EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)

Volume 41 (2022), Number 2

Neural Precomputed Radiance Transfer

Gilles Rainer[1] Adrien Bousseau[1] Tobias Ritschel[2] George Drettakis[1]

[1] Inria, Université Côte d'Azur
[2] University College London

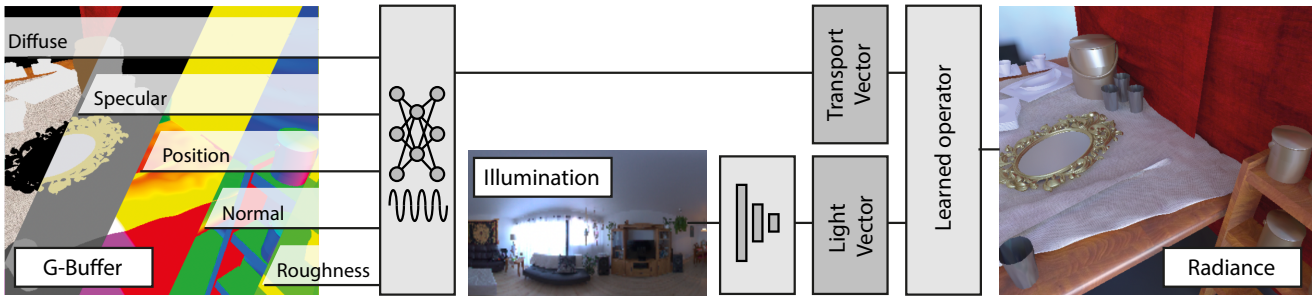


Figure 1: *We investigate the design space of scene-specific neural renderers that take environment map lighting and G-buffer pixels as input. We show that architectures based on design principles from the PRT literature can greatly improve quality at no extra cost.*

Abstract

Recent advances in neural rendering indicate immense promise for architectures that learn light transport, allowing efficient rendering of global illumination effects once such methods are trained. The training phase of these methods can be seen as a form of pre-computation, which has a long standing history in Computer Graphics. In particular, Pre-computed Radiance Transfer (PRT) achieves real-time rendering by freezing some variables of the scene (geometry, materials) and encoding the distribution of others, allowing interactive rendering at runtime. We adopt the same configuration as PRT – global illumination of static scenes under dynamic environment lighting – and investigate different neural network architectures, inspired by the design principles and theoretical analysis of PRT. We introduce four different architectures, and show that those based on knowledge of light transport models and PRT-inspired principles improve the quality of global illumination predictions at equal training time and network size, without the need for high-end ray-tracing hardware.

1. Introduction

Real-time global illumination has always been one of the major challenges of Computer Graphics. Recent progress in ray-tracing hardware [Bur20, NVI18] brings this goal closer, but at the cost of expensive, high-end ray-tracing GPUs, used for direct, exhaustive computation of path tracing. The history of global illumination research includes solutions that leverage *pre-computation* of light transport to offer fast rendering at run-time [RDGK12]. Interestingly, deep learning provides an analogue for *neural rendering* where precomputation – in the form of training – can be used to create neural representations for global illumination [NAM*17, ERB*18, GRPN20]. In this paper, we build on this similarity to propose several neural rendering methods (shown in Fig. 2), inspired by insights drawn from the extensive literature in traditional pre-computed global illumination.

Concerning traditional methods, we focus on Pre-computed Radiance Transfer (PRT) that makes certain aspects of the scene static – such as geometry and materials – and pre-computes representative configurations of the dynamic variables, such as camera and lighting. At run-time, the pre-computed data is efficiently de-compressed and interpolated. On the other hand, neural networks have demonstrated their ability to encode radiance distributions [BMSR20, MST*20], and to efficiently compress and interpolate reflectance data [RJGW19]. In both cases, pre-computation allows the light transport to be captured and represented in a compact data structure for fast evaluation at run-time.

Neural rendering techniques are rapidly offering exciting and very competitive solutions for traditional rendering, e.g., for indirect lighting [MRNK21] or complex luminaires [ZBX*21]. We claim it is important to exploit the extensive knowledge developed

by graphics research in the past 50 years when designing such neural rendering algorithms. We demonstrate this idea by exploring the application of neural networks to encode radiance in a static synthetic scene under dynamic illumination, inspired by PRT principles. The different design choices we present are guided by the well-established theory and practice of traditional PRT methods and allow us to inject physically-based inductive biases that improve the quality of the predictions.

We start by demonstrating the capacity of a small neural network to store and reproduce full global illumination of a medium-sized scene with various materials under dynamic environment illumination at interactive framerates. This neural architecture takes as input a series of G-buffers representing scene attributes from the current viewpoint, along with a target environment map, and processes each pixel independently to produce a shaded image. While this baseline shows promise, it has objectionable visual artifacts. Inspired by the theoretical analysis and the design principles of PRT, we explore three additional neural network architectures. Each alternative improves the fidelity of the run-time renderings, without additional memory cost. Our comparisons and results show that high visual quality can be achieved compared to alternative solutions, with lower memory requirements and without the need for high-end ray-tracing hardware.

2. Related Work

Our work builds on two very large domains: global illumination and neural rendering. We will thus limit our discussion of previous work to the most closely related literature. Our work targets rendering of *synthetic data*, where the full scene description – geometry, materials and lights – has been perfectly constructed, typically by an artist. Nonetheless, we have also been inspired by neural rendering methods originally developed for the acquisition and re-rendering of real-world data.

2.1. Neural Rendering for Real-world Data

Neural Rendering is a recent field that has seen an explosion in research output; a good survey is provided by Tewari et al. [TFT*20]. Techniques used for free-view synthesis, relighting and material representation have all provided methodologies inspiring our work.

Free view synthesis. Early work used appearance flow [ZTS*16], and later multi-plane image representations (e.g., [ZTF*18, MSOC*19]), that use Convolutional Neural Networks (CNNs) to synthesize novel views using a set of photos of a scene as input. Other approaches use 3D mesh proxies to learn features in texture space [TZN19] with impressive results. Recently Neural Radiance Fields (NeRF) [MST*20] represent a 3D scene using a Multi-Layer Perceptron (MLP) to store opacity (as a proxy for geometry) and view-dependent color. While all these methods target real-world data rather than our synthetic scenes, we also use CNNs and MLPs to represent different components of light transport, allowing realistic rendering.

Free viewpoint relighting. Real-world scenes built from photographs are "stuck" with the lighting conditions at the time of capture. Focusing on isolated objects, several methods attempt to relight captured scenes using neural networks, taking multiple flash-lit photos as input to directly learn lighting [GCD*20], or to explicitly reconstruct geometry, materials and lighting [ZLW*21]. Recent variants of MLP-based NeRF also attempt to learn either intermediate representations (NeRV [SDZ*21], Neural Reflectance Fields [BXS*20]) or explicit BRDFs (NeRD [BBJ*21], NeRFactor [ZSD*21]) for relighting. G-buffers, and in particular reflected directions have recently been used to help neural relighting [PMGD21]. Some solutions operate with point or directional lights [ZFT*21], and deal with environment lighting by generating a sequence of *one light at a time* renderings, often using complex light-stage [DHT*] setups for capture [PEL*21]. The main emphasis of these methods is inverse rendering to recover a representation enabling relighting from real-world data; the use of MLPs and G-buffers to encode scene properties for re-lighting share some similarities to our approach.

Materials. The use of neural networks to represent materials has received significant interest recently both for BRDFs [SRRW21] and for neural BTFs [RJGW19, RGJW20]. While focusing on capturing appearance, these methods provide interesting guidelines on representing materials even in the case of synthetic data.

2.2. Neural Rendering for Synthetic Data

Traditional rendering with synthetic data can also benefit from the use of neural networks and representations. The methodology is similar to the case of real-world data, as neural networks are also trained to encode distributions in a compact manner, but the input is produced by explicit analytic models rather than measurement devices. Geometry, materials and shading have all recently benefited from such neural representations to encode data that is costly to compute, allowing fast evaluation at run-time.

Geometry and Materials. Signed-distance fields (SDFs) are a popular continuous representation of geometry; neural representations achieve impressive levels of accuracy [PFS*19]. Different encodings such as SIREN [SMB*20] or ACORN [MLL*21] further improve the compactness and/or accuracy of such representations. The recent Neural Luminaires method goes further, and uses an MLP to represent both the geometry and the emission of complex spatial lighting [ZBX*21]. Neural methods have successfully encoded materials at the BRDF level (e.g., Deep appearance maps [MRLTF19]), or with generative models for normal distribution functions [KHX*19]. Recent solutions include multi-scale neural representation of BTFs [KMX*21] and the use of NeRF-like MLPs to encode and render volumetric textures [BGP*21].

In contrast to these approaches, we use traditional representations of geometry and materials to generate G-buffers, and focus on encoding light transport to turn these buffers into a shaded image with global illumination.

Rendering/Shading. In their pionneering work, Ren et al. [RWG*13] express indirect illumination from point lights using so-called *radiance regression functions*, represented as multiple small neural networks distributed over the scene. Deep

shading [NAM*17] presented the first complete neural renderer, which demonstrated that a neural network can learn various visual effects such as depth of field, indirect illumination etc. In a similar spirit, Eslami et al. [ERB*18] used a neural network to represent the radiance function over a class of scenes. In follow-up work, Granskog et al. [GRPN20] investigate disentangling lighting and geometry. The Neural Radiance Cache learns indirect illumination online, in the context of real-time path tracing [MRNK21], while Neural Radiosity trains a network to learn both sides of the rendering equation [HCZ21]. Neural methods have also been used extensively to denoise path-traced images [HY21].

These methods show how the power of neural networks can be used to improve or replace parts of the rendering pipeline. Our starting point is in a similar spirit, since our baseline approach is a *radiance-regressing network*, i.e., a neural network that is trained to directly output radiance from G-buffers.

2.3. Traditional PRT

Many neural rendering approaches share an expensive pre-processing step: a large amount of data is either captured from a real world scene or generated from a synthetic scene description, then used to train a network. In many cases, the goal of the neural network is not to generalize to new scenes but rather to encode scene-specific information, effectively offering a compressed representation of the input data. One of our key observations is the similarity of this process to Precomputed Radiance Transfer, which leverages scene-specific pre-processing to allow real-time run-time rendering. A complete overview of PRT and other methods for fast GI is given in the STAR report by Ritschel et al. [RDGK12]; we only cover the most relevant work here.

PRT was introduced by Sloan et al [SKS02]. The key idea is to choose an angular basis of continuous functions – Spherical Harmonics (SH) in this case – and to perform all light transport operations in that domain. In a preprocess, transfer matrices, which encode the transformation from distant light to local incoming light in basis coordinates, are computed at every scene vertex and stored. The massive data is then compressed using a variation of PCA. At run-time, the environment lighting and the materials are projected into SH, and all light transport is computed trivially via matrix multiplications in the SH domain.

The choice of angular basis is central in PRT and many options have been proposed. Spherical Harmonics are limited in terms of high frequencies, unlike Haar wavelets used by Ng et al [NRH04]. Tsai and Shih introduce spherical radial basis functions [TS06] which are a more general form of Spherical Gaussians [GKMD, WRG*09]. The exploration of bases culminated in Anisotropic Spherical Gaussians [XSD*13] which offer the most flexibility and expressiveness. Our PRT-inspired neural renderers can be seen as learning an implicit basis, which improves quality through scene-specific learning.

PRT is a generic framework that offers not only a lot of freedom in the choice of angular basis, but also in the content that is precomputed and the type of scenes that are modelled. Early work focused on reducing the storage of precomputed data [SHHS03], dynamic geometry [SLS05, PWXLPB07], dynamic materials [SZC*07], one-bounce interreflections [XCM*14] and recently near-field illumination [WR18, WCZR20].

In contrast to the majority of PRT methods that focus on direct lighting/shadowing, our neural approach models all light transport effects, i.e., full global illumination.

2.4. Neural Methods and PRT

The last few years have seen interest in using Deep Learning tools within traditional PRT frameworks. Ren et al. [RDL*15] use neural networks to learn and compress the light transport matrix as a function of light position and pixel coordinates, for a static viewpoint. Xu et al. [XSHR18] generalize this method across scenes with a deep convolutional neural network. Li et al. [LWM19] use a CNN to predict PRT-SH coefficients of a deformable object within an animation, simplifying precomputation. Similarly, Currius et al. [CDAS20] use a CNN to predict Spherical Gaussian coefficients of the transfer matrices, which works more coherently and robustly than other optimisation methods. More explicitly, Jiang and Kainz [JK21] use a CNN to upsample maps of incident radiance computed at the first intersection, to regularize local incoming lighting. In contrast, we develop a pure neural rendering method (no path tracing needed) inspired by PRT.

More recently, ideas from PRT have been used in the context of neural rendering. In the Plenoctrees method, Spherical Harmonics are used to encode the directional information of NeRF, allowing faster rendering [YLT*21]. PRT has been used as an efficient rendering method to allow relighting of humans; a neural network first learns to decompose a single image into albedo, illumination and light transport [LSY*21]. NeX [WPYS21] uses a neural basis of angular radiance distributions, that can be seen as a form of learnt PRT basis. Similar to those in Sec. 2.1, these methods learn lighting/rendering from real data, while our method operates on efficient global illumination for *synthetic scenes*.

3. Overview

In the related work, we showed the immense potential of neural networks in the traditional rendering pipeline, and outlined the analogies with PRT. Our goal is to explore the design space of neural rendering in the same context as traditional PRT, e.g. efficient low-cost rendering of static scenes with dynamic lighting via use of precomputation.

We hence operate in the same setting as classic PRT: a scene with fixed object geometries and materials, and dynamic, arbitrary environment lighting described by environment maps. We additionally fix the neural budget, i.e., training time and network size, and investigate architectures inspired by PRT methodology and principles.

Section 4 gives an overview of PRT principles and the traditional PRT framework, as opposed to the Monte-Carlo sampling approach of path tracing. Section 5 introduces our PRT-inspired architectures and motivates the design choices via analogies with the PRT background. In Section 6, we detail the experimental setup (training and data generation strategies) and provide technical details. Finally, we evaluate our alternatives and compare our results to different approaches in Section 7.

4. Background on Precomputed Radiance Transfer

We first introduce the main concepts on light transport that form the foundations for our neural-based PRT approaches.

4.1. Integral Formulation of Light Transport

In traditional rendering, the outgoing radiance L_o at a point $\mathbf{x}$ with normal $\mathbf{n}$ observed by the camera in direction ω_o is given by the Rendering Equation [Kaj86]:

$$L_o(\mathbf{x},\omega_o) = L_e(\mathbf{x},\omega_o) + \int_\Omega f_r(\mathbf{x},\omega_i,\omega_o) L_i(\mathbf{x},\omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i \quad (1)$$

where L_e is the scene's lighting, f_r is the reflectance function, L_i is the incoming radiance reflected from other locations in the scene, Ω is the hemisphere of incoming directions ω_i. In path-tracing, this integral equation is solved by Monte-Carlo aggregation of samples (rays/paths through the scene).

We focus on scenes lit with distant environment lighting, meaning the lighting only has a directional but no spatial dependency, and is only observed in the background. For clarity, we hence remove the lighting term from the equation, and reveal the material parameters (diffuse albedo $\mathbf{k}_d$, specular albedo $\mathbf{k}_s$ and roughness α) that modulate the reflectance function f_r at $\mathbf{x}$:

$$L_o(\mathbf{x},\omega_o) = \int_\Omega f_r(\omega_i,\omega_o,\mathbf{k_d},\mathbf{k}_s,\alpha) L_i(\mathbf{x},\omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i \quad (2)$$

Diffuse-Specular Separation. The reflectance function at a given point is often approximated as:

$$f_r(\omega_i,\omega_o,\mathbf{k}_d,\mathbf{k}_s,\alpha) = \frac{\mathbf{k}_d}{\pi} + \mathbf{k}_s \frac{D(\alpha)FG}{4(\omega_i \cdot \mathbf{n})(\omega_o \cdot \mathbf{n})}, \quad (3)$$

where the first term models diffuse reflectance, while the second term models view-dependent specular/glossy effects. In the second term, D, F and G respectively model the distribution of orientations of microfacets controlled by roughness α, the Fresnel factor and the geometric term. Given this split of the reflectance function, the outgoing radiance can be refactored as the sum of diffuse and specular/glossy contributions, L_o^D and L_o^S, each weighted by their respective albedos:

$$L_o(\mathbf{x},\omega_o) = \mathbf{k}_d L_o^D + \mathbf{k}_s L_o^S \quad (4)$$

$$L_o^D = \int_\Omega \frac{(\omega_i \cdot \mathbf{n})}{\pi} L_i(\mathbf{x},\omega_i) d\omega_i \quad (5)$$

$$L_o^S = \int_\Omega \frac{D(\alpha)FG}{4(\omega_i \cdot \mathbf{n})(\omega_o \cdot \mathbf{n})} L_i(\mathbf{x},\omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i \quad (6)$$

4.2. Matrix Formulation of Light Transport

One of the governing principles of Precomputed Radiance Transfer, originally described in the seminal work of Sloan et al. [SKS02], is the projection of lighting and materials into a compact, continuous function space. This approach was later formalized by Lehtinen [Leh07] in a framework that provides an operator-driven view of light transport, as opposed to the sample-driven view of Monte Carlo rendering. Within this framework, the rendering equation is expressed in abstract form as

$$L = E + \mathcal{T}L, \quad (7)$$

where E is the emitted radiance from light sources, L is the light distribution in the scene, and $\mathcal{T}$ is the *transport operator*. Due to the recursive nature of the equation, the author shows that the solution to the global illumination problem can be written as:

$$L = \mathcal{S}E \quad (8)$$

where $\mathcal{S}$ is the *solution* operator. In this abstract framework, all operators are infinite-dimensional. In practice, PRT methods define angular bases of projection to reduce the problem to a linear equation. E and L are represented as vectors of coefficients in the basis of choice, $\mathcal{S}$ is expressed as a matrix, and Equation 8 becomes the matrix-vector product:

$$\mathbf{l} = \mathsf{S}\mathbf{e} \quad (9)$$

Transport vs. Transfer. Another important principle of Precomputed Radiance Transfer is the separation of light transport into *transfer* and *convolution* with the reflectance function. Transfer refers to the transformation of the spherical distribution of distant light (the environment map) to local light L_i, incoming at the given point. This transfer contains all global illumination effects of light transport throughout the scene. The only step left to obtain outgoing radiance is the convolution of L_i with the reflectance function f_r, as expressed by Equation 2. In the operator view of light transport, this is equivalent to factoring out the last bounce of light in the scene, before it hits the camera, and expressing the convolution with the local reflection operator $\mathcal{R}$:

$$L = \mathcal{R}\mathcal{S}E. \quad (10)$$

The main reasoning behind this separation is to pre-compute the costly light transfer and store it as a matrix S, and only compute the convolution with the reflectance function at run-time, which can be done efficiently in an appropriate angular basis. Additionally, the pre-computed matrix only stores the transfer, which usually has lower-frequency spatial variation than the transport, for instance in the presence of textured objects with spatially-varying reflectance.

Diffuse-Specular Separation. Similarly to the integral formulation of Equation 4, the reflection operator can be decomposed into a diffuse and a specular term to yield:

$$L = (\mathcal{R}^D + \mathcal{R}^S)\mathcal{S}E, \quad (11)$$

where $\mathcal{R}^D$ amounts to convolving the incoming lighting with a clamped cosine lobe centered on the surface normal, while $\mathcal{R}^S$ corresponds to a convolution with the roughness-dependent glossy lobe centered on the mirrored direction. Factorizing back the reflection into the transport operator $\mathcal{S}$ reveals two different types of transport:

$$L = \mathcal{S}^D E + \mathcal{S}^S E. \quad (12)$$

This separation allows one to treat diffuse and specular/glossy transport separately, for instance by allocating more coefficients to the specular term since it typically contains higher frequencies, and by treating diffuse transport as a lower-dimensional problem as it does not depend on the view direction [SKS02].

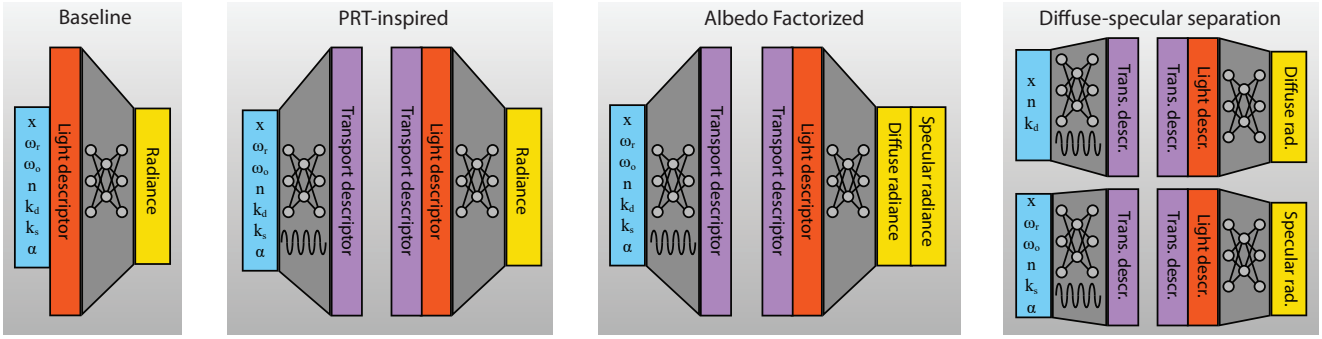


Figure 2: *The four network architectures we investigate. The light code is obtained from the environment map encoding CNN.*

Figure 3: *We use a standard CNN to encode the environment map lighting into a low-dimensional vector.*

5. Method

Traditional PRT methods leverage the linearity of the projection and the orthogonality of carefully-chosen basis functions such as Spherical Harmonics, to achieve real-time rendering with global illumination, at the cost of a loss of high-frequency content. We now revisit PRT principles from the new perspective of neural rendering, which allows us to define more expressive, non-linear representations of radiance and corresponding operators. We investigate the design space of such approaches, which leads us to present four alternatives of increasing complexity, illustrated in Figure 2.

5.1. Environment Map Encoding

The first step of most PRT methods is to project the lat-long environment map lighting E onto an angular basis, such that the map can be described by a low-dimensional vector $\mathbf{e}$.

Instead of relying on a pre-defined angular basis, we propose to embed the environment map into a *learned* compact space. We perform this dimensionality reduction using an encoder $\mathcal{F}_\mathrm{L}$, implemented as a Convolutional Neural Network (CNN) that outputs a 64-dimensional descriptor, as illustrated in Figure 3:

$$\hat{\mathbf{e}} = \mathcal{F}_\mathrm{L}(E). \tag{13}$$

We train this CNN encoder jointly with subsequent radiance transfer operators for each scene separately, which allows the encoder to focus its resources on the parts of the environment map that contribute most to the final renderings, while ignoring parts that are most often occluded.

5.2. Baseline Transport Operator

Given the compact lighting descriptor $\hat{\mathbf{e}}$, we first define a baseline architecture $\hat{\mathcal{S}}$ that we train to perform the entire light transport $L = \hat{\mathcal{S}}(\hat{\mathbf{e}})$ for a given scene. We implement this transport operator as a Multi-Layer Perceptron (MLP) with ReLU activations, which takes as input, along with $\hat{\mathbf{e}}$, scene attributes at the point to be shaded, in the form of G-buffers. These attributes include the position $\mathbf{x}$, view direction ω_o, normal direction $\mathbf{n}$, diffuse albedo $\mathbf{k}_\mathrm{d}$, specular albedo $\mathbf{k}_\mathrm{s}$, and roughness α, as appearing in Equation 2. We also provide the mirror direction $\boldsymbol{\omega}_\mathbf{r}$ – which corresponds to the reflection of the view direction around the surface normal – as an additional source of information to model specular/glossy effects. Our baseline network hence learns to approximate:

$$L = \hat{\mathcal{S}}(\hat{\mathbf{e}}, \mathbf{x}, \omega_\mathrm{o}, \omega_\mathrm{r}, \mathbf{n}, \mathbf{k}_\mathrm{d}, \mathbf{k}_\mathrm{s}, \alpha). \tag{14}$$

5.3. PRT-Inspired Transport Operator

As mentioned in Section 4.2, traditional Precomputed Radiance Transfer methods express global illumination as a fast linear operation by projecting both the input lighting and the solution operator onto a suitable basis to form the light vector $\mathbf{e}$ and the transport matrix S. Inspired by this principle, we next design a Multi-Layer Perceptron $\mathcal{F}_\mathrm{T}$ that encodes light transport at a point as a compact descriptor:

$$\hat{\mathbf{s}} = \mathcal{F}_\mathrm{T}(\mathbf{x}, \omega_\mathrm{o}, \omega_\mathrm{r}, \mathbf{n}, \mathbf{k}_\mathrm{d}, \mathbf{k}_\mathrm{s}, \alpha). \tag{15}$$

Since $\hat{\mathbf{e}}$ and $\hat{\mathbf{s}}$ are now produced by non-linear neural networks, we replace the linear matrix-vector product by a *learnt* operator Φ that applies the transport to the lighting. By analogy to Equation 8, this architecture computes:

$$L = \Phi(\hat{\mathbf{s}}, \hat{\mathbf{e}}). \tag{16}$$

From a neural network perspective, splitting the task into these two steps brings several benefits. First, the baseline method described in Section 5.2 concatenates the 64-dimensional lighting descriptor with the 19-dimensional vector of scene attributes before processing all this information with a single MLP. In contrast,

our second architecture first processes the scene attributes with $\mathcal{F}_T$ to embed them in a 64-dimensional space, such that lighting and transport are given equal dimensions before being processed by Φ. Second, using two separate MLPs allow us to adopt different architectures adapted to their respective tasks. On the one hand, since $\mathcal{F}_T$ needs to represent high-frequency light transport effects from a low-dimensional attribute vector, we implement it with SIREN activations [SMB*20] that have been shown to perform well on similar tasks. On the other hand, since Φ needs to map high-dimensional descriptors to a single RGB value, ReLU activations are more appropriate. We validate these design choices in Section 6.

5.4. Albedo Factorization

Our third design follows the insight provided by PRT that the reflection operator can be cheap to compute at run-time, and doing so alleviates unnecessary pre-computation and storage (Equation 10). However, our non-linear representation prevents treating reflection as a convolution, as done in traditional PRT. We instead only factor out the diffuse and specular albedos, which we multiply by the diffuse and specular/glossy contributions predicted from the lighting and transport descriptors:

$$(L^D, L^S) = \Phi(\hat{\mathbf{s}}, \hat{\mathbf{e}}) \tag{17}$$

$$L = \mathbf{k}_d L^D + \mathbf{k}_s L^S. \tag{18}$$

This separation of the task frees the neural networks from modeling spatially-varying albedos, which is especially beneficial on scenes with intricate textures. Apart from doubling the size of the final output, we keep the same architecture for $\mathcal{F}_T$ and Φ. In particular, we observed that even though $\mathbf{k}_d$ and $\mathbf{k}_s$ are used explicitly in the final shading computation, feeding them as input to $\mathcal{F}_T$ improves results, possibly because this additional information helps the network distinguish nearby points via the texture information.

5.5. Diffuse-Specular Separation

As noted in Section 4, the diffuse and specular terms of the reflectance involve different computations, which can benefit from being treated separately (Equation 12). Following this observation, our last design models the diffuse and specular/glossy transport operators as two distinct Multi-Layer Perceptrons. While this design induces some redundancy, since both MLPs need to learn the transfer operation $\mathcal{S}$, it allows to inject additional inductive bias within the input parameters of each MLP. Specifically, we only provide the position, normal and diffuse albedo to the MLP in charge of computing the diffuse transport descriptor $\hat{\mathbf{s}}^D = \mathcal{F}_T^D(\mathbf{x}, \mathbf{n}, \mathbf{k}_d)$, while we provide the view direction, reflection direction, roughness and specular albedo to the MLP in charge of computing the specular/glossy transport descriptor $\hat{\mathbf{s}}^S = \mathcal{F}_T^S(\mathbf{x}, \mathbf{n}, \mathbf{k}_s, \omega_o, \omega_r, \alpha)$. We also train distinct operators Φ^D and Φ^S to apply the diffuse and specular/glossy transport to the lighting, respectively. The final computation amounts to:

$$L = \mathbf{k}_d \Phi^D(\hat{\mathbf{s}}^D, \hat{\mathbf{e}}) + \mathbf{k}_s \Phi^S(\hat{\mathbf{s}}^S, \hat{\mathbf{e}}). \tag{19}$$

For fair comparisons, we decrease the number of neurons for each track to match the neural budget of the other architectures.

6. Data Generation and Training

Similarly to PRT, we operate in a setting where all time- and computation-heavy operations can be moved to a pre-processing phase with no restrictions on time budget. During this phase, we render ground truth images for various combinations of views and lighting, and train the networks, which can be seen as analog to the data compression step in PRT since the trained network equates to a lossy, compressed representation of the pre-computed renderings, with built-in interpolation.

Figure 4: *Examples from the validation dataset of* ATELIER.

Data Generation. We use the real-time path tracer Falcor [BYC*20] for data generation. For each scene, we render 3000 training images at 256×256 pixel resolution, and 200 test images at 400×400 pixel resolution (see Fig. 4 for examples). We use center sampling in the G-Buffers and output all layers that we subsequently feed to the networks (position, view direction, normal, diffuse and specular albedo, roughness); this allows precise validation without the interference of anti-aliasing.

For illumination, we use the datasets from the Laval HDR database. The indoor environment maps [GSY*17] are used to light the Atelier scene, while the dataset of outdoor environment maps [HGAL19] is used for San Miguel, Kitchen and Bedroom. In both cases, we perform a train/test split of the maps and only use them respectively for the train and test renderings. Additionally, the maps used as illumination in the training images are randomly rotated about the vertical axis, to augment the lighting conditions.

For each scene, we define an active volume where we randomly place the training and test cameras. Similarly, the viewing direction is randomly chosen within a pyramid of directions. After training, we visualize a camera path through this volume.

We opt for this camera specification strategy over a completely random one, to make sure that all renderings contribute valuable information to the learning. Otherwise some cameras might be very close to, or even on/inside objects, or looking into regions without important details. We leave adaptive or more involved camera placement strategies to future work.

Pre-processing Input Data. We pre-process the environment map inputs to the CNN encoder by normalizing by the mean, and applying a $\log(1+x)$ tonemap. The position buffer is also normalized (linearly) so that all possible position values in the scene lie between -1 and 1. The other inputs are given without transformation.

Encoder CNN Design. Our encoder consists of 5 blocks (convolution layer with a ReLU activation, max-pooling and a skip connection) followed by a fully-connected layer. The convolution layers use 32 channels and the final linear layer outputs a 64-dimensional vector, which we found to work well in practice.

Table 1: *Number of parameters per network and storage.*

	Light Encoder $\mathcal{F}_L$	Transport $\mathcal{F}_T$	Total	MB
Baseline	290,848	219,651	510,499	2.0
PRT-Insp.	290,848	186,947	477,795	1.8
Albedo Fact.	290,848	187,718	478,566	1.8
D/S Separ.	290,848	194,226	482,074	1.9

Design of the MLPs. We design the MLPs to have comparable numbers of trainable parameters. We give the naïve baseline network 4 layers of 256 neurons each. To match this, the PRT-inspired network and the network with separate diffuse-specular outputs have 2 layers of 256 for the SIREN, and 2 layers of 256 for the MLP. Finally, the network with separate tracks also has 2 layers per sub-network, but only with 172 neurons per layer. Exact numbers of parameters are given in Table 1.

Importance of the SIREN layers We choose a SIREN architecture for the transport embedding network. We show ablation scores in Table 2: we compare the baseline architecture with ReLU activations (design 1), to the baseline architecture with sine activations, as well as the PRT-inspired architecture with all ReLU activations, and the PRT-inspired architecture with sine activations in the transport MLP, ReLU in the second MLP (design 2). The results give evidence for the superiority of our final PRT-inspired design.

Table 2: *Validation loss in presence (S) or absence (R) of sine activations (SIREN) for the scenes* ATELIER *and* BEDROOM. *The third column proves that the PRT-Inspired architecture improves the predictions, even more so when using SIREN layers in the first MLP to encode high-frequency detail. Using SIREN in a naïve way (second column) performs worse than standard ReLU activations.*

Scene	ATELIER		BEDROOM	
	Loss	RMSE	Loss	RMSE
Baseline (R)	19.0	0.058	6.6	0.023
Baseline (S)	24.4	0.068	31.2	0.056
PRT-Inspired (R)	17.8	0.056	6.3	0.022
PRT-Inspired (S)	16.0	0.049	6.2	0.021

The inferior performance of an all-SIREN baseline architecture is caused by the concatenation of the light code and the transport code in the input. Mehta et al. [MGB*21] also show that conditioning SIRENs by concatenating embeddings to the input can produce surprisingly poor results (blurry images, no convergence on 3D shapes). In our case, this architecture (second row in Tab. 2) struggles on ATELIER and does not converge on BEDROOM.

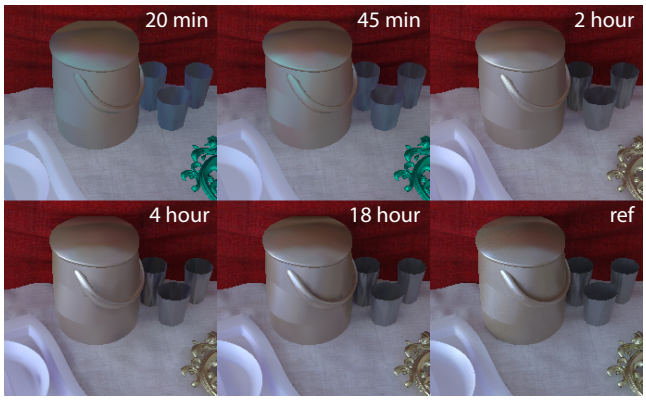


Figure 5: *Quality of the prediction (Diffuse-Specular architecture) at different times of training, compared to the reference. Note how the shadow of the handle appears after 4h. Scene:* ATELIER.

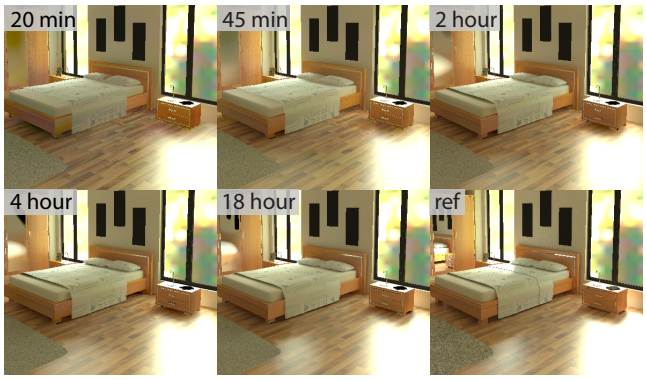


Figure 6: *Prediction quality (Diffuse-Specular architecture) during training. Note the progressive appearance of the second (left) reflection of the window on the wooden floor. Scene:* BEDROOM.

Training. We train for 500 epochs using the Adam optimizer with a learning rate of 10^{-4}. The networks are trained to minimize the L1 loss between the linear radiance data of the prediction and ground truth, after tonemapping by applying $\log(1+x)$. The effect of training duration can be seen in Fig. 5,6.

Timings. All inference timings are reported on an Intel Xeon Gold 5218R CPU with an NVIDIA RTX 3090 GPU. All data generation times are in Falcor; this is scene and sample-per-pixel dependent. On BEDROOM, 3000 training images at 256×256 resolution and 3200 samples per pixel requires 3.2 hours, while SANMIGUEL takes 4 hours. On our hardware cluster (NVIDIA RTX6000), our PyTorch implementation completes training in 16 to 18 hours. We will release our source code here to facilitate further research: https://repo-sam.inria.fr/fungraph/neural-prt.

In Tab. 3, we show the cost of inference for each of the four architectures, as well as the environment map encoder. We compare these timings with out-of-the-box timings of the Falcor real-time engine in Tab. 4. To make sure the comparison to any real-time

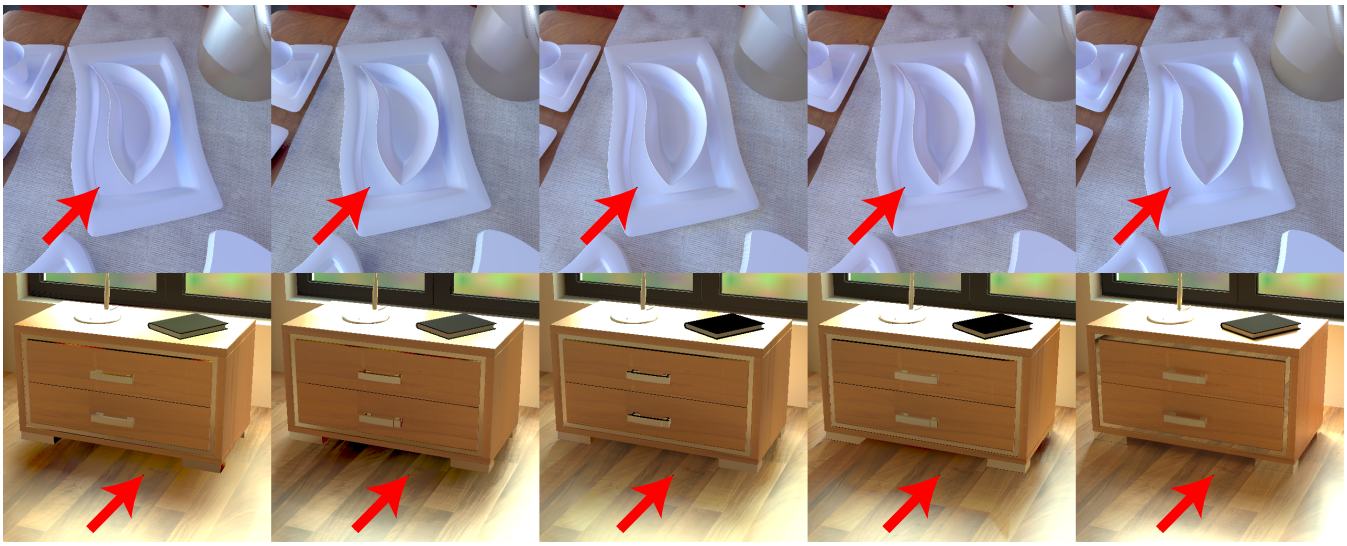


Figure 7: *Renderings of our proposed architectures on a close up of one frame from the test path of each scene, under unseen lighting.* ATELIER*: notice how the accuracy of interreflections and light bleeding between plates, as well as the overall hue, improve with each design.* BEDROOM*: the nightstand appears to be floating in the baseline model, the insertion in the scene improves with each design decision.*

Table 3: *Inference timings on a* 512×512 *image, per architecture, on a GeForce RTX 3090. These are upper bounds since the networks will not be run on background (environment map) pixels.* ***All competitors, using path-traced inputs, are given 5 spp (see Tab. 4), making them equal-speed or slower than our solutions.***

Encoder	Baseline	PRT-Insp.	Albedo Fact.	D/S Sep.
0.34 ms	9.73 ms	10.57 ms	10.60 ms	16.02 ms

Table 4: *Rendering timings of path tracing on a* 512×512 *image using the Falcor engine, evaluated on a GeForce RTX 3090. Different scenes will have slight variations in the timings due to the number of triangles, the amount of direct versus indirect light etc.*

	ATELIER	BEDROOM	KITCHEN	SANMIGUEL
G-Buffer	3 ms	3 ms	4 ms	8 ms
RTPT (5 spp)	19 ms	24 ms	27 ms	36 ms

method relying on path-traced inputs is fair, we allocate 5 samples per pixel to competitors in all subsequent figures and tables.

7. Results and Comparisons

We present results of our method and comparisons using four scenes of varying content and complexity: ATELIER (533K polygons), BEDROOM (1,499K polys), KITCHEN (1,443K polys) and SANMIGUEL (5,608K polys). The first, most simple scene was built from elements of different scenes, BEDROOM and KITCHEN are from Bitterli's resources [Bit16], SANMIGUEL is from McGuire's Computer Graphics Archive [McG17]. The three

Figure 8: *Full pathtraced reference views for the close-ups (marked by rectangles) shown in subsequent comparison figures. Videos of these camera paths rendered with each proposed method under unseen, dynamic lighting are in the supplemental material.*

last scenes show that our method can handle content of reasonable complexity.

We first show results on these four scenes for each architecture and the path-traced reference in the close-up images Fig. 7,9 (see Fig. 8 for the entire image for each scene allowing to situate the crops). We also show renderings of paths in the supplemental webpage that also contains comparisons, where the full images of each design can be evaluated in detail.

We next discuss the performance of the four alternative designs, followed by comparisons with previous rendering alternatives. It is important to note that these previous methods are the fruit of decades of research; the neural rendering algorithms we propose are a different and new paradigm. We thus do not claim to have systematically better performance or quality, but rather present conceptual comparisons and indicative quantitative and qualitative evaluation of our neural methods.

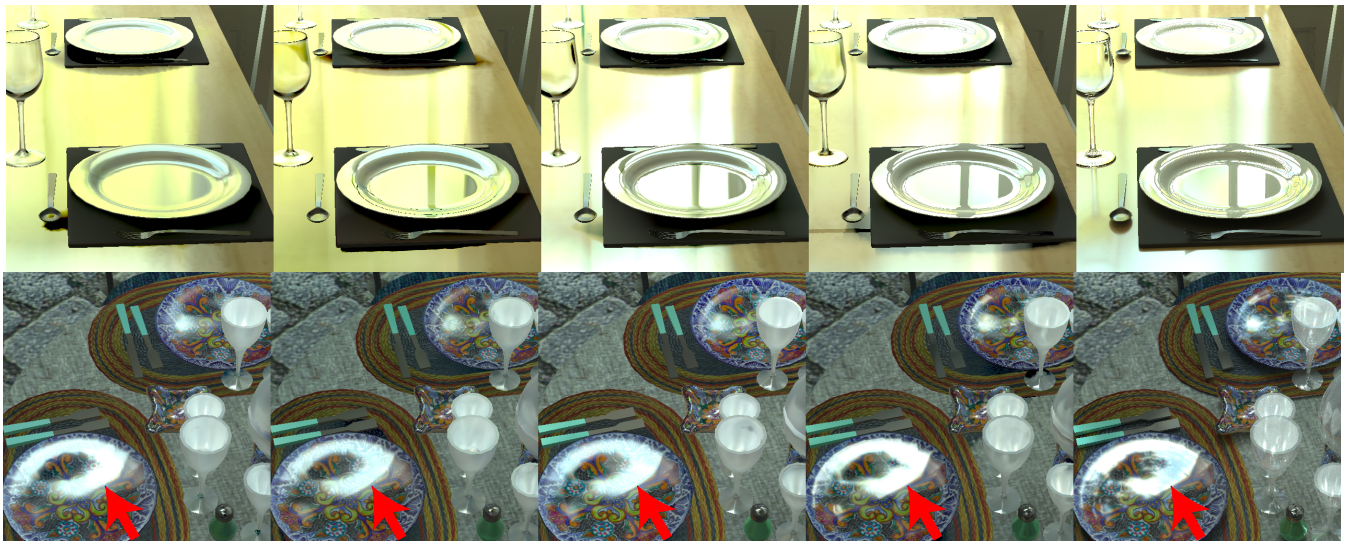


Figure 9: *Renderings of our proposed architectures on a close up of one frame from the test path of each scene, under unseen lighting.* KITCHEN*: the color of the reflection of the window on the table and the plate is correct only for the last two architectures; sharpness also improves.* SANMIGUEL*: highlight shape on both plates, and shadows between plates and table, improves in each design.*

7.1. Performance of the different models

The proposed model alternatives illustrate our overall goal of guiding neural rendering designs by well-founded principles of traditional rendering, in this case PRT.

The baseline transport operator has difficulties exploiting its full neural budget well, and struggles with very light- and view-dependent effects – highlights, shadows and interreflections are improved with the PRT-Inspired operator, and are closer to the reference with Albedo-Factorization and Diffuse-Specular separation (Fig. 7, 9). The baseline shows more temporal artifacts than the PRT-inspired solutions (please see videos in supplemental).

Table 5: *Reconstruction error* $\times 10^3$ *(Root Mean Square Error and log-loss used for training) of the proposed architectures, evaluated on the test images of each scene. Lowest (best) in bold.*

Scene	Baseline		PRT-Insp.		Albedo Fact.		Diff/Specular	
	Loss	RMSE	Loss	RMSE	Loss	RMSE	Loss	RMSE
ATELIER	18.95	57.6	15.96	49.4	14.70	47.9	**14.66**	**46.8**
BEDROOM	6.64	23.5	6.17	21.3	6.02	20.5	**5.50**	**20.0**
KITCHEN	3.15	16.6	2.79	15.0	2.69	15.1	**2.65**	**14.8**
SANMIGUEL	4.05	15.8	3.73	15.0	3.54	14.3	**3.51**	**13.9**

In the baseline design, all effects and inputs are entangled, and given a small neural budget, the network struggles to correctly learn full GI. In particular, it tends to bake strong shadows and highlights in place (see Fig. 7, ATELIER, reflection on the metal bucket in the top right). This issue is tackled by the PRT-Inspired design, where the first network has to learn a light-independent transport descriptor that only depends on the local point. This makes it much harder to bake direct lighting; and easier to bake effects like interreflections and ambient occlusion, which are not dependent on lighting (see Fig. 7, ATELIER). The second network predicts outgoing radiance given a transport descriptor and a light code of same dimensionality, so it can now aggregate them in a more balanced way.

The last two transfer-learning alternatives generally present more fidelity in the tint of the reflections and the shadows (see Fig. 9, KITCHEN). The essential difference with the previous architectures is that this factorization allows them to learn a radiance distribution which is not affected by the albedos, and hence much more coherent. Indeed, nearby points on a textured surface will have similar distributions of incoming light, with similar tints, even though their outgoing radiance will be affected by the albedo texture – this show in Fig. 7, BEDROOM, where the first two designs struggle to create a smooth shadow on the textured floor under the nightstand.

Finally, the complete separation of diffuse and specular tracks allows for a more efficient and physically-based disentanglement of view-dependent effects. The diffuse track focuses on extracting Lambertian reflectance and shadows, giving more importance to the positional input, while the specular track focuses on reflections, giving more importance to the directional input. The separation of the SIRENs allows them to weight the input importance differently, which albeit each at a lower neural budget, produces an overall result with better visual fidelity (see Fig. 9, SANMIGUEL, where the shape and intensity of the highlight on the plate matches the reference more closely). The last two architectures achieve better quality results, but this involves a trade-off since the improvement in quality comes with slower run-time inference (see Tab. 3).

We also show the quantitative effect of each design in Tab. 5. The Diffuse-Specular Separation is systematically best in all scenes,

both for the logarithmic loss used in training and Root Mean Square Error (RMSE). However, while the difference between Baseline and PRT-Inspired operators can be significant, the difference between the subsequent architectures is smaller.

7.2. Comparison to Previous Methods

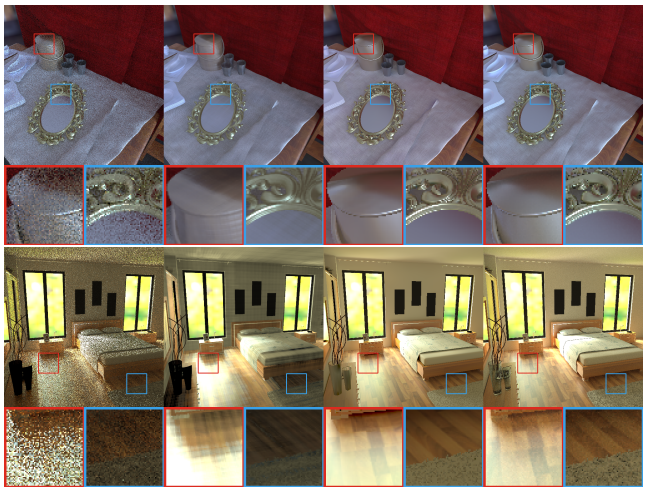
We show comparisons to real-time path-tracing in Falcor [BYC*20] (including denoising [CKS*17], [IMF*21]), and the very recent Neural Radiance Cache [MRNK21]. For these methods no precomputation is required; the comparisons are simply provided as an indication of quality that our method can provide. Note also that Falcor is a commercially-built and optimized system, while our solutions are experimental prototypes in a new design space. Finally we also show a baseline traditional PRT implementation, as an indication of the expressiveness of our neural representation compared to the traditional spherical harmonics.

RTPT RTPT+Denoiser Ours (D/S) Reference

Figure 10: *Left to right: Real-Time Path-Tracing (RTPT) with 5 spp, RTPT with Optix Denoiser (manual color correction has been applied to best match the reference), Our Diffuse-Specular Separation architecture and the ground truth.*

We first compare to a real-time path-tracing as implemented in the NVIDIA Falcor system, first with direct visualization and then with denoising. We provide a 5 sample-per-pixel budget in each scene, since this has equivalent cost to the inference step of our slowest solution, and also show images with the Optix denoiser available for Falcor [CKS*17]. We show results in Fig. 10 together with videos in supplemental. We see that the sample budget is simply insufficient to capture most of the transfer, and the denoiser can only provide blurry results at this sampling rate. In addition, the denoiser has significant temporal artifacts. We also tested a more recent denoiser [IMF*21] (the authors kindly ran their method on our scene), but scene conversion issues to another rendering engine result in slightly different images. We show this result in the supplemental webpage: we see that the quality is better, but the sample-per-pixel rate is still too low, and there are still significant temporal and spatial artifacts. Additionally, the denoising adds some overhead to the pathtracing which causes slower framerates; the Optix denoiser for instance runs in 10ms on a 512×512 frame.

We next show results for the Neural Radiance Cache [MRNK21]; the authors kindly ran their code on two



NRC NRC-cache Ours (D/S) Reference

Figure 11: *Comparison to the Neural Radiance Cache [MRNK21].*

of our scenes. We show both the final result of their method, using a 5 sample-per-pixel (spp) path-tracing and cache queries at deeper bounces, as well as a direct visualization of the cache at the first bounce, which is similar to the setting our neural network operates in. The former is noisy, since the sample budget is low, and the latter does not represent indirect light accurately, since the cache is built based on rays cast in earlier frames. When the lighting changes or the camera moves, the cache carries bias from the previously seen rays. Additionally, the artifacts of positional encoding are very visible, in the form of cross-shape artifacts.

CNN MLP CNN MLP

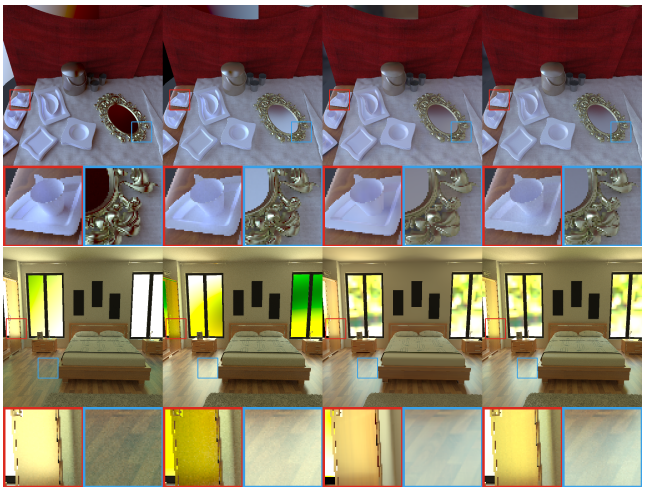
Figure 12: *Comparison of MLP (our naïve architecture) versus CNN [NAM*17], both trained per-scene at equal parameter count. Scenes:* ATELIER, BEDROOM. *Note how the U-Net visibly struggles with specular/glossy appearance. The lack of temporal coherence can be observed in the supplemental material videos.*

Regarding learning-based methods, we evaluate the performance of a per-scene trained Deep Shading [NAM*17] network with similar parameter count. In Figure 12, we compare Deep Shading's U-Net (a convolutional network) against our most naïve baseline architecture (a basic MLP), which already achieves a drastic increase in quality. CNNs learn to exploit the local G-buffer context in order to generalize plausibly to unseen geometries. Since we overfit to a single scene, treating the G-buffer context instead of an individual

Table 6: *Quantitative comparison of our method with close competitors. Metrics ($\times 10^3$) averaged over the frames of the paths shown in the supplemental material videos. For PRT, the error was only computed on the single frame shown in Figure 13.*

	ATELIER					BEDROOM				
	L_1	L_2	dSSIM	MAPE	MRSE	L_1	L_2	dSSIM	MAPE	MRSE
RTPT	124.21	71.20	597.92	328.70	230.02	49.48	13.22	579.23	683.72	511.64
RTPT+DENOISE	293.65	182.57	686.87	689.80	453.46	151.18	45.24	519.14	1889.00	1526.00
NRC	98.19	34.29	533.98	268.22	137.64	35.21	11.17	400.60	442.90	164.90
NRC-CACHE	68.75	15.21	245.93	208.58	72.22	26.21	3.36	237.2	366.50	77.44
DEEP SHADING	51.39	15.75	172.61	175.06	39.11	16.15	3.91	89.38	219.37	47.19
PRT-9	64.86	19.18	147.93	171.43	72.53	14.72	3.17	84.97	172.72	47.64
PRT-25	36.83	11.41	88.27	81.74	29.99	12.63	7.71	55.65	117.85	**19.62**
OURS (D/S)	**30.23**	**5.34**	**63.95**	**80.46**	**22.02**	**9.89**	**2.09**	**36.95**	**100.20**	21.76

pixel is a waste of capacity that is better used to learn view- and light-dependent effects, hence our choice of MLP.



PRT-9 PRT-25 Ours (D/S) Reference

Figure 13: *Comparison of PRT-9 (SH order 2) and PRT-25 (order 4) with our Diffuse-Specular. Scenes:* ATELIER, BEDROOM. *Note how the intensity of reflections is not reproduced correctly by PRT.*

We also compare to a baseline PRT re-implementation. Transfer matrices are usually stored at the vertices – we implement a version with SH-coded transport (distant to outgoing), stored at every pixel in a given fixed view. Specifically, we project the environment map to a spherical harmonic (SH) basis, and we precompute full transport for each pixel encoded with the same SH basis. A full implementation of Sloan et al. [SKS02] would precompute at vertices, resulting in significantly more blurred results. We present this comparison as an indication of the quality obtained by our method compared to PRT. In Fig. 13, we show results with 9 and with 25 coefficients (per pixel per color channel). We see that even with 25 coefficients the SH basis over- or under-estimates global illumination, compared to our solution. In terms of memory, the size of our network would equate to less than 1 coefficient per pixel.

We show numerical comparisons of all methods in Tab. 6. The quantitative results show that our method outperforms alternative solutions across the board.

7.3. Limitations

Ours (D/S) Reference

Figure 14: *Rendering of a partly unseen (at training time) area of* BEDROOM *with challenging materials (perfect mirrors). In the seen part (right side), the prediction is good even in the mirror. In the unseen part (left side), the degradation is graceful for diffuse objects and some artifacts appear in the wall mirror.*

Overall, our PRT-inspired neural renderers represent aggregate effects well, including complex indirect paths. As in any learning-based method, our solutions offer a compromise of prediction fidelity based on the frequency of observations in the training data and their weighting. The correlation (R) of our diffuse-specular prediction error with *directness* (ratio of direct over total incoming light) across test pixels is R = 0.24 for ATELIER and 0.34 for BEDROOM. The correlation of error with *specularity* (ratio of specular over specular+diffuse albedos) gave R = 0.14 for ATELIER and -0.38 for BEDROOM, which indicates that there is no meaningful relationship: The neural renderer must simply have seen sufficient training examples of the part of the scene we want to render under similar conditions (see Fig. 14), which can be addressed through more involved camera placement strategies. Angular details of high-frequency reflections or harsh direct shadows from very localized light sources can be reproduced faithfully, as long as they are observed frequently enough in the training data. This

limitation can easily be overcome by augmenting the G-Buffer inputs with limited one-bounce ray-tracing. Nonetheless, in sparsely observed regions (limits of the camera volume or the scene), the prediction degrades gracefully (Fig. 14), relying on the similarity to observed parts of the scene.

Another constraint is the trade-off between how small the MLP can be and how accurately it models every part of the scene. In this paper we investigate the design space of the architecture so we fix the size of the model and produce training sets that cover reasonable viewpoints. To appreciate the extent and variation of viewpoint and lighting, please see the supplemental videos. Scaling to very large scenes can be addressed in future work by adaptively subdividing the scene, as done by Ren et al. [RWG*13] or in the recent KiloNeRF [RPLG21]. One could also imagine that instead of taking only a point's attributes, the transport encoder could take a learnt embedding of a local intermediate-scale part of a large scene.

8. Conclusions

Designing neural renderers will be an important challenge for rendering as the field moves forward. We claim that building on the wealth of traditional rendering research will be beneficial in this process, and lead to efficient solutions.

We illustrated this claim by investigating four different neural rendering architectures for the case of static scenes with dynamic lighting. We started with a direct radiance-regressing baseline, and then demonstrated that using the theoretical foundations and algorithmic principles of PRT leads to progressively better image quality at equal computation and memory, essentially using the neural budget more efficiently. In particular, we demonstrated that learning a separate transport and light code improves the overall estimation of global illumination, e.g., indirect lighting and glossy reflections. We then applied another PRT principle, factorizing albedo, and a final design where diffuse and specular terms are separated. As in PRT, these two solutions learn a transformation more akin to *transfer* than transport. These changes permit more accurate reconstruction of shadows and/or indirect light.

In future work, we plan to treat the case of static illumination or view more efficiently. In particular, using a cache, we could avoid re-evaluating the CNN or the MLPs respectively. For the final architecture, diffuse irradiance could also be cached if the illumination does not vary. Another avenue could contain more elaborate architectures for the environment map encoder, to be robust to the spherical distortion induced by the latlong projection or to aliasing which can create temporal flickering. A more involved direction for future work would be for dynamic/deforming scenes: similarly to the dynamic lighting we treated, information about the dynamic scene (e.g., changing materials) could be encoded into a learned basis which could modulate the outgoing radiance.

A final improvement could tackle anti-aliasing: In our method we sample each pixel by a unique central ray, in order to be truly agnostic to rendering resolution / viewing distance and objectively compare representational power of different architectures without interference of filtering. Any anti-aliasing strategy can be applied orthogonally, such as averaging predictions per pixel, or running an anti-aliasing filter in post-process.

In conclusion, we hope that the proposed methodology will inspire further work, expanding the design space in neural rendering based on well-founded theoretical and practical results from traditional rendering research, in particular for different hardware platforms that may not have specific ray-tracing acceleration available.

Acknowledgments

This research was funded by the ERC Advanced grant FUNGRAPH No 788065 (http://fungraph.inria.fr). The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support. The authors would also like to thank Adobe for their generous donations, and acknowledge Fabrice Rousselle for helping with the comparison to NRC by running the code on our scenes.

References

[BBJ*21] Boss M., Braun R., Jampani V., Barron J. T., Liu C., Lensch H. P.: Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)* (2021). 2

[BGP*21] Baatz H., Granskog J., Papas M., Rousselle F., Novák J.: Nerf-tex: Neural reflectance field textures. *Computer Graphics Forum (Proc. EGSR) 40*, 4 (2021). 2

[Bit16] Bitterli B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 8

[BMSR20] Bemana M., Myszkowski K., Seidel H.-P., Ritschel T.: X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020) 39*, 6 (2020). 1

[Bur20] Burgess J.: RTX on the NVIDIA Turing GPU. *IEEE Micro 40*, 2 (2020), 36–44. 1

[BXS*20] 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. 2

[BYC*20] Benty N., Yao K.-H., Clarberg P., Chen L., Kallweit S., Foley T., Oakes M., Lavelle C., Wyman C.: The Falcor rendering framework, 08 2020. 6, 10

[CDAS20] Currius R. R., Dolonius D., Assarsson U., Sintorn E.: Spherical gaussian light-field textures for fast precomputed global illumination. *Computer Graphics Forum 39*, 2 (2020), 133–146. 3

[CKS*17] Chaitanya C. R. A., Kaplanyan A. S., Schied C., Salvi M., Lefohn A., Nowrouzezahrai D., Aila T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–12. 10

[DHT*] Debevec P., Hawkins T., Tchou C., Duiker H.-P., Sarokin W., Sagar M.: Acquiring the reflectance field of a human face. SIGGRAPH 2000, ACM., p. 145–156. 2

[ERB*18] Eslami S. A., Rezende D. J., Besse F., Viola F., Morcos A. S., Garnelo M., Ruderman A., Rusu A. A., Danihelka I., Gregor K., et al.: Neural scene representation and rendering. *Science 360*, 6394 (2018), 1204–1210. 1, 3

[GCD*20] Gao D., Chen G., Dong Y., Peers P., Xu K., Tong X.: Deferred neural lighting: Free-viewpoint relighting from unstructured photographs. *ACM Trans. Graph. 39*, 6 (Nov. 2020). 2

[GKMD] Green P., Kautz J., Matusik W., Durand F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. I3D 2006, ACM, p. 7–14. 3

[GRPN20] GRANSKOG J., ROUSSELLE F., PAPAS M., NOVÁK J.: Compositional neural scene representations for shading inference. *ACM Trans. Graph. 39*, 4 (July 2020). 1, 3

[GSY*17] GARDNER M.-A., SUNKAVALLI K., YUMER E., SHEN X., GAMBARETTO E., GAGNÉ C., LALONDE J.-F.: Learning to predict indoor illumination from a single image. *ACM Trans. Graph. 36*, 6 (Nov. 2017). 6

[HCZ21] HADADAN S., CHEN S., ZWICKER M.: Neural radiosity. *CoRR abs/2105.12319* (2021). 3

[HGAL19] HOLD-GEOFFROY Y., ATHAWALE A., LALONDE J.-F.: Deep sky modeling for single image outdoor lighting estimation. In *IEEE/CVF Conf. on Comp. Vis. & Patt. Recog.(CVPR)* (2019). 6

[HY21] HUO Y., YOON S.-E.: A survey on deep learning-based monte carlo denoising. *Computational Visual Media 7*, 2 (2021), 169–185. 3

[IMF*21] IŞIK M., MULLIA K., FISHER M., EISENMANN J., GHARBI M.: Interactive monte carlo denoising using affinity of neural features. *ACM Trans. Graph. 40*, 4 (July 2021). 10

[JK21] JIANG G., KAINZ B.: Deep radiance caching: Convolutional autoencoders deeper in ray tracing. *Computers & Graphics 94* (2021), 22–31. 3

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 143–150. 4

[KHX*19] KUZNETSOV A., HAŠAN M., XU Z., YAN L.-Q., WALTER B., KALANTARI N. K., MARSCHNER S., RAMAMOORTHI R.: Learning generative models for rendering specular microgeometry. *ACM Trans. Graph. 38*, 6 (Nov. 2019). 2

[KMX*21] KUZNETSOV A., MULLIA K., XU Z., HAŠAN M., RAMAMOORTHI R.: NeuMIP: Multi-resolution neural materials. *ACM Trans. Graph. 40*, 4 (July 2021). 2

[Leh07] LEHTINEN J.: A framework for precomputed and captured light transport. *ACM Trans. Graph. 26*, 4 (Oct. 2007), 13–es. 4

[LSY*21] LAGUNAS M., SUN X., YANG J., VILLEGAS R., ZHANG J., SHU Z., MASIA B., GUTIERREZ D.: Single-image full-body human relighting. In *Eurographics Symposium on Rendering (EGSR)* (2021). 3

[LWM19] LI Y., WIEDEMANN P., MITCHELL K.: Deep precomputed radiance transfer for deformable objects. *Proc. ACM Comput. Graph. Interact. Tech. 2*, 1 (June 2019). 3

[McG17] MCGUIRE M.: Computer graphics archive, 2017. https://casual-effects.com/data. 8

[MGB*21] MEHTA I., GHARBI M., BARNES C., SHECHTMAN E., RAMAMOORTHI R., CHANDRAKER M.: Modulated periodic activations for generalizable local functional representations, 2021. 7

[MLL*21] MARTEL J. N. P., LINDELL D. B., LIN C. Z., CHAN E. R., MONTEIRO M., WETZSTEIN G.: Acorn: Adaptive coordinate networks for neural scene representation. *ACM Trans. Graph. 40*, 4 (July 2021). 2

[MRLTF19] MAXIMOV M., RITSCHEL T., LEAL-TAIXE L., FRITZ M.: Deep appearance maps. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 8728–8737. 2

[MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Trans. Graph. 40*, 4 (July 2021). 1, 3, 10

[MSOC*19] MILDENHALL B., SRINIVASAN P. P., ORTIZ-CAYON R., KALANTARI N. K., RAMAMOORTHI R., NG R., KAR A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* (2019). 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 1, 2

[NAM*17] NALBACH O., ARABADZHIYSKA E., MEHTA D., H-P. S., RITSCHEL T.: Deep shading: Convolutional neural networks for screen space shading. *Computer Graphics Forum (Proc. EGSR) 36*, 4 (2017). 1, 3, 10

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 477–487. 3

[NVI18] NVIDIA: NVIDIA Turing GPU architecture: Graphics reinvented, 2018. 1

[PEL*21] PANDEY R., ESCOLANO S. O., LEGENDRE C., HÄNE C., BOUAZIZ S., RHEMANN C., DEBEVEC P., FANELLO S.: Total relighting: Learning to relight portraits for background replacement. *ACM Trans. Graph. 40*, 4 (July 2021). 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 2

[PMGD21] PHILIP J., MORGENTHALER S., GHARBI M., DRETTAKIS G.: Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Transactions on Graphics 40*, 5 (2021). 2

[PWXLPB07] PAN M., WANG XINGUO LIU R., PENG Q., BAO H.: Precomputed radiance transfer field for rendering interreflections in dynamic scenes. *Computer Graphics Forum 26*, 3 (2007), 485–493. 3

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Comput. Graph. Forum 31*, 1 (Feb. 2012), 160–188. 1, 3

[RDL*15] REN P., DONG Y., LIN S., TONG X., GUO B.: Image based relighting using neural networks. *ACM Trans. Graph. 34*, 4 (jul 2015). 3

[RGJW20] RAINER G., GHOSH A., JAKOB W., WEYRICH T.: Unified neural encoding of BTFs. *Computer Graphics Forum (Proc. Eurographics) 39*, 2 (July 2020), 167–178. 2

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural BTF compression and interpolation. *Computer Graphics Forum (Proc. Eurographics) 38*, 2 (May 2019), 235–244. 1, 2

[RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. 12

[RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Trans. Graph. 32*, 4 (July 2013). 2, 12

[SDZ*21] 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. In *CVPR* (2021). 2

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. In *ACM SIGGRAPH* (July 2003), ACM. 3

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (July 2002), 527–536. 3, 4, 11

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. SIGGRAPH 2005, ACM, p. 1216–1224. 3

[SMB*20] SITZMANN V., MARTEL J. N., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Proc. NeurIPS* (2020). 2, 6

[SRRW21] SZTRAJMAN A., RAINER G., RITSCHEL T., WEYRICH T.: Neural BRDF representation and importance sampling. *Computer Graphics Forum* (2021). 2

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic BRDFs. *ACM Trans. Graph. 26*, 3 (July 2007), 27–es. 3

[TFT*20] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., SUNKAVALLI K., MARTIN-BRUALLA R., SIMON T., SARAGIH J., NIESSNER M., PANDEY R., FANELLO S., WETZSTEIN G., ZHU J.-Y., THEOBALT C., AGRAWALA M., SHECHTMAN E., GOLDMAN D. B., ZOLLHOFER M.: State of the art on neural rendering. *Computer Graphics Forum 39*, 2 (2020), 701–727. 2

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25*, 3 (July 2006), 967–976. 3

[TZN19] THIES J., ZOLLHÖFER M., NIESSNER M.: Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph. 38*, 4 (July 2019). 2

[WCZR20] WU L., CAI G., ZHAO S., RAMAMOORTHI R.: Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. *ACM Trans. Graph. 39*, 4 (jul 2020). 3

[WPYS21] WIZADWONGSA S., PHONGTHAWEE P., YENPHRAPHAI J., SUWAJANAKORN S.: Nex: Real-time view synthesis with neural basis expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021). 3

[WR18] WANG J., RAMAMOORTHI R.: Analytic spherical harmonic coefficients for polygonal area lights. *ACM Trans. Graph. 37*, 4 (July 2018). 3

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 1–10. 3

[XCM*14] XU K., CAO Y.-P., MA L.-Q., DONG Z., WANG R., HU S.-M.: A practical algorithm for rendering interreflections with all-frequency brdfs. *ACM Trans. Graph. 33*, 1 (Feb. 2014). 3

[XSD*13] XU K., SUN W.-L., DONG Z., ZHAO D.-Y., WU R.-D., HU S.-M.: Anisotropic spherical gaussians. *ACM Trans. Graph. 32*, 6 (Nov. 2013). 3

[XSHR18] XU Z., SUNKAVALLI K., HADAP S., RAMAMOORTHI R.: Deep image-based relighting from optimal sparse samples. *ACM Trans. Graph. 37*, 4 (jul 2018). 3

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: PlenOctrees for real-time rendering of neural radiance fields. In *ICCV* (2021). 3

[ZBX*21] ZHU J., BAI Y., XU Z., BAKO S., VELÁZQUEZ-ARMENDÁRIZ E., WANG L., SEN P., HAŠAN M., YAN L.-Q.: Neural complex luminaires: Representation and rendering. *ACM Trans. Graph. 40*, 4 (July 2021). 1, 2

[ZFT*21] ZHANG X., FANELLO S., TSAI Y.-T., SUN T., XUE T., PANDEY R., ORTS-ESCOLANO S., DAVIDSON P., RHEMANN C., DEBEVEC P., BARRON J. T., RAMAMOORTHI R., FREEMAN W. T.: Neural light transport for relighting and view synthesis. *ACM Trans. Graph. 40*, 1 (Jan. 2021). 2

[ZLW*21] ZHANG K., LUAN F., WANG Q., BALA K., SNAVELY N.: Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021). 2

[ZSD*21] 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. *arXiv preprint arXiv:2106.01970* (2021). 2

[ZTF*18] ZHOU T., TUCKER R., FLYNN J., FYFFE G., SNAVELY N.: Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. 37*, 4 (July 2018). 2

[ZTS*16] ZHOU T., TULSIANI S., SUN W., MALIK J., EFROS A. A.: View synthesis by appearance flow. In *European Conference on Computer Vision* (2016). 2