

Neural Free-Viewpoint Relighting for Glossy Indirect Illumination

Nithin Raghavan,^{1*} Yan Xiao,^{1*} Kai-En Lin,¹ Tiancheng Sun,² Sai Bi,³ Zexiang Xu,³ Tzu-Mao Li,¹ Ravi Ramamoorthi¹

¹University of California, San Diego ²Google ³Adobe Research

* denotes equal contribution

Abstract

Precomputed Radiance Transfer (PRT) remains an attractive solution for real-time rendering of complex light transport effects such as glossy global illumination. After precomputation, we can relight the scene with new environment maps while changing viewpoint in real-time. However, practical PRT methods are usually limited to low-frequency spherical harmonic lighting. All-frequency techniques using wavelets are promising but have so far had little practical impact. The curse of dimensionality and much higher data requirements have typically limited them to relighting with fixed view or only direct lighting with triple product integrals. In this paper, we demonstrate a hybrid neural-wavelet PRT solution to high-frequency indirect illumination, including glossy reflection, for relighting with changing view. Specifically, we seek to represent the light transport function in the Haar wavelet basis. For global illumination, we learn the wavelet transport using a small multi-layer perceptron (MLP) applied to a feature field as a function of spatial location and wavelet index, with reflected direction and material parameters being other MLP inputs. We optimize/learn the feature field (compactly represented by a tensor decomposition) and MLP parameters from multiple images of the scene under different lighting and viewing conditions. We demonstrate real-time (512 x 512 at 24 FPS, 800 x 600 at 13 FPS) precomputed rendering of challenging scenes involving view-dependent reflections and even caustics.

CCS Concepts

- Computing methodologies → Reflectance modeling;

1. Introduction

Interactive rendering of scenes with complex global illumination effects remains a long-standing challenge in computer graphics. Precomputed Radiance Transfer (PRT) [SKS02], which enables interactive relighting by precomputing the light transport of a static scene, remains an attractive solution. However, the practical impact of PRT has largely been limited to low-frequency spherical harmonic methods. All-frequency methods using Haar wavelets were proposed to address this shortcoming, but required substantially larger data storage, and were therefore limited to fixed viewpoint [NRH03], triple products for direct lighting only [NRH04] or lower-frequency BRDF in-out factorizations [LSS04, WTL06]. Obtaining true all-frequency relighting with changing view-dependent glossy global illumination effects requires precomputing, storing and rendering with a high-resolution 6D light transport tensor for spatial, light and view variation, which has remained intractable because of the exponential growth in data size with dimensionality.

With the advent of deep learning and implicit neural representations, we have a new mathematical toolbox of function approximators that can be used to revisit this challenge. Indeed, work on neural radiance fields [MST*20, TSM*20] showed that high-dimensional spatio-angular radiance functions can be learned by a simple multi-layer perceptron (MLP), and these ideas have been ap-

plied to directly solve the rendering equation with neural function representations [HCZ21]. However, simply approximating the light transport matrix in a neural basis is insufficient for PRT, since one needs to compute light transport integrals in real-time as is done in spherical harmonics or wavelets.

In this paper, we leverage the seminal early PRT work, modern MLP-based function approximators and recent rendering advances to tackle these problems. We focus on indirect lighting, including glossy view-dependent global illumination. Several approaches to real-time direct lighting exist, including the original triple product formulation [NRH04], and ReSTIR [BWP*20]. We leverage real-time raytracing in OptiX on modern RTX graphics cards [NVI18b, PBD*10] with importance sampling of the environment map and Monte Carlo denoising [ZJL*15] in OptiX [NVI18a]. However, such a direct path-tracing approach is still not real-time for complex light transport paths involving multi-bounce glossy indirect reflections or caustic patterns. Our major technical contributions and design decisions include:

Haar Wavelet Representation: As in the original wavelet-based PRT algorithms, we seek to project the lighting and light transport into Haar wavelets, while keeping a small number (typically 64) of the most important lighting coefficients. This enables a real-time wavelet dot product and projection of the environment map as in previous work, and differs from recent neural PRT ap-

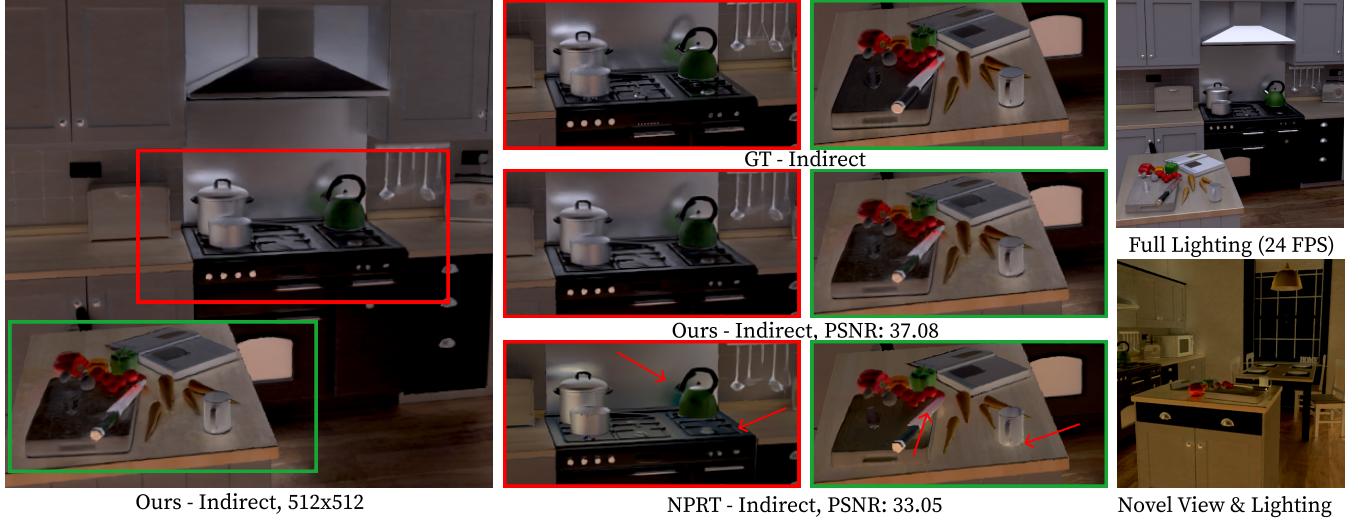


Figure 1: We develop a precomputed radiance transfer (PRT) method, based on a hybrid neural-wavelet representation. Our method enables high-frequency relighting with changing view and glossy indirect illumination. Left: Indirect illumination (which our method focuses on) rendered at 24 FPS on an RTX 4090 with our system using 64 Haar wavelets for the environment map and our learned MLP light transport. Middle: Comparison to Neural PRT [RBRD22] and ground truth. Neural PRT does not handle high-frequency view-dependent effects as well as our method (notice the missing glossy reflections pointed out by the arrows), and has a slight tonal shift on the stove and the pots. Right: Showing the rendering combined with direct lighting, and different lighting environments and views of the same scene rendered in real-time.

proaches [RBRD22, XZW^{*}22], which require separate neural layers to compute dot products within neural functions. While Rainer et al.’s method [RBRD22] is suitable for largely diffuse scenes, the quality of indirect view-dependent effects is often less accurate. Their neural approximation of the linear dot product can also lead to a tonal shift of the image. By working directly with wavelets, our approach better preserves high-frequency effects such as glossy reflections, and it has the theoretical benefits in remaining grounded in linear function spaces (see Fig. 1 and quantitative comparisons in Table 1).

Light Transport Representation: The key challenge is the representation of the light transport coefficient for a given view, spatial location and wavelet index. For direct lighting, the 6D light transport can be factorized into a product of 4D view-independent visibility and 4D BRDF functions, with wavelet coefficients of the product computed using triple products [NRH04]. However, it is not possible to extend this formulation to global illumination. We make two important modifications, enabled by modern MLP-based learning algorithms. First, instead of visibility, we learn a feature vector parameterized by spatial location and wavelet index. To enable compact storage and fast training and evaluation, we decompose the feature field with tensor factorization [CXG^{*}22], where the 3D spatial component is represented using a multiresolution hash grid [MESK22]. To our knowledge, this is the first method to combine the use of tensor factorization and multiresolution hash grids. Finally, we use a small MLP that takes as input the feature vector, reflection direction, normal, and BRDF parameters, and outputs the transport coefficients. The MLP and feature field tensors

are all trained on images of the scene under different views and environment maps.

Real-Time Rendering: We demonstrate real-time rendering with precomputed light transport, including glossy effects with changing lighting and view. The size of our final representation is compact (113 MB for the scene in Fig. 1), significantly smaller than an explicit 6D representation, and even smaller than early wavelet-based relighting methods without view-dependence [NRH03]. We believe our work addresses a long unresolved challenge in PRT methods, to enable high-frequency lighting and viewpoint variation with global illumination (see Fig. 1, rendered at 24fps), while giving a new capability to real-time rendering.

2. Related Work

PRT research has always relied on new mathematical representations beyond spherical harmonics and wavelets, such as zonal harmonics [SLS05], clustered principal components (CPCA) [SHHS03], spherical Gaussians with tensor approximation [TS06], and von-Mises Fisher approximations of the transfer function [LWLD11]. Our work can be seen as a natural progression along this line involving MLP-based neural function approximators. We limit ourselves to the standard PRT setting of static scenes with distant environment map illumination, and do not consider near-field area sources [MSW04], or dynamic objects [ZHL^{*}05]. We are distinct from direct-to-indirect transfer methods [HPB06, BDBS22], which cannot easily handle complex view-dependent global illumination. We do take inspiration from them in handling direct lighting separately. We refer readers to the comprehensive

survey by Ramamoorthi [Ram09], which points out the unsolved nature of all-frequency relighting with changing viewpoint and glossy objects. They also note that triple products [NRH04] are limited by the inability to support spatial compression (CPCA), while BRDF factorization methods [LSSS04, WTL06] can require more than a hundred terms for high-frequency materials [MTR08].

Ren et al. [RDL*15, RWG*13] first introduced the use of neural networks to regress global illumination and image-based relighting. In contrast, we focus on the classic PRT problem with environment maps, and introduce a novel light transport representation. Most recently, Rainer et al. [RBRD22] introduced a neural PRT solution with diffuse-specular separation. They do not directly use wavelets, unlike our method, but use a convolutional neural network to extract lighting features. In contrast, our method is a novel hybrid using neural networks to predict Haar wavelet transport coefficients, and we demonstrate better glossy effects in our results, and better quantitative metrics (see Table 1 in results).

Xu et al. [XZW*22] introduce lightweight neural basis functions, and neural networks for double and triple product integrals. As with most neural bases, there is no guarantee of orthonormality and a separate network is needed for the dot products. In contrast, we leverage standard orthonormality and approximation with the most significant coefficients by performing dot products in wavelets, while using neural networks only to represent wavelet transport coefficients. Moreover, Xu et al. only demonstrate fixed view, and the inherent limitations of triple product integrals require a restriction to direct lighting. Our work also relates to research on neural materials and layering [FWH*22] and recent efforts in acquisition of light transport from real scenes [LTL*22] but we have very different goals.

We acknowledge the significant recent progress in real-time path tracing and denoising [SKW*17, CKS*17] without the need for any precomputation. A comprehensive discussion of these methods is out of scope, and they are largely orthogonal to our PRT-based approach. We do note that they are usually still limited in capturing complex multi-bounce light transport like glossy reflections at the low sample counts required for real-time applications. We do leverage this research by denoising the direct lighting. Although our PRT indirect renderings are of high quality, and not affected by Monte Carlo noise in the traditional sense, we do observe a small benefit from denoising, see Table 1 and Fig. 8.

3. Overview

We now provide a brief overview of our method. The light transport equation is given by,

$$B(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\Omega} T(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o) L(\boldsymbol{\omega}) d\boldsymbol{\omega}, \quad (1)$$

where B is the outgoing radiance we seek, as a function of surface position \mathbf{x} , and outgoing direction $\boldsymbol{\omega}_o$. It is given by an integral of the environment map lighting L , a function of incident direction $\boldsymbol{\omega}$, multiplied by the light transport function T , which is a function of spatial location \mathbf{x} and incident and outgoing angles $(\boldsymbol{\omega}, \boldsymbol{\omega}_o)$. For our purposes T will represent only the global illumination component, with direct lighting computed separately.

In PRT, we precompute the light transport T and dynamically

change the lighting L and view $\boldsymbol{\omega}_o$. We follow previous PRT methods by projecting the lighting (at run-time) and transport (precomputed) into a suitable basis—Haar wavelets on cubemap faces as in previous work [NRH03]

$$\begin{aligned} L(\boldsymbol{\omega}) &= \sum_j L_j \Psi_j(\boldsymbol{\omega}) \\ T(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o) &= \sum_k T_k(\mathbf{x}, \boldsymbol{\omega}_o) \Psi_k(\boldsymbol{\omega}), \end{aligned} \quad (2)$$

where Ψ_k are the basis functions indexed by k , and L_k and T_k are the lighting and transport coefficients. The basis expansion in T is only over the incident direction $\boldsymbol{\omega}$ which is being integrated. We achieve real-time rendering simply by taking the dot-product,

$$\begin{aligned} B(\mathbf{x}, \boldsymbol{\omega}_o) &= \sum_j \sum_k L_j T_k(\mathbf{x}, \boldsymbol{\omega}_o) \int_{\Omega} \Psi_j(\boldsymbol{\omega}) \Psi_k(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &= \sum_{k \in K} L_k T_k(\mathbf{x}, \boldsymbol{\omega}_o) \\ &= \mathbf{L} \cdot \mathbf{T}(\mathbf{x}, \boldsymbol{\omega}_o), \end{aligned} \quad (3)$$

where \mathbf{L} and \mathbf{T} represent vectors of all coefficients k , and the integral simplifies by the orthonormality of basis functions. This simplicity and the resulting practicality for real-time rendering is not possible when using a (non-orthonormal) neural basis as in earlier work [RBRD22, XZW*22]. These works therefore require a separate more complex network to perform approximate integration/dot products. Efficiency in the summation or dot-product is obtained by considering only a set K of the largest wavelet coefficients in the lighting (we typically use 64); this is indicated in the second line above. The entire transport T must still be precomputed, but only the coefficients in K will be used (this set can change at each frame with the lighting).

It remains to compute and represent T and T_k . As motivation, we first review the triple product approach used for direct lighting [NRH04]. In that case, the transport is simply the point-wise product of (view-independent) visibility V and cosine-weighted BRDF ρ , with wavelet coefficients computed using triple products,

$$\begin{aligned} T^d(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o) &= V(\mathbf{x}, \boldsymbol{\omega}) \rho(\boldsymbol{\omega}, \boldsymbol{\omega}_o) \\ T_k^d(\mathbf{x}, \boldsymbol{\omega}_o) &= \sum_i \sum_j C_{ijk} V_i(\mathbf{x}) \rho_j(\boldsymbol{\omega}_o) \\ B^d(\mathbf{x}, \boldsymbol{\omega}_o) = \sum_k L_k T_k^d(\mathbf{x}, \boldsymbol{\omega}_o) &= \sum_i \sum_j \sum_k C_{ijk} V_i(\mathbf{x}) \rho_j(\boldsymbol{\omega}_o) L_k, \end{aligned}$$

where C_{ijk} are the tripling coefficients and we use the superscript d to specify this is for direct lighting only (these equations are not used in our system; they are for illustration and motivation only). Note that the original triple product method directly used the integration with the lighting (last line above) without explicitly forming the transport coefficient above, but this formulation is equivalent.

For global illumination, no such simple form exists and we will represent $T_k(\mathbf{x}, \boldsymbol{\omega}_o)$ instead by a neural network. However, we are inspired by the formulation above and modify it in two key ways. First, as there is no closed-form expression for the convolution of visibility terms for an arbitrary number of ray bounces, we replace the visibility in the above formulation with a view-independent general feature vector, which is a function of output wavelet coefficient k and spatial position \mathbf{x} . This promotes a compact fac-

torization of light transport that allows the network to learn these terms. Second, we replace the simple multiplication of visibility and BRDF (and related triple product wavelet formulation) by a small multi-layer perceptron (MLP) that takes as input the feature vector, surface normal, reflected direction and BRDF parameters (diffuse and specular coefficients, roughness) and outputs the transport coefficient T_k . We provide the mathematical details in the next section.

4. Mathematical Framework

We now present a hybrid wavelet-neural framework, where transport is computed in the wavelet basis as in the classical works, but transport coefficients are determined by a neural network. Regression directly in the wavelet basis has several advantages. It is well-established that the discrete transport operators are sparse in the wavelet domain, as most of the frequencies are concentrated in relatively few entries. This makes the problem of memorizing the light transport for a particular scene tractable. Second, we can compute the rendering equation directly, avoiding the need for low-frequency approximations or using neural networks as renderers. This allows for both view and lighting variations, enabling full generalization for complex light transport effects.

Representing Light Transport: Specifically, we represent the transport coefficients as,

$$T_k(\mathbf{x}, \boldsymbol{\omega}_o) = f(\mathbf{h}_k(\mathbf{x}), \boldsymbol{\omega}_r(\mathbf{x}), \mathbf{n}(\mathbf{x}), \mathbf{p}(\mathbf{x}); \Theta), \quad (4)$$

where \mathbf{h}_k is a feature vector as a function of spatial coordinate \mathbf{x} and wavelet index k . The feature field \mathbf{h} in essence captures how a wavelet light k scatters with global illumination when encountering scene point \mathbf{x} . f is a small multilayer perceptron (MLP) network that decodes the feature vector \mathbf{h}_k into the appropriate light transport wavelet coefficient T_k . Additional inputs to the MLP are the reflected direction $\boldsymbol{\omega}_r$, the reflection of the outgoing direction $\boldsymbol{\omega}_o$ about the surface normal $\mathbf{n}(\mathbf{x})$, all in global coordinates. It is well known that using $\boldsymbol{\omega}_r$ instead of $\boldsymbol{\omega}_o$ enables more coherent functions that are easier to optimize/regress for [RH02, VHM*22]. We also pass in the BRDF parameters which we denote as a vector \mathbf{p} , which could be spatially-varying. We adopt a standard GGX/Trowbridge-Reitz reflection model [WMLT07, TR75], with parameters \mathbf{p} including the diffuse and specular colors \mathbf{k}_d and \mathbf{k}_s and roughness σ . Θ denotes the parameters of the MLP.

Feature Field Representation: We have so far considered the feature vector $\mathbf{h}_k(\mathbf{x})$ for a given wavelet index k and spatial point \mathbf{x} . For compact representation, it is convenient to explicitly write the feature field \mathbf{h} as a tensor H with explicit parameters/indices, (we use notation $[]$ for accessing feature grids and $()$ for functions)

$$\mathbf{h} \equiv H[\mathbf{x}, \mathbf{k}, l], \quad (5)$$

where spatial location is designated by \mathbf{x} as before, a 3D vector. It is convenient for later representation to view the wavelet index as a 2D vector \mathbf{k} , corresponding to position on a cubemap after non-standard Haar wavelet decomposition. Finally, l is the 1D index of the feature vector (typically we use a vector of length 64). Note that explicitly representing H can be prohibitive, given it is effectively a 6D tensor. Therefore, we develop a compact tensor factorization, inspired by previous tensor approximations in the PRT and NeRF

Algorithm 1 PRT Rendering

```

1: procedure RENDER( $L, \{S_m\}, \{W_m\}, \{U_m\}, M, \Theta$ )
2:    $L \rightarrow L_k$             $\triangleright$  Wavelet Transform Lighting, Equation 3
3:    $K \rightarrow$  set of largest wavelet coefficients  $L_k$ 
4:   for all pixels do
5:     Determine  $\mathbf{x}, \boldsymbol{\omega}_r, \mathbf{n}, \mathbf{p}$             $\triangleright$  Inputs to Equation 4
6:     for  $k \in K$  do
7:        $\mathbf{h}_k = \sum_{m=1}^M S_m[\mathbf{x}] W_m[\mathbf{k}] U_m$             $\triangleright$  Equation 7
8:        $T_k = f(\mathbf{h}_k, \boldsymbol{\omega}_r, \mathbf{n}, \mathbf{p}; \Theta)$             $\triangleright$  MLP in Equation 4
9:     end for
10:     $B = \sum_{k \in K} L_k T_k$             $\triangleright$  Equation 3
11:     $B = B + B^d$             $\triangleright$  Add denoised direct lighting
12:  end for
13: end procedure

```

literature [TS06, CXG*22]. This approach also has similarities to PCA matrix decompositions, although we use a multiresolution hash grid [MESK22] for further compression rather than clustering as in previous PRT works [NNJ05, SHHS03, TS06]. Specifically, we use a CP tensor decomposition along the three modes (spatial \mathbf{x} , wavelet \mathbf{k} , feature l) with M terms to write,

$$H[\mathbf{x}, \mathbf{k}, l] \approx \sum_{m=1}^M S_m[\mathbf{x}] W_m[\mathbf{k}] U_m[l]. \quad (6)$$

In the equation above, $S_m[\mathbf{x}]$ is itself a 3D spatial feature grid depending on spatial coordinate \mathbf{x} , with trilinear interpolation to obtain the value at any \mathbf{x} . We represent S_m as a three-dimensional multiresolution hash encoding [MESK22] for ease of training and evaluation. This differs from most previous works that store information on scene vertices. In our experiments, we found that such a volumetric representation results in fewer view-dependent artifacts than a scene vertex representation (see Table 6) or a learned neural texture (single-resolution hash grid), and is easier to implement and compress, since parameterization of geometry remains a difficult problem. Note that the rendering costs of volumetric methods are independent of the level of detail of the scene; this has been exploited in previous works involving neural scene-to-volume computation [BSK23]. $W_m[\mathbf{k}]$ is a two-dimensional grid that stores a feature vector for each wavelet. Since the environment map is represented with a cubemap, wavelets and W_m can also be represented as a cubemap. Finally, $U_m[l]$ represents the “feature” dimension, which is a 1D vector for each m , where \mathbf{U} itself is simply a learnable matrix.

Given the tensor decomposition of the feature field, we can evaluate the feature vector in Equation 4 at runtime as follows,

$$\mathbf{h}_k(\mathbf{x}) = \sum_{m=1}^M S_m[\mathbf{x}] W_m[\mathbf{k}] \mathbf{U}_m, \quad (7)$$

where \mathbf{U}_m denotes the vector corresponding to all l in $U_m[l]$.

High-Level Rendering Algorithm: Algorithm 1 shows the pseudocode of our global illumination rendering algorithm. We first decompose the environment map into wavelets (Equation 3, line 2) and pick the set of largest wavelet coefficients K (line 3). The feature field $\mathbf{h} \equiv \{S_m, W_m, \mathbf{U}_m\}$ is stored and learned compactly using a tensor decomposition and multiresolution hash grid, as discussed

above. For a given pixel, we use rasterization or raytracing to find the primary hit at a pixel, with spatial location \mathbf{x} , normal \mathbf{n} , outgoing/reflected directions ω_o and ω_r , and BRDF parameters ρ (line 5).

Now, for each wavelet index $k \in K$, we determine the feature vector $\mathbf{h}_k(\mathbf{x})$ (see Equation 7, line 7). We now evaluate the MLP $f(\cdot)$ in Equation 4 (line 8) to obtain the transport coefficient T_k . Once the vector of all transport coefficients T_k with $k \in K$ is obtained, we determine the final color by performing the dot product with lighting in Equation 3 (line 10). We also add in the denoised direct lighting, computed separately (line 11).

5. Implementation and Algorithm

We now proceed to discuss the implementation and algorithm, based on the mathematical framework in the previous section.

Precomputation: Rendering. As with all PRT algorithms, there is a precomputation step. In our case, this involves not only offline rendering of the scene of interest, but also learning the relevant light transport parameters. We use a GPU-accelerated path tracer in OptiX with denoising to produce 512x512, 1024 samples per pixel ground truth images for training. Each image takes 1-3 seconds to render and it is not interactive, underscoring the need for our real-time PRT algorithm. The image resolution for real-time rendering can be changed arbitrarily at run-time, and we use higher-resolution 800 × 600 renders in some of our results.

For a given scene, we render approximately 4000 images under different environment maps and viewing conditions. We use 1000 indoor cubemaps and rotate each by 120 and 240 degrees to obtain the 3000 training lighting conditions. We only select indoor ones instead of outdoor ones since nonlinear wavelet selection on those tends to result in a larger quantity of meaningful wavelet coefficients [NRH03]. We generate 2000 camera locations using trajectories placed in the scene, and for each camera, we randomly select 2 environment maps from our training pool. We use one-sample-per-pixel raycasting to obtain the geometry parameters, reflection direction and BRDF parameters for these training views. This precomputation step takes about 1-3 hours. Note that the number of images is almost an order of magnitude less than the number needed in early wavelet methods, even for fixed view [NRH03].

We found that for highly specular areas, the algorithm requires multiple samples of view-dependent effects under different lighting conditions. For simple scenes (FOUR ANIMALS) where the camera can see almost every object at a given time, we place the cameras on predetermined spherical trajectories. For scenes that have many occluded areas (KITCHEN and ARMADILLO) we add an additional helical trajectory.

Precomputation: Learning Light Transport. The trainable parameters in our formulation are the feature grids $\{S_m\}$, $\{W_m\}$ and $\{\mathbf{U}_m\}$ as well as the parameters for the MLP f , which we denote as Θ . In particular, $\{S_m\}$ is represented as a multiresolution hash grid, which concatenates features obtained by trilinear interpolation across resolutions. Though past PRT methods have generally stored the feature vectors representing exitant radiance densely along the vertices of a mesh, we found that using such a volumetric representation significantly improves performance (see Table 4). $\{W_m\}$ is represented as a neural texture at the same resolution as the cubemap, $6 \cdot 64 \cdot 64$. We set the number of terms M for both feature grids

to be 64, which we found gives the best tradeoff between accuracy and speed, and we also set the feature dimension of the hash-grid to be 64 (so \mathbf{U} becomes a square matrix) as we found reducing this value does not meaningfully reduce the computation time. For ease of implementation, the learnable matrix \mathbf{U} is represented as a single-layer fully-fused MLP with no bias. f is implemented as a two-layer fully-fused MLP with width 128. The total size of our precomputed data is about 113 MB, the bulk of which stores the 3D multiresolution hashgrid representing $\{S_m\}$. This is substantially less than previous methods [NRH03, NRH04] even though we are considering full 6D indirect light transport. Our goal is to optimize

$$\{S_m\}, \{W_m\}, \{\mathbf{U}_m\}, \Theta = \arg \min \mathcal{L}(I(S_m, W_m, \mathbf{U}_m, \Theta), I_0), \quad (8)$$

where I is the image rendered using the procedure in Algorithm 1 and I_0 is the ground truth rendering discussed above.

At each training step, we randomly select an environment map from the training data, perform the non-standard wavelet decomposition over cubemap faces as in [NRH03] and select 300 wavelets. The choice of 300 is motivated by past findings ([NRH03], [NRH04]) noting that over 99% L^2 accuracy can be obtained by choosing less than 1% of the total wavelets. We importance sample half of these wavelets via unweighted selection from the environment map, and as the largest entries of the ground-truth wavelet transport matrix are uncorrelated with such a purely top- k selection, we uniformly sample wavelet indices for the other half to form K and L_k . We found that performing the wavelet transform without premultiplying the environment map entries by their solid angle factors (in effect, allowing the network to learn these) tends to produce better results.

We then sample 2048 pixels from the subset of our training data corresponding to this environment map and pass them through our algorithm to obtain the wavelet coefficients T_k corresponding to the indices k , which we multiply with L_k to obtain our final rendering. The network tends to converge much more slowly on the highly view-dependent areas of the scene, so we adopt a specialized importance sampling strategy on these pixels (see Fig. 2). In addition to the geometry buffer, we compute the empirical variances of all the hit points of the scene (stored in half-precision) and the high-frequency regions (obtained by subtracting a low-pass filtered version of the ground-truth indirect illumination from the original image). To deal with moderate-frequency regions we also importance sample based on the product of the specular coefficient times the roughness complement $k_s \cdot (1 - \sigma)$, and deal with all other regions via standard uniform sampling. We treat the output of these strategies as a probability distribution and sample 512 pixels from each accordingly.

We opt for such an image-based strategy as it is faster than supervision using the full ground-truth light-transport T . The latter, which would entail generating a 6D tensor at resolutions of $6 \times 64 \times 64$ for lighting and view (as used for cubemaps in [NRH03, NRH04]), would require over $(6 \times 64 \times 64)^2 \approx 6 \times 10^8$ images. Additionally, experiments showed that even if this tensor were subsampled at multiple views, the resulting convergence of the network was inadequate to getting good results on novel-view specularities. In the future, a more adaptive active exploration approach may be helpful to increase the training time spent on hard-

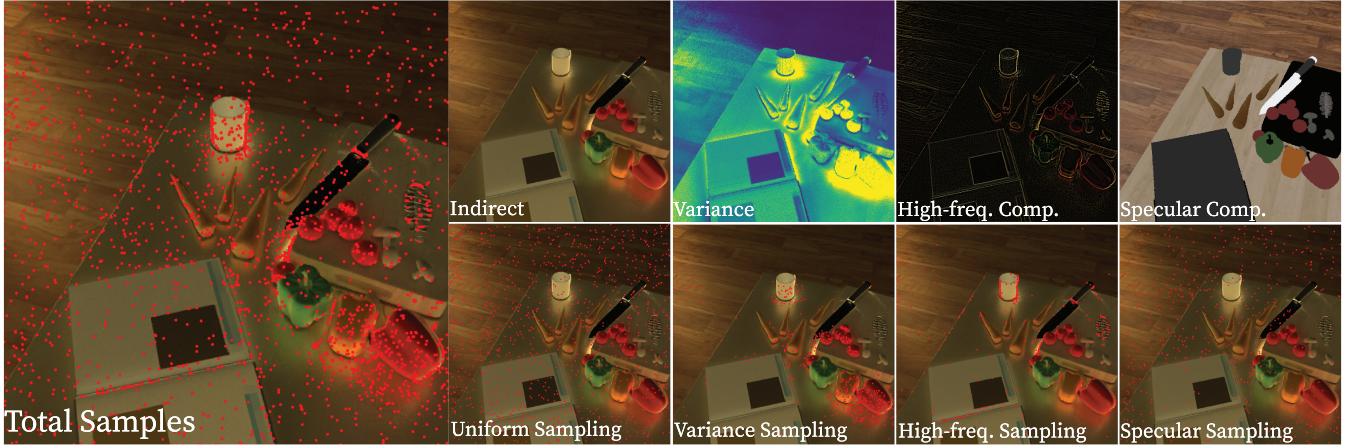


Figure 2: Sampling strategy used for precomputation/learning of indirect light transport. The left image shows the total sample point distribution. The points are made up of uniform samples over the image, and concentrations of samples in regions of high view-variance, high-frequency regions, and specular materials.

to-learn examples and prevent overfitting on the diffuse parts of the scene [DPD22].

We now compute the loss. Past works have demonstrated that error from applying an L_2 loss directly on output HDR images tends to be disproportionately affected by bright regions, so we apply a tonemap to our prediction and the ground-truth rendering before we take the loss. We use the μ -law tonemapping [KR17] $\text{TM}(x) = \text{sgn}(x) \frac{\log(\mu|x|+\epsilon)}{\log(\mu+\epsilon)}$, with $\mu = 10$ and $\epsilon = 1$, and define the loss as $\mathcal{L}(I, I_0) = \|\text{TM}(I) - \text{TM}(I_0)\|_2^2$. The extension to the negative numbers is required as our network operates directly in the wavelet domain, so our initial network predictions may result in negative colors after multiplication.

We discuss the resolutions and encoding of the feature grids and MLPs. For the three-dimensional multiresolution hash grid $\{S_m\}$, we use 32 levels, 2 features per level, base resolution 16, a hash table size of 2^{19} and a per-level scale of 1.3. This takes up the bulk of the total size of our method at 107 MB. While choosing a smaller hash table would result in a smaller model size, we found that it corresponded to a significant decrease in performance; see Table 3 for ablations on different hash table sizes. For the two-dimensional grid $\{W_m\}$, we store $6 \cdot 64 \cdot 64 \cdot 64$ values as full-precision floats, which takes up 6.1 MB. We represent the learnable matrix \mathbf{U} as a single-layer neural network with no bias or nonlinearity, taking up 36 KB. The final MLP takes as input the normal, reflection direction, and BRDF parameters and encodes only the reflection direction with spherical harmonics of maximum degree 4. The input roughness σ is additionally mapped to $\frac{\log(25\sigma+1)}{\log(25+1)}$ for better resolution in areas with low roughness. An evaluation of our encoding scheme can be found in Table 4. This final MLP has 128 neurons and 2 hidden layers with ReLU activations, resulting in a size of 124 KB. Further significant compression is possible using dithering and aggressive quantization of values (we use floats). The total training time is typically around 16 hours on an NVIDIA RTX 3090Ti.

Real-Time Rendering. Our renderer is implemented in C++ with Optix, CUDA, and Tiny-CUDA-NN [MRNK21]. As noted earlier,

direct lighting is computed separately by importance sampling the environment map and denoising (other approaches could also be used). We compute indirect lighting per Algorithm 1. Tiny-CUDA-NN is used for our neural rendering step to obtain the coefficients T_k by evaluating the MLP f , and we have a final CUDA kernel to compute the dot product of the transport coefficients and the environment map wavelet coefficients.

In practice, we have found a modest benefit from denoising the indirect lighting as well to avoid wavelet noise and we, therefore, apply denoising to the combined direct and indirect (this takes only about 1.5 ms and adds minimal overhead). However, our results remain of high quality even without denoising (see Fig. 8).

6. Results

We show results from our system, including screen-captures of the scenes we demonstrate (see video for real-time captures), comparisons to previous work, and an evaluation of some of our parameters. We compare to the best performing model described in Neural PRT [RBRD22], diffuse-specular separation.

6.1. Evaluation Methods

We created our evaluation dataset of lighting and views for numerical comparisons using held-out environments and views. We selected 70 unseen indoor and outdoor environment maps. Our evaluation views include 5 hand-picked locations that cover most objects in the scene and roughly 500 locations generated using evaluation trajectories, which consist of helices, circular sweeps, or figure eights. We include videos generated using some of these evaluation trajectories in the supplementary material, and visual results show representative lighting environments and views. Note that PSNR numbers in Figs. 1 and 3 refer to the scene with specific lighting/viewing configuration shown, and differ slightly from the averages over all configurations reported in Table 1. For all the performance metrics (PSNR, SSIM, and LPIPS) reported in the tables, we show full direct+indirect / indirect only.

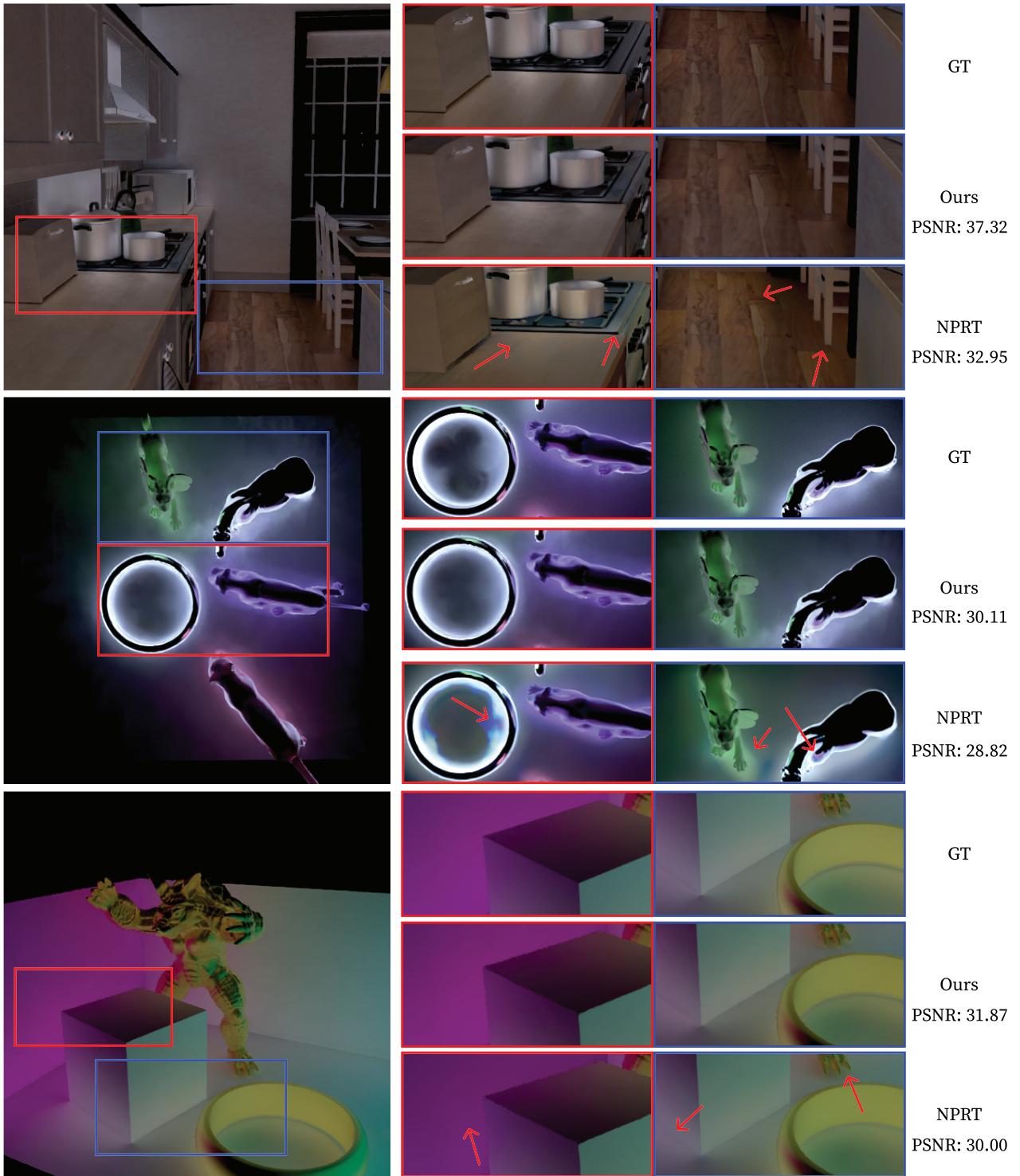


Figure 3: Visual comparisons of our method against Neural PRT [RBRD22]. Our method is able to reconstruct complex indirect illumination effects. For example, see the glossy reflections on the table and the floor in the KITCHEN scene (top row), the caustics in the FOUR ANIMALS scene (mid row), and the color bleeding in the ARMADILLO scene (bottom row).

6.2. Visual Results

In Figs. 1 and 3, we show example images from three scenes (all these results use denoising on both our method and the Neural PRT comparisons). These are all rendered at 24fps for 512×512 images and 13fps for 800×600 images on an NVIDIA RTX 4090. We see a range of glossy indirect reflection effects, including view-dependent caustics (see the FOUR ANIMALS scene). Capturing these high-frequency global illumination effects has been very challenging in previous PRT algorithms, since full high-resolution 6D lighting and view variation is required. Our video clearly shows smooth motions for changing view and relighting. In Fig. 1, we see the KITCHEN scene with glossy indirect reflections on the wall and the table. Our method produces accurate indirect illumination and overall global illumination rendering. In contrast, Neural PRT [RBRD22] works well for largely diffuse interreflections, but cannot perform well for high-frequency indirect highlights. Additionally, the top row of Fig. 3 shows a different view of the KITCHEN scene. Neural PRT produces an overall browner tone compared to the reference, with a color shift on the stove. It also misses glossy reflections on the table and the floor, making chair legs look flat and unnatural. Note that we retrained Neural PRT on each scene, using the same data as our method.

We do not include comparisons with other methods, as traditional non-learning PRT approaches [NRH03, NRH04] are limited to fixed view or diffuse for double-products and direct lighting only for triple-products [XZW*22]. Liu, et al. [LSS04] and Wang, et al. [WTL06]’s approaches use the in-out BRDF factorization which is limited to lower-frequency reflections, while Hasan et al.’s method [HPB06] cannot render caustic paths and is similarly restricted to fixed view.

The middle row of Fig. 3 shows the FOUR ANIMALS scene with challenging light transport with glossy reflection and a ring casting view-dependent caustics on a ground plane. These all-frequency view-dependent indirect reflections have historically been difficult for PRT methods, but our method produces fairly accurate results, where Neural PRT produces high frequency artifacts for the ring caustics and incorrect glossy reflections. Additionally, from the video of the FOUR ANIMALS scene, we show that our method is also more temporally stable when it comes to rotation of high-frequency effects, while NPRT tends to be smoother with more incorrect shading. Finally, the bottom row of Fig. 3 shows the ARMADILLO scene. Even in the largely diffuse regions, we still perform better than Neural PRT, since Neural PRT suffers from a color shift on the wall in the left inset and incorrect interreflections as indicated in the right inset. Note the missing edge of the cube and the lack of indirect reflections on the ground from the claws. The color shift of Neural PRT in the results above are likely due to the fact that Neural PRT does not use an orthonormal basis and has to approximate the linear dot product with a non-linear neural network. To further investigate the behavior on diffuse reflections, we also include an almost entirely DIFFUSE KITCHEN scene (all roughnesses set to 1), shown in Fig. 4. We see that our performance is substantially better, because we do not suffer from color shifts or other network artifacts in computing wavelet dot-products.

Finally, Fig. 5 makes a direct comparison of real-time low sample count (44 samples per pixel) path tracing with denoising in OptiX to our PRT rendering at equal time. Note that this a highly favorable case for OptiX since the scene is geometrically simple, enabling a moderate brute-force path tracing sample count. This sample count would be substantially lower in more complex production and game scenes. Nevertheless, indirect reflections from the path tracer miss detail near the toes, head and wings of the animals, which are captured by our PRT rendering. We have also observed greater temporal flickering in real-time path-traced renderings.

tiX to our PRT rendering at equal time. Note that this a highly favorable case for OptiX since the scene is geometrically simple, enabling a moderate brute-force path tracing sample count. This sample count would be substantially lower in more complex production and game scenes. Nevertheless, indirect reflections from the path tracer miss detail near the toes, head and wings of the animals, which are captured by our PRT rendering. We have also observed greater temporal flickering in real-time path-traced renderings.

6.3. Quantitative Results

Table 1 shows quantitative comparisons of Neural PRT and Our Method, both with and without denoising. We show results for both the full image and indirect lighting only on the metrics of PSNR, SSIM and LPIPS. The main results on the left of the table are on novel lights and trajectories *far* from the training data, corresponding to the result figures, and showing the generalization ability. The right of Table 1 shows additional statistics on held-out (*near*) views in training trajectories for both NPRT and our method.

For all scenes and metrics, our method has significantly better accuracy than Neural PRT. This is true both in scenes with strong glossy indirect reflections and complex caustics like KITCHEN and FOUR ANIMALS, and even when much of the global illumination is largely diffuse (ARMADILLO and DIFFUSE KITCHEN). Both NPRT and our method have a small metric drop from training trajectories (near views) to novel ones (far views), and we perform better on all metrics in both cases. This indicates our novel training strategy can generalize to unseen views and lighting conditions.

Our indirect PRT method does not have standard Monte Carlo noise, and denoising only provides a small (but important) boost. Even without denoising, we are better than Neural PRT with denoising on the PSNR metric, and comparable on SSIM in most scenes. Neural PRT does show a smaller bump in metrics after denoising than does our method; this is due to the fact that we predict coefficients directly in an orthonormal basis, which can result in noisier predictions in higher frequency regions. However, our method without denoising is also better than Neural PRT without denoising on almost all metrics.

Table 2 shows the runtime performance of our method on an RTX 4090. Note that the performance is essentially the same for all scenes, independent of the scene’s geometric complexity, because we use a volumetric hash-grid. The rendering time for one frame scales approximately linearly with resolution (512×512 is about twice as fast as 800×600) and the number of wavelets used (we used 64 wavelets for results in this paper, comparable to the number used in previous work [NRH03, NRH04], but 32 wavelets may be suitable in lower-frequency scenes, with 128 wavelets needed in very challenging scenes for higher fidelity). Rendering time remains interactive in all cases with real-time performance of 24fps or higher achieved for lower resolutions or number of wavelets.

6.4. Evaluation

We now evaluate various components of our algorithm. Figure 2 shows our adaptive sampling approach to training during the pre-computation phase. We first uniformly select samples on the scene

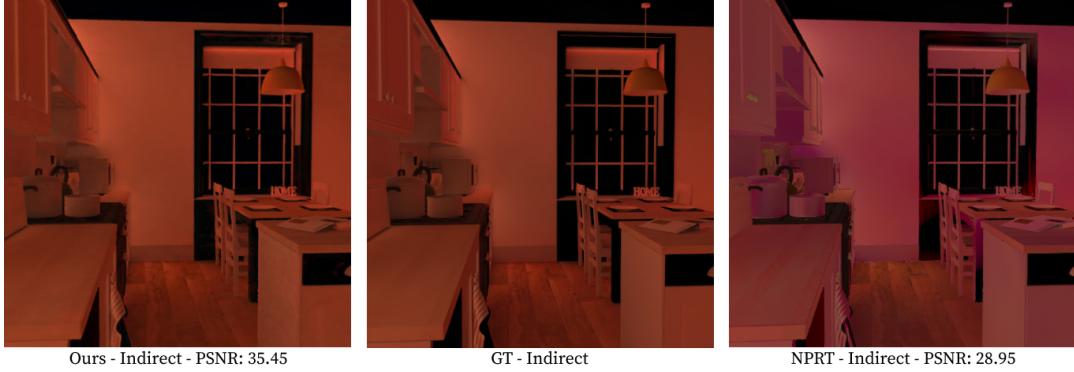


Figure 4: Comparison of our method with NPRT on DIFFUSE KITCHEN evaluated on a red environment map dissimilar to lighting conditions seen during training. In addition to an overall tone shift, the shading on certain objects (such as the pots on the stove) is inaccurate for NPRT, indicating our method is better able to generalize to unseen lighting conditions even for purely diffuse objects.

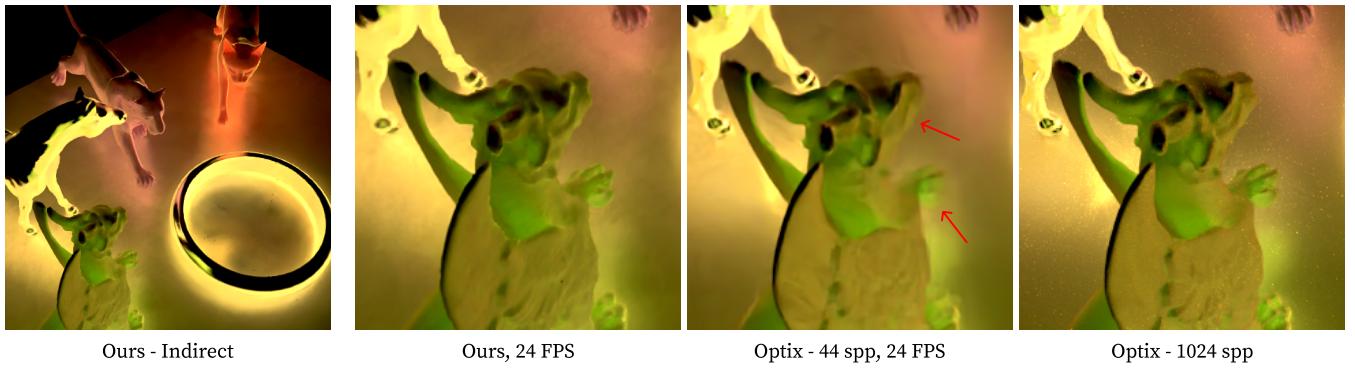


Figure 5: Comparison of our PRT method against Optix path-traced renderings with the same rendering time (44 samples per pixel) under indirect lighting. Note the lack of high-frequency details around the head and the toes of the animal figurine in the OptiX render.

Table 1: Quantitative comparison of our method and Neural PRT (NPRT), both with and without denoising, for full direct+indirect / indirect only. These metrics are evaluated on views and lighting conditions both near and far from the training set. Our method is better on all metrics on all scenes.

Metric	Methods	Denoising	Far Views				Near Views			
			Four Animals	Kitchen	Armadillo	Diffuse Kitchen	Four Animals	Kitchen	Armadillo	Diffuse Kitchen
PSNR(↑)	NPRT	✗	28.08 / 24.35	33.80 / 31.43	34.53 / 30.92	43.20 / 39.65	28.23 / 24.44	35.05 / 32.08	34.30 / 30.89	43.63 / 40.10
	NPRT	✓	28.53 / 25.09	34.56 / 32.45	34.74 / 31.53	43.45 / 40.55	28.73 / 25.19	35.58 / 32.88	34.52 / 31.53	43.83 / 41.16
	Ours	✗	28.53 / 25.16	35.83 / 33.42	36.18 / 32.45	45.82 / 42.24	28.77 / 25.38	36.71 / 34.93	36.59 / 32.68	46.18 / 42.68
	Ours	✓	29.62 / 26.21	36.59 / 34.31	36.48 / 33.17	46.27 / 43.69	29.96 / 26.45	37.64 / 35.59	36.94 / 33.44	46.44 / 43.97
SSIM(↑)	NPRT	✗	0.9233 / 0.8408	0.9577 / 0.8957	0.9743 / 0.9445	0.9910 / 0.9540	0.9295 / 0.8550	0.9478 / 0.9128	0.9758 / 0.9483	0.9923 / 0.9607
	NPRT	✓	0.9328 / 0.8527	0.9668 / 0.9165	0.9783 / 0.9511	0.9913 / 0.9620	0.9394 / 0.8665	0.9683 / 0.9271	0.9799 / 0.9551	0.9927 / 0.9685
	Ours	✗	0.9064 / 0.8218	0.9647 / 0.9170	0.9750 / 0.9460	0.9945 / 0.9783	0.9143 / 0.8374	0.9681 / 0.9211	0.9773 / 0.9512	0.9943 / 0.9765
	Ours	✓	0.9390 / 0.8642	0.9767 / 0.9414	0.9822 / 0.9596	0.9947 / 0.9862	0.9441 / 0.8751	0.9797 / 0.9434	0.9840 / 0.9637	0.9950 / 0.9834
LPIPS(↓)	NPRT	✗	0.0838 / 0.2049	0.0541 / 0.1792	0.0296 / 0.0851	0.0104 / 0.1009	0.0803 / 0.1948	0.0501 / 0.1665	0.0287 / 0.0844	0.0105 / 0.0921
	NPRT	✓	0.0547 / 0.1788	0.0280 / 0.0970	0.0185 / 0.0559	0.0073 / 0.0593	0.0496 / 0.1620	0.0275 / 0.1021	0.0187 / 0.0503	0.0071 / 0.0480
	Ours	✗	0.1315 / 0.1882	0.0498 / 0.1399	0.0293 / 0.0617	0.0059 / 0.0626	0.1202 / 0.1774	0.0549 / 0.1408	0.0257 / 0.0609	0.0063 / 0.0621
	Ours	✓	0.0489 / 0.1609	0.0161 / 0.0611	0.0121 / 0.0420	0.0029 / 0.0226	0.0453 / 0.1456	0.0211 / 0.0787	0.0112 / 0.0364	0.0038 / 0.0277

based on the indirect light. We then allocate additional samples in regions of high variance with respect to view, high-frequency regions, and those with high specular coefficients. The total sample distribution is shown in the leftmost image.

Table 3 shows an evaluation of different hashmap resolutions on the FOUR ANIMALS scene. We find empirically that a hash table

size of 2^{19} produces the best results, and also has a reasonable storage size.

In Table 4, we evaluated different encodings for the MLP. They all performed similarly, but best PSNR was achieved with a spherical harmonic encoding only for ω_r , with no impact on frame rate. Table 5 analyzes the encoding on just ω_r further on the more specular FOUR ANIMALS by considering learned feature vectors to en-

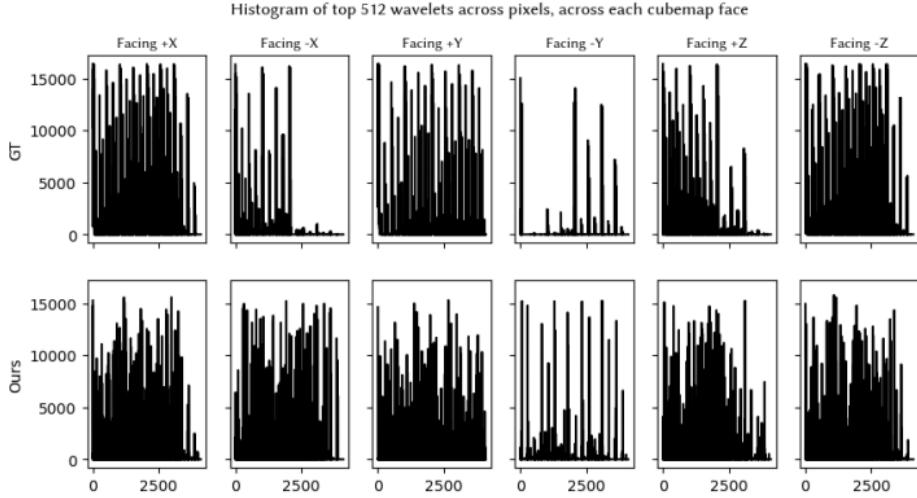


Figure 6: We fix a particular view of KITCHEN from our far-views dataset and render out the ground truth wavelet transport matrix as in [NRH03]. We compare it to the equivalent transport matrix output via our method by visualizing the histogram of the top 512 wavelets over all the pixels in the final image.

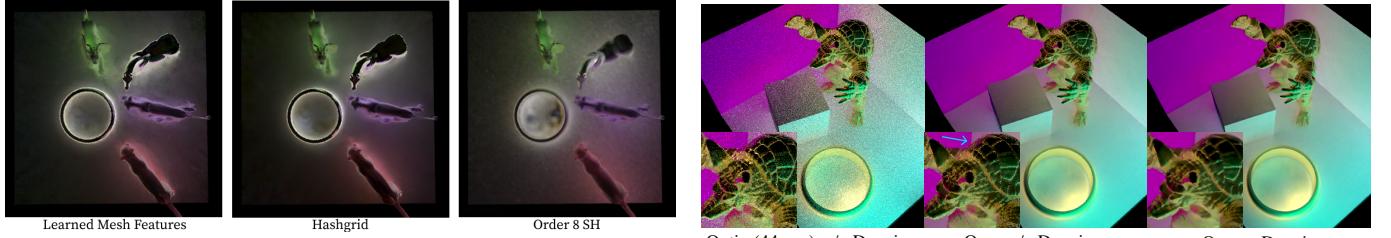


Figure 7: Ablation study on storing the learned feature vectors on mesh vertices (learned and SH) vs a hashgrid. Spherical harmonics with a maximum degree of 8 stored on mesh vertices [SKS02] are unable to properly represent caustics and the sharp reflections on the horse as expected, and also show some ringing and other artifacts. The learned mesh vertex features, while much better than SH, still cannot properly represent caustics, show a lot more noise in the reflections around the griffin and horse, and require a well-subdivided mesh. The use of the hashgrid solves these issues while being invariant to scaling of the scene.

code ω_r on the FOUR ANIMALS scene. For these learned feature vector experiments, we converted ω_r to UV coordinates within a differentiable cubemap and learned a hash grid for each face, which we called DCE. To compare this with the SH embedding, we considered a high- and low-frequency hash grid configuration to explore the impact of potentially overfitting to training views. To further explore the contribution of encoding ω_r , we also conducted experiments where the encoding is multiplied with the position and wavelet encodings in the initial CP decomposition phase to ablate its overall impact on the algorithm. In general, these approaches do not perform better than the simple spherical harmonic encoding, which we use for all of our results. A more thorough study on band-limiting the angular component of these neural algorithms may be interesting as future work.

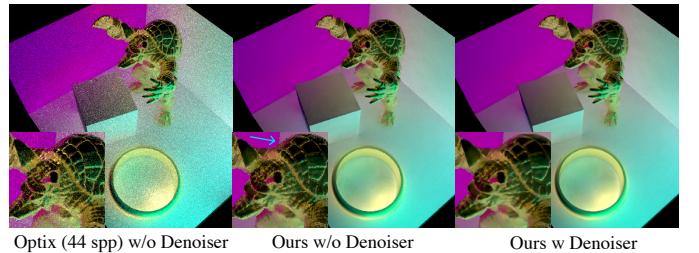


Figure 8: The effects of the denoiser on our proposed method. Standard OptiX path-tracing in equal time (left) is very noisy, and denoising does not address all issues (see Fig. 5). In contrast, our PRT rendering (middle) is already high quality without Monte Carlo noise, but does have a small amount of wavelet noise, and applying a denoiser helps improve the results (right) while adding almost no overhead.

Figure 6 visualizes the wavelet statistics of a particular view. While the shape of the distribution we learn is different from the ground truth, most of the energy of these wavelets is nonetheless contained within relatively few entries. Note that this is for the full transport matrix; as in previous work [NRH03], for rendering an image we can make use of the most important wavelet coefficients in the lighting, dramatically reducing the number of wavelets needed to 64 in our case.

In Table 6, we compare the use of hashgrids for S_m versus storing this information on vertices, using both learned feature vectors and spherical harmonics (maximum degree 8 using the traditional approach [SKS02]), showing the benefit of using the volumetric hashgrid. We do this on the FOUR ANIMALS scene, as this best showcases the challenges of storing the feature vectors on mesh

Table 2: Runtime Performance of our method with different resolutions, and # of wavelets. We achieve interactivity in all cases.

Resolution	#Wavelets	Framerate (FPS)	Frame time (ms)
512×512	32	42	24
	64	24	42
	128	12	87
800×600	32	25	40
	64	13	78
	128	6	180

Table 3: Quantitative results of our model evaluated on FOUR ANIMALS with different hashmap resolutions (direct+indirect/indirect only). The results shown are after denoising. We find that a hashmap size of 2^{19} produces the best results in all metrics.

HM size	Total size	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
2^{17}	34 MB	27.91 / 24.33	0.9219 / 0.8240	0.0687 / 0.1868
2^{18}	62 MB	29.61 / 26.10	0.9370 / 0.8592	0.0508 / 0.1631
2^{19}	113 MB	29.62 / 26.21	0.9390 / 0.8642	0.0489 / 0.1609
2^{20}	213 MB	28.73 / 25.37	0.9328 / 0.8404	0.0542 / 0.1677

vertices. We perform barycentric interpolation on the mesh vertices closest to the hit point. In the learned method, we apply a nonlinearity (softplus) to increase the representative capacity of the model. Spherical Harmonics perform worse than our method, lacking sharp reflection details and showing ringing and other artifacts on high-frequency light transport effects like caustics. This underscores that spherical harmonics are primarily a low-frequency representation. The learned feature vectors stored on the vertices perform better, but are noisier and still demonstrate inability to reconstruct caustics. Additionally, they require a well-subdivided mesh, so they would not be invariant to scaling scene complexity. Figure 7 shows a visual comparison.

Figure 8 shows results before (middle) and after (right) denoising, indicating that our initial results are already high quality, but a small amount of wavelet noise can be removed by the standard OptiX denoiser. The left image is a comparison to an equal time path-traced image without denoising which is substantially worse. Note that denoising brute-force path tracing does not resolve complex interreflections, as shown in Fig 5.

6.5. Limitations

Most limitations are inherited from previous PRT algorithms. The results, while significantly higher quality than previous work are not perfect, since we use only 64 wavelets, and also approximate the transport coefficients. Very high-frequency effects like mirrors are not perfectly reproduced (nor handled in previous techniques), and this can be seen in Figure 9 where we evaluate our method on the PBRT Bathroom scene with the mirror set to 0.05 roughness. For reference, the full table of metrics is listed in Table 7, evaluated on far views only. Some flicker can occasionally be seen in relighting as the selected wavelet coefficients change between

Table 4: Ablation study of different encodings on the KITCHEN scene. OB refers to the one-blob encoding [MMR*19]; SH refers to maximum degree-4 spherical harmonics. While encoding everything with one-blob performs slightly better than spherical harmonics in some categories, we chose to use spherical harmonics as it was superior to encoding everything in PSNR (for best results after denoising) while being extremely close in the other metrics.

Encoding	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
No encoding	21.84 / 17.84	0.8667 / 0.1294	0.0871 / 0.7739
OB(n), OB(ω_r), OB(σ)	35.75 / 33.35	0.9647 / 0.9179	0.0495 / 0.1365
Just OB(ω_r)	35.74 / 33.38	0.9641 / 0.9166	0.0498 / 0.1402
Just SH(ω_r)	35.83 / 33.42	0.9647 / 0.9170	0.0498 / 0.1399

Table 5: Ablation study of different encodings on solely ω_r on the FOUR ANIMALS scene. DCE refers to a differentiable hashgrid-based cubemap encoding the reflected direction; DCFE refers to a differentiable hashgrid-based cubemap encoding that is multiplied with the vertex and wavelet features (as a total factorization of the transport tensor).

Encoding	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
DCE(ω_r) (high freq.)	28.79 / 25.65	0.9315 / 0.8536	0.0615 / 0.1766
DCF(ω_r) (high freq.)	17.71 / 13.31	0.8111 / 0.4442	0.1871 / 0.4454
DCE(ω_r) (low freq.)	28.94 / 25.72	0.9324 / 0.8551	0.0591 / 0.1726
DCF(ω_r) (low freq.)	16.72 / 12.65	0.8089 / 0.4070	0.1811 / 0.4427
SH(ω_r) (Ours)	29.62 / 26.21	0.9390 / 0.8642	0.0489 / 0.1609

environments (we minimize this by using area-weighted selection, which minimizes visual error by quickly resolving the diffuse colors as in [NRH03]). Our optimization/training time for each scene can involve several hours, which is significantly higher than earlier non-learning approaches.

Finally, our volumetric hashgrid, while significantly improving quality, does use more space than would a pure MLP or CNN approach, or in some cases a vertex-based method. Neural PRT does have a smaller model size/faster evaluation due to its weights being constrained within a single neural network (it uses only MLPs/CNNs rather than a feature grid). Our contribution is to provide substantially higher quality compared to Neural PRT, while using data sizes significantly lower than previous wavelet-based PRT methods – our hashgrid is substantially more efficient than explicit transport matrix storage in early PRT work, often requiring at least an order of magnitude less storage. An analogy can be made with NeRF-like models where the tradeoff is that feature fields can provide higher accuracy at the cost of higher required storage space (still much less than explicitly tabulated representations). An interesting future direction is to quantify the tradeoff between explicit feature fields and implicit methods in the PRT space.

7. Conclusions and Future Work

All-frequency relighting for indirect glossy reflections with changing illumination and view has been one of the long-standing challenges for precomputed radiance transfer, and real-time rendering in general. In this paper, we have taken an important step toward this goal, showing that a new approach leveraging modern MLP,

Table 6: Comparison of storing S_m on mesh vertices versus a hashgrid. We compare to a learned mesh vertex-based scheme and to spherical harmonics with maximum degree of 8 [SKS02].

Feature scheme	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
Max. Deg. 8 SH	23.14 / 20.61	0.9147 / 0.8066	0.0655 / 0.2072
Mesh Vertices	28.38 / 25.08	0.9282 / 0.8383	0.0675 / 0.2090
Hashgrid	29.62 / 26.21	0.9390 / 0.8642	0.0489 / 0.1609

Table 7: Quantitative comparison on PBRT Bathroom. Our numbers after denoising are higher in all categories than NPRT, but our method finds difficulty in reconstructing perfect mirror effects. Results shown are full / indirect and evaluated on far views only.

Methods	Denoising	PSNR(\uparrow)	SSIM(\uparrow)	LPIPS(\downarrow)
NPRT	\times	30.25 / 28.94	0.9335 / 0.8817	0.0904 / 0.1841
NPRT	\checkmark	30.44 / 29.32	0.9375 / 0.8968	0.0720 / 0.1254
Ours	\times	30.75 / 29.72	0.9289 / 0.8839	0.1157 / 0.2056
Ours	\checkmark	31.14 / 30.24	0.9459 / 0.9116	0.0686 / 0.1129



Figure 9: An example limitation of our method on PBRT Bathroom (numbers provided for image only; averages can be found in Table 7). The mirror is Cook-Torrance with roughness 0.05. At a roughness this low both methods perform poorly, but while the overall highlights of NPRT are more inaccurate than ours, their capture of the reflection is more accurate (our method finds it hard to reconstruct the painting next to the mirror).

hashgrid, and novel factorization techniques can address the challenge of glossy global illumination, obtaining the best of both traditional orthogonal Haar wavelet decomposition and neural light transport approximation. In future work, we wish to consider alternative factorizations and feature grids that may be more accurate and compact, and alternatives to the hash-grid that can be computed directly on the object/scene surface. More broadly, this paper has introduced a neural representation of 6D light transport that may be applicable in many other areas including acquisition of the appearance of real scenes, and for modeling of neural materials.

8. Acknowledgements

We thank Peter-Pike Sloan, Alexandr Kuznetsov, Lingqi Yan, Ari Silvennoinen, Michal Iwanicki, Yash Belhe, Mohammad Shafiei, Pratul Srinivasan, Zhengqin Li and Alexander Mai for comments and discussions. We additionally thank Alexander Mai, Falko Kuester, Mustafa Yaldiz and Xiaoshuai Zhang for generously allowing us to use their compute resources, and we thank Gilles Rainer for answering questions. This work was funded in part from

NSF grants 2212085, 2100237 and 2120019, the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing. We also acknowledge gifts from Google, Adobe, Qualcomm, Meta and a Sony Research Award.

References

- [BDBS22] BELCOUR L., DELIOT T., BARBIER W., SOLER C.: A data-driven paradigm for precomputed radiance transfer. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3 (jul 2022). [2](#)
- [BSK23] BAKO S., SEN P., KAPLANYAN A.: Deep appearance prefiltering. *ACM Transactions on Graphics* 42, 2 (jan 2023). [4](#)
- [BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (SIGGRAPH 20)* 39, 4 (2020). [1](#)
- [CKS*17] CHAITANYA C., KAPLANYAN A., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZHARAI D., AILA T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (SIGGRAPH 17)* 36, 4 (2017). [3](#)
- [CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: TensoRF: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)* (2022). [2, 4](#)
- [DPD22] DIOLATZIS S., PHILIP J., DRETTAKIS G.: Active exploration for neural global illumination of variable scenes. *ACM Transactions on Graphics* (2022). [6](#)
- [FWH*22] FAN J., WANG B., HASAN M., YANG J., YAN L.: Neural layered BRDFs. In *SIGGRAPH 22* (2022). [3](#)
- [HCZ21] HADADAN S., CHEN S., ZWICKER M.: Neural radiosity. *ACM Transactions on Graphics (SIGGRAPH 21)* 40, 6 (2021), 236:1–236:11. [1](#)
- [HPB06] HASAN M., PELLACINI F., BALA K.: Direct to indirect transfer for cinematic relighting. *ACM Transactions on Graphics (Proc. SIGGRAPH 06)* 25, 3 (2006), 1089–1097. [2, 8](#)
- [KR17] KALANTARI N. K., RAMAMOORTHI R.: Deep high dynamic range imaging of dynamic scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36, 4 (2017). [6](#)
- [LSS04] LIU X., SLOAN P., SHUM H., SNYDER J.: All-frequency pre-computed radiance transfer for glossy objects. *EuroGraphics Symposium on Rendering* 04 (2004), 337–344. [1, 3, 8](#)
- [LTL*22] LYU L., TEWARI A., LEIMKUHLER T., HABERMANN M., THEOBALT C.: Neural radiance transfer fields for relightable novel-view synthesis with global illumination. In *European Conference on Computer Vision (ECCV 22)* (2022), pp. 17:153–17:169. [3](#)
- [LWLDD11] LAURIJSSEN J., WANG R., LAGAE A., DUTRE P.: Pre-computed gathering of multi-bounce glossy reflections. *Computer Graphics Forum* 30, 8 (2011), 2270–2278. [2](#)
- [MESK22] MULLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (SIGGRAPH 22)* 41, 4 (2022). [2, 4](#)
- [MMR*19] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVAK J.: Neural importance sampling. *ACM Trans. Graph.* 38, 5 (Oct. 2019), 145:1–145:19. [11](#)
- [MRNK21] MULLER T., ROUSSELLE F., NOVAK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics (SIGGRAPH 21)* 40, 4 (2021), 36:1–36:16. [6](#)
- [MST*20] MILDENHALL B., SRINIVASAN P., TANCIK M., BARRON J., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)* (2020), pp. I-405–I-421. [1](#)

- [MSW04] MEI C., SHI J., WU F.: Rendering with Spherical Radiance Transport Maps. *Computer Graphics Forum (Eurographics 04)* 23, 3 (2004), 281–290. 2
- [MTR08] MAHAJAN D., TSENG Y., RAMAMOORTHI R.: An analysis of the in-out BRDF factorization for view-dependent relighting. *Computer Graphics Forum (EGSR 08)* 27, 4 (2008), 1137–1145. 3
- [NNJ05] NISHINO K., NAYAR S., JEBARA T.: Clustered blockwise PCA for representing visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 10 (2005), 1675–1679. 4
- [NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 376–381. 1, 2, 3, 5, 8, 10, 11
- [NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics (Proc. SIGGRAPH 04)* 23, 3 (2004), 475–485. 1, 2, 3, 5, 8
- [NVI18a] NVIDIA: Optix ray tracing engine, 2018. <https://developer.nvidia.com/optix>. 1
- [NVI18b] NVIDIA: Rtx platform, 2018. <https://developer.nvidia.com/rtx>. 1
- [PBD*10] PARKER S., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29, 4 (2010), 66:1–66:13. 1
- [Ram09] RAMAMOORTHI R.: Precomputation-Based Rendering. *Foundations and Trends in Computer Graphics and Vision* 3, 4 (2009), 281–369. 3
- [RBRD22] RAINER G., BOUSSEAU A., RITSCHEL T., DRETTAKIS G.: Neural precomputed radiance transfer. *Computer Graphics Forum (Proc. EG 2022)* 41, 2 (2022), 365–378. 2, 3, 6, 7, 8
- [RDL*15] REN P., DONG Y., LIN S., TONG X., GUO B.: Image based relighting using neural networks. *ACM Transactions on Graphics (SIGGRAPH 15)* 34, 4 (2015), 111:1–111:12. 3
- [RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency Space Environment Map Rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH 02)* 21, 3 (2002), 517–526. 4
- [RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Transactions on Graphics (SIGGRAPH 13)* 32, 4 (2013), 130:1–130:12. 3
- [SHHS03] SLOAN P., HALL J., HART J., SNYDER J.: Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 382–391. 2, 4
- [SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics (Proc. SIGGRAPH 02)* 21, 3 (2002), 527–536. 1, 10, 12
- [SKW*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *High Performance Graphics* (2017), p. 2. 3
- [SLS05] SLOAN P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 05)* 24, 3 (2005), 1216–1224. 2
- [TR75] TROWBRIDGE T., REITZ K. P.: Average irregularity representation of a rough surface for ray reflection. *J. Opt. Soc. Am.* 65, 5 (1975), 531–536. 4
- [TS06] TSAI Y., SHIH Z.: All-Frequency Precomputed Radiance Transfer using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 06)* 25, 3 (2006), 967–976. 2, 4
- [TSM*20] TANCIK M., SRINIVASAN P., MILDENHALL B., FRIDOVICHKEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS* (2020). 1
- [VHM*22] VERBIN D., HEDMAN P., MILDENHALL B., ZICKLER T., BARRON J., SRINIVASAN P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *Computer Vision and Pattern Recognition (CVPR)* (2022). 4
- [WMLT07] WALTER B., MARSCHNER S., LI H., TORRANCE K.: Microfacet models for refraction through rough surfaces. In *EuroGraphics Symposium on Rendering (EGSR)* (2007), pp. 195–206. 4
- [WTL06] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of glossy objects. *ACM Transactions on Graphics* 25, 2 (2006), 293–318. 1, 3, 8
- [XZW*22] XU Z., ZENG Z., WU L., WANG L., YAN L.: Lightweight neural basis functions for all-frequency shading. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 22)* (2022). 2, 3, 8
- [ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.: Precomputed shadow fields for dynamic scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 05)* 24, 3 (2005), 1196–1201. 2
- [ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum (EUROGRAPHICS STAR 2015)* 34, 2 (2015), 667–681. 1