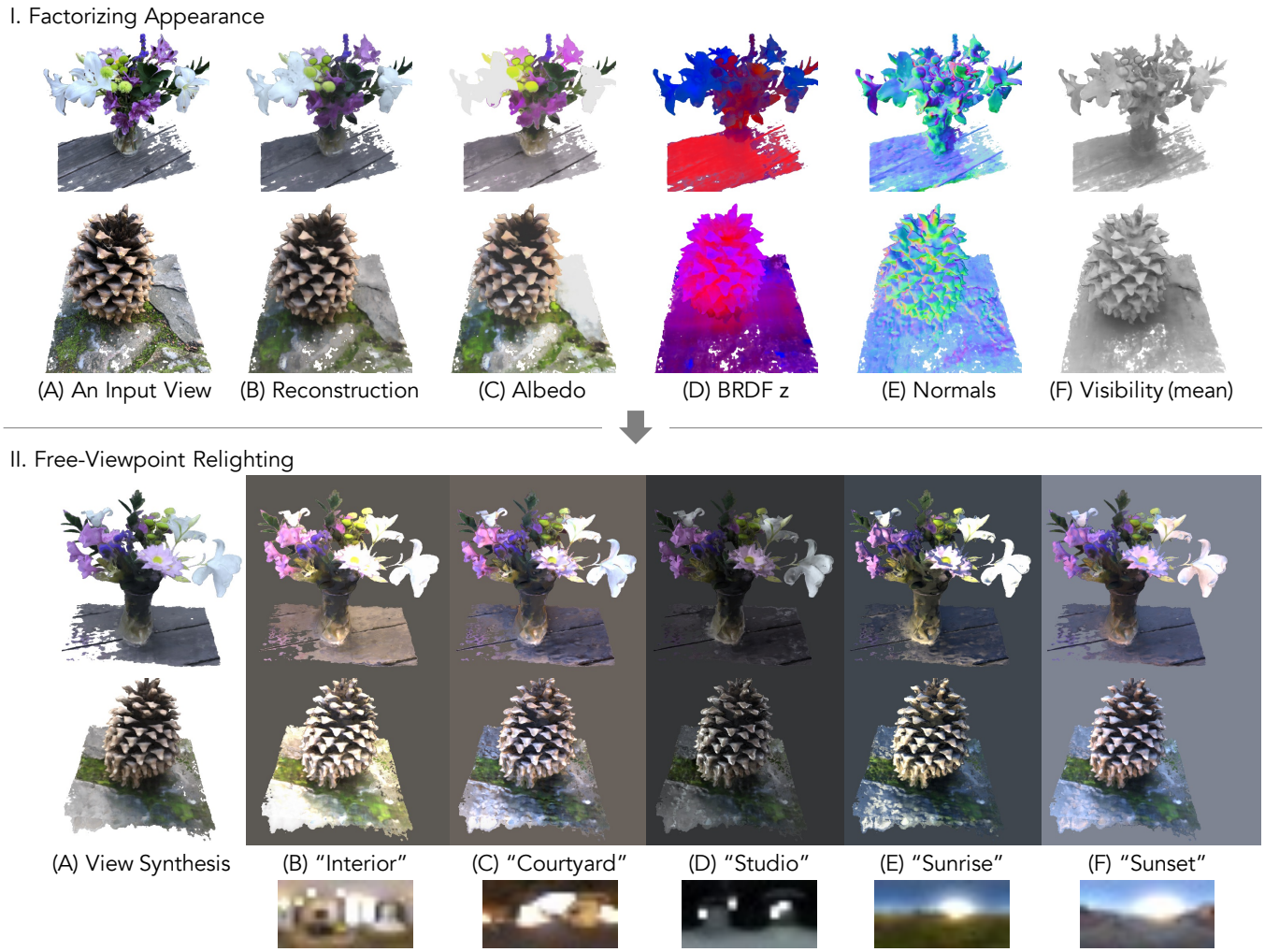


Fig. 6. **Results of real-world captures.** (I) Given posed multi-view images of a real-world object lit by an unknown illumination condition (A), NeRFactor factorizes the scene appearance into 3D fields of albedo (C), spatially-varying BRDF latent codes (D), surface normals (E), and light visibility for all incoming light directions, visualized here as ambient occlusion (F). Note how the estimated albedo of the flowers are shading-free. (II) With this factorization, one can synthesize novel views of the scene relit by any arbitrary lighting. Even on these challenging real-world scenes, NeRFactor is able to synthesize realistic specularities and shadows across various lighting conditions.

learning-based state of the art in the tasks of appearance factorization and relighting. For quantitative evaluations, we use Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [Wang et al. 2004], and Learned Perceptual Image Patch Similarity (LPIPS) [Zhang et al. 2018] as metrics.

5.1 Estimation Consistency Across Different Illuminations

In this experiment, we study how different illumination conditions affect the albedo estimation by NeRFactor. More specifically, we probe how consistent the estimated albedo predictions are across different input illumination conditions. To this end, we light the `ficus` scene with four drastically different lighting conditions as shown in Figure 8, and then estimate the albedo with NeRFactor from these four sets of multi-view images.

As Figure 8 shows, NeRFactor's predictions are similar across the four input illuminations, with pairwise PSNR $\geq$ 34.7 dB. Note that the performance on Case D is worse (e.g., the specularity residuals on the vase) than on Case C, despite that both cases seem to have the sun as the primary light source. The reason is that Case D had the sun pixels properly measured by Stumpfel et al. [2004], whereas Case C is an internet light probe that clipped the sun pixels. Therefore, Case D has a much higher-frequency lighting condition than Case C, making it a harder case for NeRFactor to correctly factorize the appearance.

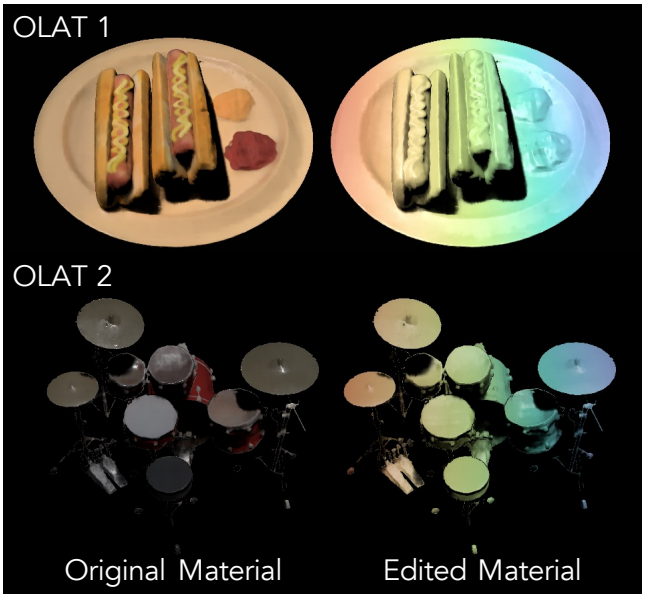


Fig. 7. **Material editing and relighting.** The factorization produced by NeRFactor can be used for material editing. Here we show the original materials of two scenes relit by two OLAT conditions (left) alongside the edited materials relit by the same OLAT conditions (right). Specifically, we overrode the predicted $\boldsymbol{z}_{\text{BRDF}}$ to that of `pearl-paint` in the MERL dataset and the predicted albedo to colors interpolated from the `turbo` colormap, varying spatially based on the surface points' x-coordinates.

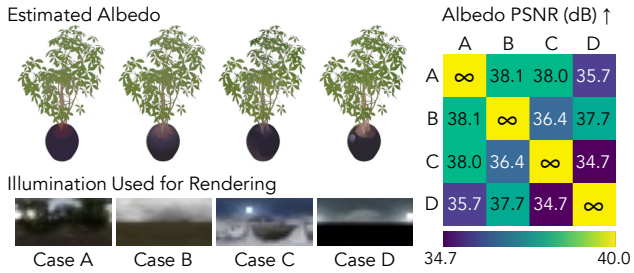


Fig. 8. **Albedo estimation consistency across different input illumination conditions.** The albedo fields recovered by NeRFactor are largely consistent across varying illumination conditions of the input images. Although both Case C and Case D have the sun as the primary light source, the performance on Case D is worse (e.g., the specularity residuals on the vase) because it is a challenging high-frequency lighting condition that has the sun intensity properly measured by Stumpfel et al. [2004], while Case C is an internet light probe that clips the sun intensity.

5.2 Ablation Studies

In this section, we compare NeRFactor against other reasonable design alternatives by ablating each of the important model components and observing whether there is performance drop, both qualitatively and quantitatively.

Learned BRDFs vs. microfacet BRDFs. Instead of using an MLP to parametrize the BRDF and pretraining it on an external BRDF dataset to learn data-driven priors, one can adopt an analytic BRDF model such as the microfacet model of Walter et al. [2007] and ask an MLP to predict spatially-varying roughness for the microfacet BRDF. As Table 1 shows, this model variant achieves good performance across all tasks, but overall underperforms NeRFactor. Note that to improve this variant, we removed the smoothness constraint on the predicted roughness because even a tiny smoothness weight still drove the optimization to the local optimum of predicting maximum roughness everywhere (this local optimum is a "safe" solution that renders everything more diffuse to satisfy the ℓ^2 reconstruction loss). As such, this model variance sometimes produces noisy rendering due to its non-smooth BRDFs in the supplemental video.

With vs. without geometry pretraining. As shown in Figure 2a and discussed in Section 3, we pretrain the normal and visibility MLPs to just reproduce the NeRF values given $\boldsymbol{x}_{\text{surf}}$ before plugging them into the joint optimization (where they are then fine-tuned together with the rest of the pipeline), to prevent the albedo MLP from mistakenly attempting to explain way the shadows. Alternatively, one can train these two geometry MLPs from scratch together with the pipeline. As Table 1 shows, this variant indeed predicts worse albedo with shading residuals (Figure 9 [C]) and overall underperforms NeRFactor.

With vs. without smoothness constraints. In Section 3, we introduce our simple yet effective spatial smoothness constraints in the context of MLPs and their crucial role in this underconstrained setup. Ablating these smoothness constraints does not prevent this variant from performing well on view synthesis (similar to how NeRF is capable of high-quality view synthesis without any smoothness constraints) as shown in Table 1, but does hurt this variant's performance on other tasks such as albedo estimation and relighting. Qualitatively, this variant produces noisy estimations insufficient for relighting (Figure 9 [B]).

Estimating the shape vs. using NeRF's shape. If we ablate the normal and visibility MLPs entirely, this variant is essentially using NeRF's normals and visibility without improving upon them (hence "using NeRF's shape"). As Table 1 and the supplemental video show, even though the estimated reflectance is smooth (encouraged by the smoothness priors from the full model), the noisy NeRF normals and visibility produce artifacts in the final rendering.

5.3 Baseline Comparisons

In this section, we compare NeRFactor with both classic and deep learning-based state of the art in the tasks of appearance factorization and free-viewpoint relighting.

SIRFS [Barron and Malik 2015]. We compare the factorization by NeRFactor with that of the classic SIRFS method [Barron and Malik 2015], both qualitatively and quantitatively. SIRFS is a single-image method that decomposes appearance into surface normals, albedo, and shading (not shadowing) in the input view. In contrast, NeRFactor is a multi-view approach that estimates these properties plus BRDFs and visibility (hence, shadows) in the full 3D space alongside the unknown illumination. In other words, NeRFactor gets to observe many more views than SIRFS, which observes only one

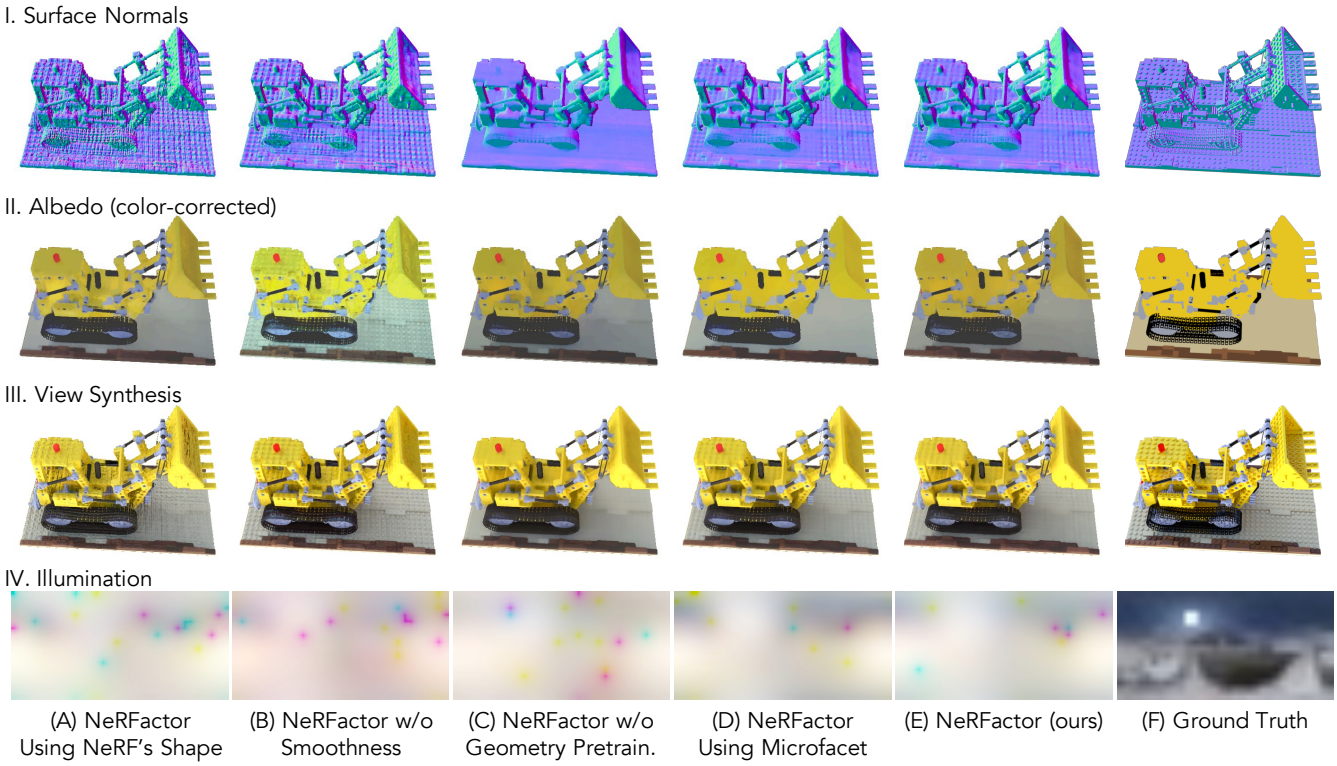


Fig. 9. **Ablation studies.** (A) One can fix the geometry to that of NeRF and estimate only the reflectance and illumination by ablating the normal and visibility MLPs of NeRFactor, but the NeRF geometry is too noisy (I) to be used for relighting (supplemental video). (B) Ablating the smoothness regularization leads to noisy geometry and albedo (I and II). (C) If we train the normal and visibility MLPs from scratch during the joint optimization (i.e., no pretraining), the recovered albedo may mistakenly attempt to explain shading and shadows (III). (D) If we replace the learned BRDF with an MLP predicting the roughness parameter of a microfacet BRDF, the predicted reflectance either falls into the local optimum of maximum roughness everywhere or becomes non-smooth spatially (not pictured here; see the supplemental video). (E) NeRFactor is able to recover *plausible* normals, albedo, and illumination without any direct supervision on any factor. The illuminations recovered by NeRFactor, though oversmoothed, correctly capture the location of the sun. For visualization, we apply gamma correction to the raw albedo values and the predicted HDR light probes ($\gamma = 2.2$). In addition, we performed the same per-channel albedo scaling as is done in Table 1 to resolve the global scale ambiguity.

Table 1. **Quantitative evaluations.** Reported numbers are the arithmetic means of all four synthetic scenes (`hotdog`, `ficus`, `lego`, and `drums`) over eight uniformly sampled novel views. The top three performing techniques for each metric are highlighted in red, orange, and yellow, respectively. For Tasks IV and V, we relight the scenes with 16 novel lighting conditions: eight OLAT conditions plus the eight light probes included in Blender. Since albedo is assumed to be recoverable only up to a per-channel scale ambiguity, we scale each RGB channel of all albedo predictions to match the scale of the corresponding ground-truth albedo channel (and then apply a $\gamma = 2.2$ gamma correction) before computing the albedo errors. Although ablating the smoothness constraints ("w/o smoothness") achieves good view synthesis quality under the original illumination, the noisy estimates lead to poor relighting performance. NeRFactor achieves the top overall performance across most metrics, although for some metrics it is outperformed by the microfacet variant ("using microfacet"), which tends to either converge to the local optimum of maximum roughness everywhere or produce non-spatially-smooth BRDFs (see the supplemental video). We are unable to present normal, view synthesis, or relighting metrics for SIRFS [Barron and Malik 2015] as it does not support non-orthographic cameras or "world-space" geometry (although Figure 10 shows that the geometry recovered by SIRFS is inaccurate). †Oxholm and Nishino [2014] require the ground-truth illumination, which we provide, and this baseline represents a significantly enhanced version (see Section 5.3).

	I. Normals	**II. Albedo**			**III. View Synthesis**			**IV. FV Relighting (point)**			**V. FV Relighting (image)**		
	Angle° ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SIRFS	-	26.0204	0.9420	0.0719	-	-	-	-	-	-	-	-	-
Oxholm & Nishino†	32.0104	26.3248	0.9448	0.0870	29.8093	0.9275	0.0810	20.9979	0.8407	0.1610	22.2783	0.8762	0.1364
NeRFactor	22.1327	28.7099	0.9533	0.0621	32.5362	0.9461	0.0457	23.6206	0.8647	0.1264	26.6275	0.9026	0.0917
using microfacet	22.1804	29.1608	0.9571	0.0567	32.4409	0.9457	0.0458	23.7885	0.8642	0.1256	26.5970	0.9011	0.0925
w/o geom. pretrain.	25.5302	27.7936	0.9480	0.0677	32.3835	0.9449	0.0491	23.1689	0.8585	0.1384	25.8185	0.8966	0.1027
w/o smoothness	26.2229	27.7389	0.9179	0.0853	32.7156	0.9450	0.0405	23.0119	0.8455	0.1283	26.0416	0.8887	0.0920
using NeRF's shape	32.0634	27.8183	0.9419	0.0689	30.7022	0.9210	0.0614	22.0181	0.8237	0.1470	24.8908	0.8651	0.1154

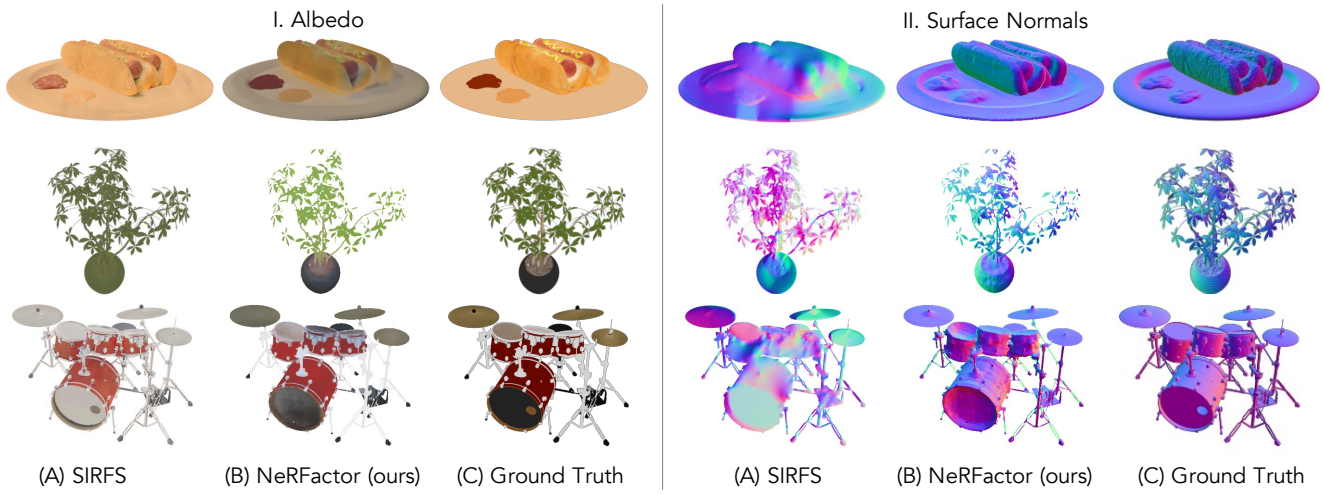


Fig. 10. **Comparisons with SIRFS in shape and albedo estimation.** Here we compare NeRFactor against SIRFS [Barron and Malik 2015] that is also intended to recover normals, albedo, and shading (not shadow) given only one illumination condition. Although the albedo estimates produced by SIRFS are reasonable, the surface normals are highly inaccurate (likely due to SIRFS's inability to use multiple images to inform shape estimation).

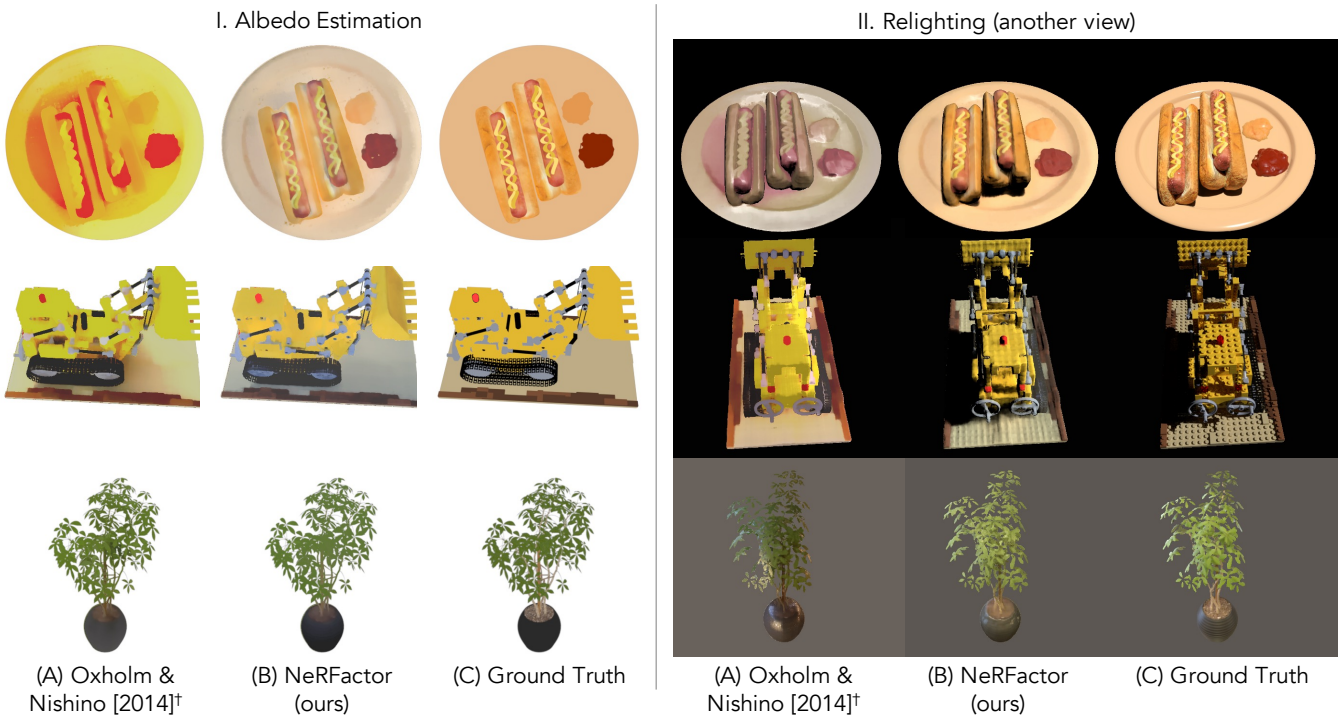


Fig. 11. **Comparisons with a significantly enhanced version of Oxholm and Nishino [2014] in albedo estimation and relighting.** (I) Their method is unable to remove shadow residuals from albedo for `hotdog` and `lego` likely due to its inability to model visibility or shadows, although it produces reasonable albedo estimation for `ficus` wherein shading (instead of shadowing) predominates. In contrast, NeRFactor produces albedo maps with little to no shading. (II) As expected, the baseline's relighting results are negatively affected by the shadow residuals in albedo (e.g., the red shade on the plate of `hotdog`). Furthermore, because their approach does not support spatially-varying BRDFs, the `hotdog` buns and `ficus` leaves are mistakenly estimated to be as specular as the plate and the vase, respectively. NeRFactor, on the other hand, correctly estimates different materials for different parts of the scenes. Note also how NeRFactor is able to synthesize hard shadows in `hotdog` and `lego`, while the baseline does not model visibility or shadows. †See Section 5.3 for how we significantly enhanced the approach of Oxholm and Nishino [2014]; in addition, we provide the baseline with the ground-truth illumination, since unlike NeRFactor, it does not estimate the lighting condition

(A) Philip et al. [2019] (B) NeRFactor (ours) (C) Ground Truth

	PSNR ↑	SSIM ↑	LPIPS ↓
(A) Philip et al. [2019]	20.0397	0.5000	0.1812
(A) Philip et al. [2019] + Masks	21.8620	0.8436	**0.1140**
(B) NeRFactor (ours)	**22.9625**	**0.8592**	0.1230

Fig. 12. **Comparisons with Philip et al. [2019] in point light relighting.** The approach of Philip et al. [2019] is a neural IBR method that supports relighting with a primary light source and synthesizing realistic shadows. The "yellow fog" (A) is likely due to the badly reconstructed geometry by their approach. For a more generous comparison, we additionally compute the errors after masking out these fog artifacts with the ground-truth masks ("Philip et al. [2019] + Masks"). Same as in Table 1, we scale each prediction's RGB channels independently to match the corresponding ground truth's global intensity and color tones. The numbers here are averages over eight test OLAT conditions. Quantitatively, NeRFactor outperforms "Philip et al. [2019] + Masks" in PSNR and SSIM, while the perceptual metric LPIPS favors the latter for this IBR baseline's sharp images. However, qualitatively, shadows synthesized by NeRFactor resemble the ground truth more, while the baseline's shadows tend to be overly soft (top) or cover a less accurate region (bottom). Note that unlike NeRFactor, this baseline does not support relighting with arbitrary lighting (such as another random light probe).

view. Under this setup, NeRFactor outperforms SIRFS quantitatively as shown by Table 1. Figure 10 shows that although SIRFS achieves reasonable albedo estimation, it produces inaccurate surface normals likely due to its inability to incorporate multiple views or to reason about shape in "world space." In addition, SIRFS is unable to render the scene from arbitrary viewpoints or synthesize shadows during relighting.

Oxholm and Nishino [2014]. Given that SIRFS is single-view, we additionally compare NeRFactor with a significantly improved version of the multi-view approach by Oxholm and Nishino [2014] that estimates the shape and non-spatially-varying BRDF under a *known* lighting condition. Due to the source code being unavailable, we re-implemented this method, capturing the main ideas of smoothness regularization on shape and data-driven BRDF priors, and then enhanced it with a better shape initialization (visual hull → NeRF shape) and the ability to model spatially-varying albedo (the original paper considers only non-spatially-varying BRDFs). Other differences include representing the shape with a surface normal MLP instead of mesh, and expressing the predicted BRDF with a pretrained BRDF MLP instead of MERL BRDF bases [Nishino 2009; Nishino and Lombardi 2011; Lombardi and Nishino 2012]. Also note that this baseline has the advantage of receiving the ground-truth illumination as input, whereas NeRFactor has to estimate illumination together with shape and reflectance.

As shown in Figure 11 (I), even though this improved version of Oxholm and Nishino [2014] has access to the ground-truth illumination, it struggles to remove shadow residuals from the albedo estimation because of its inability to model visibility (`hotdog` and `lego`). As expected, these residuals in albedo negatively affect the relighting results in Figure 11 (II) (e.g., the red shade on the `hotdog` plate). Moreover, because the BRDF estimated by this baseline is not spatially-varying, BRDFs of the `hotdog` buns and the `ficus` leaves are incorrectly estimated to be as specular as the plate and vase, respectively. Finally, this baseline is unable to synthesize non-local light transport effects such as shadows (`hotdog` and `lego`), in contrast to NeRFactor that correctly produces realistic hard cast shadows under the OLAT conditions.

Philip et al. [2019]. The recent work of Philip et al. [2019] presents a technique to relight large-scale scenes, and specifically focuses on synthesizing realistic shadows. The input to their system is similar to ours: multi-view images of a scene lit by an unknown lighting condition. However, their technique only supports synthesizing images illuminated by a single primary light source, such as the sun (in contrast to NeRFactor, which supports any arbitrary light probe). As such, we compare it with NeRFactor only on the task of point light relighting.

As Figure 12 demonstrates, NeRFactor qualitatively outperforms this baseline and synthesizes hard shadows that better resemble the ground truth. The "yellow fog" in the background of their results (Figure 12 [A]) is likely due to poor geometry reconstruction by their method. Because their network is trained on outdoor scenes (not images with backgrounds), we additionally compute error metrics after masking out the yellow fog with the ground-truth object masks ("Philip et al. [2019] + Masks") for a more generous comparison. As the table in Figure 12 shows, NeRFactor outperforms "Philip et al. [2019] + Masks" in terms of both PSNR and SSIM. The "masked" variant of Philip et al. [2019] achieves a lower (better) LPIPS score because it renders new viewpoints by reprojecting observed images using estimated proxy geometry, as is typical of Image-Based Rendering (IBR) algorithms. Thus, it retains the high-frequency details present in the input images, resulting in a lower LPIPS score. However, as a physically-based (re-)rendering approach that operates fully in the 3D space, NeRFactor synthesizes hard shadows that better match the ground truth and supports relighting with arbitrary light probes such as "Studio", which has four major light sources in Figure 6.

6 LIMITATIONS

Although we demonstrate that NeRFactor outperforms baseline methods and variants with different design choices, there are a few important limitations. First, to keep light visibility computation tractable, we limit the resolution of the light probe images to 16×32, a resolution that may be insufficient for generating very hard

shadows and recovering very high-frequency BRDFs. As such, when the object is lit by a very high-frequency illumination such as the one in Figure 8 (Case D) where the sun pixels are fully HDR, there might be specularity or shadow residuals in the albedo estimation such as those on the vase. Second, for fast rendering, we consider only single-bounce direct illumination, so NeRFactor does not properly account for indirect illumination effects. Finally, NeRFactor initializes its geometry estimation with NeRF. While it is able to fix errors made by NeRF up to a certain degree, it can fail if NeRF estimates particularly poor geometry in a manner that happens to not affect view synthesis. We observe this in the real scenes, which contain faraway incorrect "floating" geometry that is not visible from the input cameras but casts shadows on the objects of interest.

7 CONCLUSION

In this paper, we have presented Neural Radiance Factorization (NeRFactor), a method that recovers an object's shape and reflectance from posed multi-view images. Importantly, NeRFactor recovers these properties from images under an unknown illumination condition, while the majority of prior work requires observations under multiple known illumination conditions. To address the ill-posed nature of this problem, NeRFactor relies on priors to estimate a set of plausible shape, reflectance, and illumination that collectively explain the observed images. These priors include simple yet effective spatial smoothness constraints (implemented in the context of MLPs) and a data-driven prior on real-world BRDFs. We demonstrate that NeRFactor achieves high-quality geometry sufficient for relighting and view synthesis, produces convincing albedo as well as spatially-varying BRDFs, and generates lighting estimations that correctly reflect the presence or absence of dominant light sources. With NeRFactor's factorization, we can relight the object with point lights or light probe images, render images from arbitrary viewpoints, and even edit the object's albedo and BRDF. We believe this work makes important progress towards the goal of recovering fully-featured 3D graphics assets from casually-captured photos.

ACKNOWLEDGMENTS

We thank Julien Philip, Tiancheng Sun, and Zhang Chen for their help in providing their data or results, Zhoutong Zhang, Xuaner (Cecilia) Zhang, Yun-Tai Tsai, Jiawen Chen, Tzu-Mao Li, Yonglong Tian, and Noah Snavely for fruitful discussions, Noa Glaser and David Salesin for their constructive comments on the paper. This work was partially funded by the MIT-Air Force AI Accelerator.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. 2015. Two-Shot SVBRDF Capture for Stationary Materials. *TOG* (2015).

Jonathan T. Barron and Jitendra Malik. 2015. Shape, Illumination, and Reflectance From Shading. *TPAMI* (2015).

Sai Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020. Neural Reflectance Fields for Appearance Acquisition. *arXiv* (2020).

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. 2018. Optimizing the Latent Space of Generative Networks. In *ICML*.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik Lensch. 2020. NeRD: Neural Reflectance Decomposition From Image Collections. *arXiv* (2020).

Gershon Buchsbaum. 1980. A Spatial Processor Model for Object Colour Perception. *Journal of the Franklin Institute* (1980).

Zhang Chen, Anpei Chen, Guli Zhang, Chengyuan Wang, Yu Ji, Kiriakos N Kutulakos, and Jingyi Yu. 2020. A Neural Rendering Framework for Free-Viewpoint Relighting. In *CVPR*.

Paul Debevec. 1998. Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics With Global Illumination and High Dynamic Range Photography. *TOG* (1998).

Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. 2014. Appearance-From-Motion: Recovering Spatially Varying Surface Reflectance Under Unknown Lighting. *TOG* (2014).

Sing Choong Foo. 2015. *A Gonioreflectometer for Measuring the Bidirectional Reflectance of Material for Use in Illumination Computation.* Master's thesis. Cornell University.

Guan Gao, Guojun Chen, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2020. Deferred Neural Lighting: Free-Viewpoint Relighting From Unstructured Photographs. *TOG* (2020).

Stamatios Georgoulis, Vincent Vanweddingen, Marc Proesmans, and Luc Van Gool. 2015. A Gaussian Process Latent Variable Model for BRDF Inference. In *ICCV*.

Clement Godard, Peter Hedman, Wenbin Li, and Gabriel J. Brostow. 2015. Multi-View Reconstruction of Highly Specular Surfaces in Uncontrolled Environments. In *3DV*.

Purvi Goel, Loudon Cohen, James Guesman, Vikas Thamizharasan, James Tompkin, and Daniel Ritchie. 2020. Shape From Tracing: Towards Reconstructing 3D Object Geometry and SVBRDF Material From Images via Differentiable Path Tracing. In *3DV*.

Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, Danhang Tang, Anastasia Tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Paul Debevec, and Shahram Izadi. 2019. The Relightables: Volumetric Performance Capture of Humans With Realistic Relighting. *TOG* (2019).

Zhuo Hui, Kalyan Sunkavalli, Joon-Young Lee, Sunil Hadap, Jian Wang, and Aswin C. Sankaranarayanan. 2017. Reflectance Capture Using Univariate Sampling of BRDFs. In *ICCV*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Pierre-Yves Laffont, Adrien Bousseau, and George Drettakis. 2012. Rich Intrinsic Image Decomposition of Outdoor Scenes From Multiple Views. *TVCG* (2012).

Edwin H. Land and John J. McCann. 1971. Lightness and Retinex Theory. *JOSA* (1971).

Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. 2003. Image-Based Reconstruction of Spatial Appearance and Geometric Detail. *TOG* (2003).

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2020. Inverse Rendering for Complex Indoor Scenes: Shape, Spatially-Varying Lighting and SVBRDF From a Single Image. In *CVPR*.

Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Learning To Reconstruct Shape and Spatially-Varying Reflectance From a Single Image. *TOG* (2018).

Daniel Lichy, Jiaye Wu, Soumyadip Sengupta, and David W. Jacobs. 2021. Shape and Material Capture at Home. In *CVPR*.

Stephen Lombardi and Ko Nishino. 2012. Reflectance and Natural Illumination From a Single Image. In *ECCV*.

Stephen R. Marschner. 1998. *Inverse Rendering for Computer Graphics.* Ph.D. Dissertation. Cornell University.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A Data-Driven Reflectance Model. *TOG* (2003).

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *CVPR*.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H. Kim. 2018. Practical SVBRDF Acquisition of 3D Objects With Unstructured Flash Photography. *TOG* (2018).

Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. 2015. On Optimal, Minimal BRDF Sampling for Reflectance Acquisition. *TOG* (2015).

Ko Nishino. 2009. Directional Statistics BRDF Model. In *ICCV*.
Ko Nishino and Stephen Lombardi. 2011. Directional Statistics-Based Reflectance Model for Isotropic Bidirectional Reflectance Distribution Functions. *JOSA A* (2011).
Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. *arXiv* (2021).
Geoffrey Oxholm and Ko Nishino. 2014. Multiview Shape and Reflectance From Natural Illumination. In *CVPR*.
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *CVPR*.
Jeong Joon Park, Aleksander Holynski, and Steve Seitz. 2020. Seeing the World in a Bag of Chips. In *CVPR*.
Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A. Efros, and George Drettakis. 2019. Multi-View Relighting Using a Geometry-Aware Network. *TOG* (2019).
Ravi Ramamoorthi and Pat Hanrahan. 2001. A Signal-Processing Framework for Inverse Rendering. *TOG* (2001).
Ravi Ramamoorthi and Pat Hanrahan. 2004. A Signal-Processing Framework for Reflection. *TOG* (2004).
Szymon M. Rusinkiewicz. 1998. A New Change of Variables for Efficient BRDF Representation. In *Eurographics Workshop on Rendering Techniques*.
Shen Sang and Manmohan Chandraker. 2020. Single-Shot Neural Relighting and SVBRDF Estimation. In *ECCV*.
Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. 2003. Illumination From Shadows. *TPAMI* (2003).
Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. 1997. Object Shape and Reflectance Modeling from Observation. *TOG* (1997).
Carolin Schmitt, Simon Donne, Gernot Riegler, Vladlen Koltun, and Andreas Geiger. 2020. On Joint Estimation of Pose, Geometry and SVBRDF From a Handheld Scanner. In *CVPR*.
Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-From-Motion Revisited. In *CVPR*.
Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W. Jacobs, and Jan Kautz. 2019. Neural Inverse Rendering of an Indoor Scene from a Single Image. In *ICCV*.
Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matt Tancik, Ben Mildenhall, and Jonathan T. Barron. 2021. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *CVPR*.
Jessi Stumpfel, Chris Tchou, Andrew Jones, Tim Hawkins, Andreas Wenger, and Paul Debevec. 2004. Direct HDR Capture of the Sun and Sky. In *AFRIGRAPH*.
Tiancheng Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. 2018. Connecting Measured BRDFs to Analytic BRDFs by Data-Driven Diffuse-Specular Separation. *TOG* (2018).
Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*.
Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction Through Rough Surfaces. *Rendering Techniques* (2007).
Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *TIP* (2004).
Greg Ward and Rob Shakespeare. 1998. Rendering With Radiance: The Art and Science of Lighting Visualization. (1998).
Xin Wei, Guojun Chen, Yue Dong, Stephen Lin, and Xin Tong. 2020. Object-Based Illumination Estimation With Rendering-Aware Neural Networks. In *ECCV*.
Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. 2016. Recovering Shape and Spatially-Varying Surface Reflectance Under Unknown Illumination. *TOG* (2016).
Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *NeurIPS*.
Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. 1999. Inverse Global Illumination: Recovering Reflectance Models of Real Scenes From Photographs. *TOG* (1999).
Ye Yu and William A. P. Smith. 2019. InverseRenderNet: Learning Single Image Inverse Rendering. In *CVPR*.
Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021. PhySG: Inverse Rendering with Spherical Gaussians for Physics-Based Material Editing and Relighting. In *CVPR*.
Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.
Xiuming Zhang, Sean Fanello, Yun-Ta Tsai, Tiancheng Sun, Tianfan Xue, Rohit Pandey, Sergio Orts-Escolano, Philip Davidson, Christoph Rhemann, Paul Debevec, Jonathan T. Barron, Ravi Ramamoorthi, and William T. Freeman. 2020. Neural Light Transport for Relighting and View Synthesis. *TOG* (2020).