Lightweight Neural Basis Functions for All-Frequency Shading

Zilin Xu
Shandong University
China
zilin.xu@mail.sdu.edu.cn

Zheng Zeng
University of California, Santa Barbara
USA
zhengzeng@cs.ucsb.edu

Lifan Wu
NVIDIA
USA
lifanw@nvidia.com

Lu Wang*
Shandong University
China
luwang_hcivr@sdu.edu.cn

Ling-Qi Yan
University of California, Santa Barbara
USA
lingqi@cs.ucsb.edu

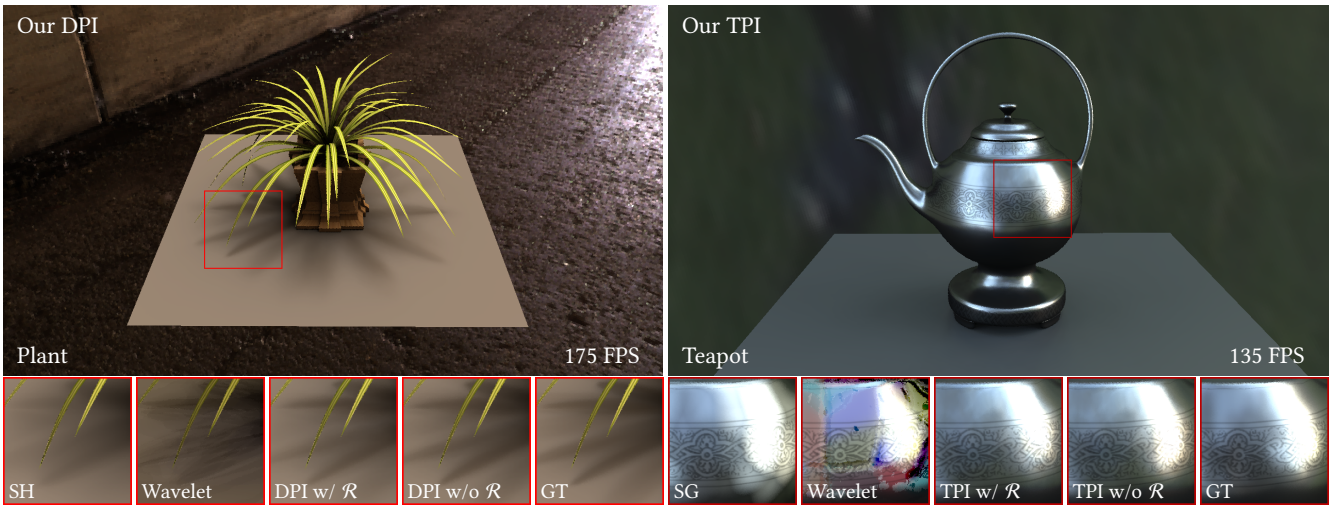

Figure 1: Our lightweight neural basis functions compactly represent complex 2D spherical functions (lighting, visibility and BRDF), and efficiently support practical computation, such as Double Product Integral (DPI, left) and Triple Product Integral (TPI, right), pervasively used for all-frequency shading. Our neural basis functions faithfully preserve all-frequency details from the ground truth (GT). However, spherical harmonics (SH) suffer from the absence of all-frequency shadows, while spherical Gaussians (SG) and wavelets lose details at equal compression rates compared with ours (0.39%). Also, our neural basis functions easily support efficient rotation ($\mathcal{R}$), and they are temporally stable without flickering as opposed to wavelets.

ABSTRACT

Basis functions provide both the abilities for compact representation and the properties for efficient computation. Therefore, they are pervasively used in rendering to perform all-frequency shading. However, common basis functions, including spherical harmonics (SH), wavelets, and spherical Gaussians (SG) all have their own limitations, such as low-frequency for SH, not rotationally invariant for wavelets, and no multiple product support for SG. In this paper, we present *neural basis functions*, an implicit and data-driven set of basis functions that circumvents the limitations with all desired properties. We first introduce a representation neural network that takes any general 2D spherical function (e.g. environment lighting, BRDF, and visibility) as input and projects it onto the latent space as coefficients of our neural basis functions. Then, we design several lightweight neural networks that perform different types of computation, giving our basis functions different computational properties such as double/triple product integrals and rotations. We demonstrate the practicality of our neural basis functions by integrating them into all-frequency shading applications, showing that our method not only achieves a compression rate of 0.39% and 10×-40× better performance than wavelets at equal quality, but

*Corresponding author: luwang_hcivr@sdu.edu.cn.

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

ACM ISBN 978-1-4503-9470-3/22/12…$15.00
https://doi.org/10.1145/3550469.3555386

also renders all-frequency lighting effects in real-time without the aforementioned limitations from classic basis functions.

CCS CONCEPTS

• **Computing methodologies → Rendering**.

KEYWORDS

basis function, all-frequency, neural rendering

ACM Reference Format:
Zilin Xu, Zheng Zeng, Lifan Wu, Lu Wang, and Ling-Qi Yan. 2022. Lightweight Neural Basis Functions for All-Frequency Shading. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3550469.3555386

1 INTRODUCTION

Basis functions are useful in rendering in computer graphics. They not only enable compact representations of 2D spherical functions, such as environment lighting, Bidirectional Reflectance Distribution Functions (BRDFs), and visibility functions, but also provide nice computational properties that simplify complex light transport and dynamic all-frequency shading. The desired properties include the expressiveness of capturing all-frequency lighting effects, rotational invariance, and the ability of supporting efficient double or triple product integrals.

However, different types of basis functions have certain limitations. Even for 2D basis functions, which have been studied for decades, trade-offs still have to be made according to the properties of different basis functions. For example, it is pervasively believed that spherical harmonics (SH) are only suitable to capture low-frequency signals, wavelets do not easily support spherical rotations, and spherical Gaussians (SG) do not have orthonormality.

The unique limitations of each type of basis functions inspire us to think about the question: can we design a set of basis functions that has all the desired properties and does not have these commonly encountered objectionable limitations? This question can be mathematically difficult, but in this paper, we make the first attempt to solve this question in a data-driven way. We also demonstrate the practicality of our data-driven neural basis functions in all-frequency shading applications utilizing basis functions.

Our high-level idea is using neural networks to handle the representation and computation tasks of 2D functions. We first present an autoencoder-style representation neural network that takes any general 2D spherical function as input and learns to compress it into a latent vector. The latent vector acts as the coefficients of our implicit basis functions. Then, we focus on common types of computations (such as rotations, double product integrals, and triple product integrals) applied in all-frequency shading, and train a series of lightweight networks that endow our basis functions with the desired computational properties. The lightweight networks act as individual operators to perform computation solely in the latent space, just like the way that we operate on the coefficients of traditional basis functions.

With our neural basis functions, we demonstrate that a 2D spherical function can be significantly compressed with much higher frequencies retained, even when compared with wavelets using equal storage. Moreover, double and triple product integrals, traditionally believed expensive for SH (triple product only) and SG, can be faithfully calculated using our computational networks. Furthermore, once represented using our neural basis functions, the original function will *never have to be expanded* again. All computations are immediately performed in the latent space, such as rotations using our rotation network and subsequent shading (which involves double/triple product integrals) using our integration networks. We believe our method has opened up new possibilities to incorporate lightweight neural networks in the core rendering process.

2 RELATED WORK

In this section, we briefly review techniques that are most relevant to us. We categorize them into three different types:

Basis functions. Various types of (2D) basis functions, such as spherical harmonics (SH), spherical Gaussians (SG), and wavelets, have been widely used in rendering [Ritschel et al. 2012]. Basis functions are usually defined in the spherical domain, providing compact representation for lighting, BRDFs, and visibility functions. A small number of SH basis functions are capable of approximating low-frequency environment maps and materials [Cabral et al. 1987; Westin et al. 1992; Ramamoorthi and Hanrahan 2001a,b; Marques et al. 2022]. While it requires a lot of SH functions to approximate all-frequency lighting effects, a sparse set of wavelets can faithfully reproduce them [Ng et al. 2003, 2004; Sun and Mukherjee 2006; Sun and Ramamoorthi 2009]. To model surface reflectance, many existing works used SG mixtures (or its anisotropic variants) to approximate the normal distribution function (NDF) of a rough surface [Han et al. 2007; Wang et al. 2009; Xu et al. 2013]. Since different kinds of basis functions have certain appealing mathematical properties, they are effective on some specific rendering applications. However, they all have individual issues as will be pointed out in Sec. 3, which are the main problem that we tackle in this paper. Our goal is to use a data-driven approach to figure out a set of neural basis functions that has all the desired properties, including compact all-frequency representation, rotational invariance, and efficient double/triple integral computation.

Neural approaches and lightweight designs in core rendering. Recently, there is abundant work using neural networks to synthesize photorealistic images [Tewari et al. 2020] and to perform inverse rendering [Wang et al. 2018; Sztrajman et al. 2020; Calian et al. 2018]. A series of works [Mildenhall et al. 2020; Bi et al. 2020; Zhang et al. 2021] learned neural radiance fields (NeRFs) for synthesizing novel views of the given scenes. However, most of them do not focus on core rendering. We mainly discuss methods that use neural networks to improve and aid the physically based rendering process. For example, deep shading [Nalbach et al. 2017] and neural direct-to-indirect light transport [Xin et al. 2020] exploit easy-to-acquire screen space auxiliary information to predict more complex shading effects, and neural complex luminaires [Zhu et al. 2021] design three lightweight neural networks to replace the key operations for lighting in the modern offline rendering framework: light radiance evaluation, sampling and pdf (probability density function) calculation. These approaches are usually applied as shaders in the rendering process, as opposed to those

using neural networks to completely bypass/replace the rendering process. Since performance is the key to core rendering, especially for real-time rendering where an inference time of 50 milliseconds is considered prohibitively expensive, we decide to use lightweight neural networks in contrast to the complex ones used in computer vision applications. Despite being lightweight, we try to inject more prior knowledge into the networks to keep them as lightweight as possible.

Neural basis functions and operations in the latent space. The concept of "neural basis" has been used in the neural rendering community [Wizadwongsa et al. 2021; Garbin et al. 2021; Granskog et al. 2020]. This line of research work often involves dimension reduction, for example, separating directions and positions into two independent parts. In this way, one part can be considered as coefficients, and the other is treated as basis functions. The separated representation indeed looks like the use of basis functions, but they are significantly different because they only represent/compress information but do not support computation/operation. There has been recent work on performing computations in the latent space, e.g., rotation-invariant or rotation-equivariant convolutional neural networks for spherical signals [Cohen et al. 2018; Esteves et al. 2018]. In particular, Rhodin et al. [2018] learned a geometry-aware latent representation by treating the latent vector as a set of 3D points and applying rotations directly on the latent vector. Rainer et al. [2022] train neural networks to encode environment lighting and scene attributes (position, normal, etc.) into the latent space, and predict the radiance from arbitrary combinations of these two types of latent vectors. Neural network itself can also be embedded into the latent space [Sztrajman et al. 2021] or be treated as latent variables [Hu et al. 2020]. Our goal is to ensure neural basis functions have certain mathematical properties, by introducing different computation neural networks as different operations completely performed in the latent space, leading to a operand-operator style neural solution, similar to the concurrent work by Fan et al. [2022].

3 PRELIMINARIES

Basis functions are a (possibly infinite) set of functions $\mathbf{B}_i(x)$, $i \in \{1 \dots N\}$ that can be used to approximately represent any function $f(x)$ as a linear combination:

$$f(x) \approx \sum_{i=1}^{N} a_i \mathbf{B}_i(x), \tag{1}$$

where x is a variable of a certain dimension and the a_i-s are the coefficients corresponding to the basis functions. We are especially interested in 2D spherical functions as they are widely used in rendering for representing lighting, BRDFs (strictly, its 2D slice instead of its full 4D form), and visibility functions. We do not intend to model all high-dimensional signals that would be required in rendering.

Projection. To obtain each coefficient a_i, we need to project the original function onto each basis function, which is the product integral of $f(x)$ and $\mathbf{B}_i$:

$$a_i = \int f(x)\, \mathbf{B}_i(x)\, \mathrm{d}x. \tag{2}$$

Orthonormality. A set of basis functions is orthonormal over the domain Ω if

$$\int_{\Omega} \mathbf{B}_i(x)\, \mathbf{B}_j(x)\, \mathrm{d}x = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{3}$$

Given orthonormal basis functions, it becomes very convenient to compute the product integral of two functions, i.e. *double product integral* (DPI). With both functions projected onto this set of basis functions, the DPI result is a simple dot product of the coefficients.

Rotational invariance. We first define a rotation operator $R(f, \alpha)$ that rotates a spherical function f with some angles α with respect to a fixed central axes of the origin spherical function. In general, a set of basis functions defined in spherical domain is rotational invariant if

$$R(f(x), \alpha) \approx \sum_{i=1}^{N} a_i R(\mathbf{B}_i(x), \alpha). \tag{4}$$

Rotational invariance is an important property for practicality because rotating basis functions itself is usually much cheaper and more efficient compared with rotating and re-projecting the original spherical function. Note that the coefficients a_i are invariant after rotation.

Multiple product integrals. When multiple ($M > 2$) functions are represented using the same set of basis functions, the integral of the product of M basis functions is

$$c_{ij\cdots k} = \int \underbrace{\mathbf{B}_i(x)\, \mathbf{B}_j(x) \cdots \mathbf{B}_k(x)}_{\text{M terms}}\, \mathrm{d}x. \tag{5}$$

The multiple product integrals are especially useful in rendering. In this paper, we focus on the two most widely used cases: the *double product integral* (DPI, when $M = 2$) for all-frequency shading, and the *triple product integral* (TPI, when $M = 3$) for dynamic relighting. And next we will give a brief introduction of them.

All-frequency shading via DPI. The key idea of this application is to separate and move the fixed portion of light transport to the precomputation stage. For example, Sloan et al. [2002] handle environment lighting on diffuse objects by partitioning the rendering equation into two parts, the incident lighting and the transport function at each shading point with normal $\mathbf{n}$:

$$L(\boldsymbol{\omega}_o) = \int_{\Omega} L(\boldsymbol{\omega}_i) \underbrace{V(\boldsymbol{\omega}_i) f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \max(0, \mathbf{n} \cdot \omega_i)}_{\text{transport function}}\, \mathrm{d}\boldsymbol{\omega}_i, \tag{6}$$

where the environment lighting $L(\boldsymbol{\omega}_i)$ is a 2D spherical function, and the precomputed transport function that consists of a constant Bidirectional Reflectance Distribution Function (BRDF) f_r for diffuse materials and a 2D spherical visibility function $V(\boldsymbol{\omega}_i)$ specifying whether the light is occluded in different directions, is also a 2D spherical function of $\boldsymbol{\omega}_i$. Therefore, computing all-frequency shading boils down to evaluating double product integrals (DPI).

Dynamic relighting via TPI. One important variation of the abovementioned shading application is using triple product integrals (TPI) for relighting with a fixed view. The idea is to split the rendering

Table 1: Strengths and weaknesses of different basis functions. Note orthonormality is out of consideration, since its original purpose is to provide an efficient computation of product integrals, which is already satisfied by our design of DPI/TPI neural network.

	SH	Wavelet	SG	Ours
All-frequency	×	✓	✓	✓
Rotational invariance	✓	×	✓	✓
Orthonormality	✓	✓	×	N/A
DPI support	✓	✓	×	✓
TPI support	×	✓	×	✓

equation into three different parts: lighting, visibility, and the BRDF (with the cosine term) as

$$L(\boldsymbol{\omega}_o) = \int_\Omega L(\boldsymbol{\omega}_i) \underbrace{V(\boldsymbol{\omega}_i)}_{\text{visibility}} \underbrace{f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \max(0, \mathbf{n} \cdot \omega_i)}_{\text{cosine-weighted BRDF}} \mathrm{d}\boldsymbol{\omega}_i. \quad (7)$$

With a fixed view direction $\boldsymbol{\omega}_o$, the BRDF f_r will become a 2D spherical function of $\boldsymbol{\omega}_i$, so the BRDF term can be precomputed and represented using basis functions at every pixel. When the lighting condition changes, we only need to update the coefficients of the lighting function using a projection as defined in Eq. (2). Then, we can compute a triple product integral for each pixel to generate a new image efficiently.

Note that, the fixed view relighting can be extended to dynamic view by simply precomputing all BRDF slices from different views with classic basis functions [Ng et al. 2004] and generating tables of 2D BRDF slices. In this way, shading with dynamic views becomes no different from that with a fixed view, which are irrelevant to the choice of basis functions. Therefore, we only focus on the fixed view relighting for simplicity.

Though pervasively used, traditional basis functions defined in the 2D spherical domain all have their own limitations. We have briefly discussed several types of commonly used basis functions in Sec. 1, and we summarize their strengths and weaknesses in Table 1. A more detailed discussion is provided in the supplemental document.

4 NEURAL BASIS FUNCTIONS

4.1 Motivation and overview

Being aware of the various limitations of different basis functions, we would like to design a set of functions that keep all the good properties but do not have the issues. We first briefly summarize and analyze the key design principles:

- high compression rate, ideally introducing an order of magnitude less storage cost as wavelets at the same quality;
- capturing all-frequency lighting effects, i.e. keeping more high-frequency details such as shadows and glossy highlights with equal storage cost as compared to any other types of basis functions;
- fast rotation like SH, allowing operations only on the coefficients rather than on the original functions;

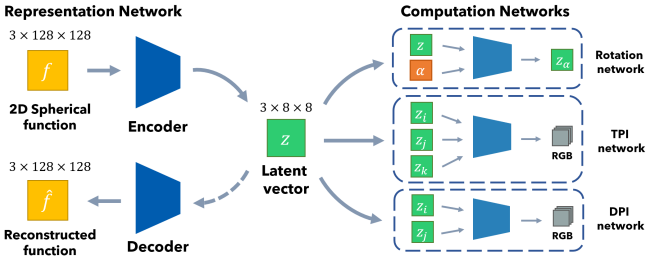

Figure 2: The overall scheme of our neural basis functions. Our encoder ($\mathcal{E}$) compresses a 2D spherical function to a short latent vector, which serves as coefficients of our implicit neural basis functions. Although our decoder ($\mathcal{D}$) is capable of converting the latent vector to the original function, it is not used in subsequent computations. All the mathematical properties are guaranteed by our lightweight computation networks: rotation ($\mathcal{R}$), DPI and TPI. They conduct computations solely in the latent space and ensure the real-time performance.

- efficient double and triple product integrals; and
- flexibility to support only desired operations and constraints rather than all mathematical requirements for basis functions. For example, we do not enforce orthonormality since its original purpose is to efficiently compute product integrals, which is already satisfied by our neural DPI/TPI operations.

With our neural basis functions, we are able to achieve all these goals. For a 2D spherical function, we compress it into a latent vector using a representation network (Sec. 4.2). The elements from the latent vector serve as coefficients, and the basis functions are completely implicit. We have reached a compression rate of 0.39%, equivalent to wavelets keeping 0.19% coefficients (because the sparse wavelet coefficients must be stored together with their indices), but as accurate as wavelets using 10×-40× more storage than ours. For the computations, we focus on rotation (Sec. 4.3), DPI (Sec. 4.4), and TPI (Sec. 4.5), all conducted solely in the latent space. Fig. 2 shows the overall scheme of our neural basis functions, and we present the detailed network designs in the supplemental document.

4.2 Representation

We use an autoencoder as the *Representation* ($\mathcal{P}$) *network*. Given a 2D spherical function f (discretized using the octahedral mapping introduced by Clarberg [2008]) as input, the encoder $\mathcal{E}(\cdot)$ can compress it into a low-dimensional latent vector $\mathbf{z}$:

$$\mathbf{z} = \mathcal{E}(f). \quad (8)$$

The elements from $\mathbf{z}$ serve as coefficients of our implicit basis functions. We resample the original function f with the resolution of $3 \times 128 \times 128$, and the encoded latent vector $\mathbf{z}$ has a dimension of $3 \times 8 \times 8$, indicating a compression rate of 0.39%. Then, the decoder $\mathcal{D}(\cdot)$ reconstructs a spherical function $\hat{f}$ from the latent vector $\mathbf{z}$:

$$\hat{f} = \mathcal{D}(\mathbf{z}). \quad (9)$$

Our goal is to approximate $\hat{f}$ to the original function f as closely as possible. Note that we just use our full representation network to encode 2D spherical functions into latent vectors in the training stage – only the encoded low-dimensional latent vectors are used during rendering, rather than the expensive decoder.

4.3 Rotation

Our *Rotation network* ($\mathcal{R}$) takes in a latent vector $\mathbf{z}$ and rotation parameters α, producing a new latent vector:

$$\mathbf{z}_\alpha = \mathcal{R}(\mathbf{z}, \alpha). \tag{10}$$

The reconstructed function from the rotated latent vector should closely resemble the rotated function:

$$\mathcal{D}(\mathbf{z}_\alpha) \approx R(f, \alpha), \tag{11}$$

where $R(\cdot,\cdot)$ is the rotation operator defined in Eq. (4).

For the representation of the rotation parameters α, we use the continuous rotation representation $SO(3)$ [Zhou et al. 2019] rather than Euler angles or quaternions, because the latter ones are not continuous representations for spherical rotations and thus are not suitable for neural networks to learn.

4.4 Double Product Integral

To support the double product integral (DPI) operation and achieve fast computation at runtime, we design a lightweight *Double Product Integral* (DPI) *Network*. Given two latent vectors $\mathbf{z}_i = \mathcal{E}(f_i)$ and $\mathbf{z}_j = \mathcal{E}(f_j)$ that are compressed by the encoder, representing two spherical functions, the DPI network takes these two latent vectors as input and performs double product integral in the latent space:

$$\mathrm{DPI}(\mathbf{z}_i, \mathbf{z}_j) \approx \int_\Omega f_i(\boldsymbol{\omega}) f_j(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}, \tag{12}$$

here $f_i(\boldsymbol{\omega}) \cdot f_j(\boldsymbol{\omega})$ is the double product of the corresponding spherical functions of latent vectors $\mathbf{z}_i$ and $\mathbf{z}_j$. For all-frequency shading applications, these two spherical functions are the environment lighting and the transport function, and the output of the DPI network is a color value indicating the outgoing radiance.

4.5 Triple Product Integral

To support efficient triple product integral (TPI) operation, we introduce a *Triple Product Integral* (TPI) *Network*.

Similar to the DPI network, given three latent vectors $\mathbf{z}_i = \mathcal{E}(f_i)$, $\mathbf{z}_j = \mathcal{E}(f_j)$ and $\mathbf{z}_k = \mathcal{E}(f_k)$ compressed by the encoder, the TPI network perform triple product integral in the latent space:

$$\mathrm{TPI}(\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k) \approx \int_\Omega f_i(\boldsymbol{\omega}) f_j(\boldsymbol{\omega}) f_k(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}, \tag{13}$$

here the three functions are environment lighting, visibility and BRDF, and the output of the TPI network is a color value of the outgoing radiance after relighting.

5 TRAINING AND INFERENCE

In this section, we give a brief introduction of our training strategy and loss functions. Then we describe necessary information regarding rendering integration and runtime inference. For more details, please refer to our supplementary document.

5.1 Data preparation

Our training data consists of general environment maps and geometric data (including visibilities, BRDFs and transport functions). All are 2D spherical functions. The environment maps are collected from the Internet. We randomly rotate the environment maps during training for data augmentation. And the geometric data is ray traced from randomly generated scenes with complex occlusion and materials (we use the Disney principled BRDFs [Burley and Studios 2012] for generality). In the end, we have about 400 environment maps and 3 million geometric data entries (each of them is a 2D image) in total. These 2D spherical functions are either downsampled (for environment maps) or generated (for geometric data) to a resolution of 128×128.

We use a hybrid training strategy with a pre-training step and a fine-tuning step. We first pre-train our networks with all the data, to acquire the initial weights of each network. Then for a specific scene, we perform fine-tuning to further improve our networks, in order to provide us scene-specific neural basis functions with better quality. We continue training the pre-trained model using about 30% geometric data from this specific scene, together with several environment maps (numbers are specified in Table 2). More detailed information such as training time can be found in the supplemental document.

5.2 Loss functions

We introduce several specially designed loss functions to train our representation and computation networks. First, inspired by previous work on image compression [Agustsson et al. 2019], we use VGG perceptual loss $\mathcal{L}_{\mathrm{VGG}}$ jointly with ℓ_1 loss to train the representation network. We found that the VGG loss has a significant benefit on preserving local details. Since we have to deal with HDR environment maps and glossy BRDFs, we preprocess HDR values with the μ-law compression $\mathcal{T}(\cdot)$ [Kalantari and Ramamoorthi 2017] with $\mu = 16$, inherited from the settings by Guo et al. [2019]. The final loss function $\mathcal{L}_{\mathrm{ED}}$ for training our representation network can be formally written as:

$$\mathcal{L}_{\mathrm{ED}} = \mathcal{L}_{\mathrm{VGG}}(\hat{f}, \mathcal{T}(f)) + \ell_1(\hat{f}, \mathcal{T}(f)), \tag{14}$$

where $\hat{f} = \mathcal{D}(\mathcal{E}(\mathcal{T}(f)))$.

Recall that the DPI network takes in two latent vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ and outputs a value representing the double product integral of the original spherical functions f_i and f_j. We design the loss function as follows:

$$\mathcal{L}_{\mathrm{DPI}} = \mathcal{L}_{\mathrm{rel}}\left(\mathrm{DPI}(\mathbf{z}_i, \mathbf{z}_j), \frac{\det(\mathbf{J})}{N}\sum_{n=1}^{N} f_i(\boldsymbol{\omega}_n) f_j(\boldsymbol{\omega}_n)\right), \tag{15}$$

where N is the pixel number of the images generated by the octahedral mapping [Clarberg 2008], $\det(\mathbf{J})$ is the Jacobian determinant of the spherical mapping (4π in our case), $\mathcal{L}_{\mathrm{rel}}$ is a relative loss we found that can improve the convergence of learning shadows:

$$\mathcal{L}_{\mathrm{rel}}(x, y) = \frac{|x - y|}{|y| + \epsilon}, \tag{16}$$

where $\epsilon = 0.01$ is a small constant to avoid division by zero. Similarly, in the TPI case, the loss function $\mathcal{L}_{\text{TPI}}$ can be written as:

$$\mathcal{L}_{\text{TPI}} = \mathcal{L}_{\text{rel}}\left(\text{TPI}(\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k), \frac{\det(\mathbf{J})}{N}\sum_{n}^{N} f_i(\boldsymbol{\omega}_n) f_j(\boldsymbol{\omega}_n) f_k(\boldsymbol{\omega}_n)\right). \tag{17}$$

Based on different needs for computation from different applications, we jointly train the representation network with either the DPI network or the TPI network with a combined loss of $\lambda\mathcal{L}_{\text{ED}} + \mathcal{L}_{\text{DPI/TPI}}$. We use a fixed $\lambda = 0.05$ as the weight of $\mathcal{L}_{\text{ED}}$.

After training the representation network and the DPI/TPI computation network, we train the rotation network $\mathcal{R}$ independently. Given a latent vector $\mathbf{z}$ representing a function f and the rotation representation α, we perform rotation in the latent space and calculate the loss in the 2D image space of the original spherical functions. Recall the definition of the rotation function R (Eq. (4)), the loss function for training the rotation network is:

$$\mathcal{L}_{\mathcal{R}} = \ell_1(\hat{\mathbf{z}}_\alpha, \mathbf{z}_\alpha), \tag{18}$$

where $\mathbf{z}_\alpha = \mathcal{E}(\mathcal{T}(R(f, \alpha)))$ and $\hat{\mathbf{z}}_\alpha = \mathcal{R}(\mathbf{z}, \alpha)$.

5.3 Inference/Rendering integration

Our all-frequency shading applications are implemented using the NVIDIA OptiX [Parker et al. 2010] and CUDA. During runtime, we use NVIDIA TensorRT [NVIDIA 2021] to infer neural networks. We use float16 precision for TensorRT, which does not affect the quality but will accelerate the inference. Note that the float16 format is only for inference. For fair comparison, we still use float32 to compute our storage cost.

6 RESULTS

In this section, we first validate the desired properties of our neural basis functions, including the ability of capturing all-frequency light effects ($\mathcal{P}$) and rotation in the latent space ($\mathcal{R}$). Then we demonstrate results rendered with dynamic shading using our computation networks (DPI and TPI).

We will also compare our method with other basis functions. To make fair comparisons with them, we enforce them to use the same amount of storage. For spherical harmonics (SH), we use the first seven degrees of SH basis functions. For wavelets, we keep the approximation coefficient and the first 31 largest coefficients since the wavelet coefficients must be sparsely stored together with their indices. For spherical Gaussians (SG), we use 16 SG lobes to approximate a spherical function. Each isotropic SG at least requires 4 floating numbers to specify its 2D center, 1D bandwidth and 1D amplitude. SG parameters are optimized using L-BFGS [Li et al. 2019]. We use structural dissimilarity (DSSIM) as the error metric, and we tonemapped the HDR results to sRGB for DSSIM evaluation.

6.1 All-frequency light effects

To demonstrate our representation network's ability of capturing all-frequency effects in spherical functions, we show in Fig. 3 the comparison of compression followed by reconstruction among our encoder/decoder network, SH, SG, and wavelet. Using equal storage, SH produces ringing artefacts when using equal (1×) storage since its degree is not high enough, SG leads to over smoothed

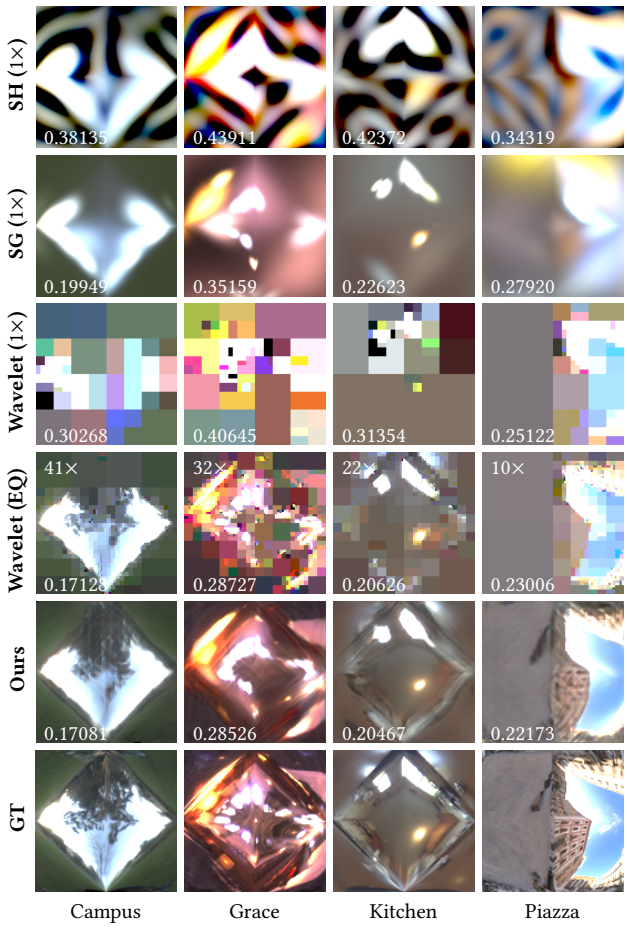

Figure 3: Comparison of compression followed by reconstruction among our representation network, SH, SG, and wavelet. The reconstructed spherical functions using our representation network match the original functions more accurately while others have their own issues.

images without high-frequency details, and the highly compressed wavelet loses most details and shows severe blocky artifacts. In contrast, the reconstructed spherical functions using our neural basis functions match the original functions more accurately. Achieving equal quality (EQ) using wavelet requires up to 41× more storage compared with ours.

6.2 Rotation

As shown in Fig. 4, with a latent vector and a $SO(3)$ rotation matrix, our rotation network can perform rotation operation in the latent space. We compare the spherical functions reconstructed from the rotated latent vectors ("Decoder w/ $\mathcal{R}$" column) with the rotated spherical functions ("GT" column), and our results accurately match the ground truth. We also feed the rotated spherical functions to the encoder as input and then forward the resulting latent vectors to the decoder. The reconstruction results ("Decoder w/o $\mathcal{R}$" column) are close to ours, indicating that our trained representation network

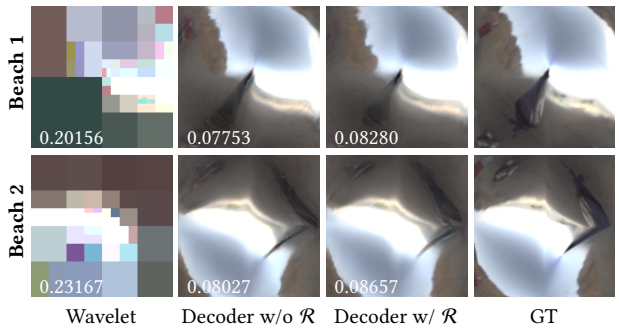

Figure 4: We compare the spherical functions reconstructed in different ways. The results indicate that our representation network forms a meaningful latent space that supports efficient rotation, leading to better quality than wavelet.

can generate meaningful latent vectors that support efficient rotation. Compared to the wavelet results at equal storage, our method leads to significantly better image quality and temporal stability. During rendering, the rotation will be applied on the latent vector of the original, unrotated environment lighting. We show animated rotation results in the supplemental video.

6.3 All-frequency shading by computation networks

We compare our rendering results with other non-neural baselines using spherical functions represented by SH, SG, and wavelet. All the rendering results are generated using a desktop with an NVIDIA 3090 GPU. The scene configurations and performance statistics are summarized in Table 2.

DPI network. In Fig. 5, we compare rendering results using our DPI network with images rendered using SH and wavelet in all-frequency shading. With equal storage, a small number of SH coefficients cannot capture high-frequency shadows, and highly compressed wavelet coefficients suffer from significant artifacts. In contrast, our method accurately reproduces the high-frequency shadows under a sharp spotlight.

TPI network. In Fig. 6, we compare rendering results using our TPI network with images rendered using wavelet and SG basis functions in dynamic relighting. The figure in the top right corner shows the reconstructed environment lighting of each method. Computing a new set of wavelet coefficients after rotating a spherical function is expensive. We need first to rotate the spherical function explicitly and then project the rotated function onto wavelet basis. Since wavelet compression only keeps the largest coefficients and truncates the rest, wavelet coefficients at consecutive frames may change significantly, causing severe flickering artifacts. Flickering is even more evident when using highly compressed wavelet coefficients in our case. As for spherical functions represented by SG, they can be rotated efficiently. However, with insufficient SG lobes, it cannot accurately capture high-frequency effects such as shadows and glossy highlights. In contrast, our neural basis functions and TPI network can generate high-quality images in real-time frame rates. Even with dynamic environmental lighting, our rendering results are temporally stable (shown in the supplemental video).

Table 2: The scene configurations and performance statistics. Our runtime performance of each scene meets the real-time requirements. The running time of the rotation network is negligible since the rotation is only operated on lighting.

Scene	Figure	#Valid pixels	#Trained lighting	DPI (ms)	TPI (ms)	FPS
Plant (teaser)	Fig. 1	249324	2	5.7	N/A	175.4
Teapot	Fig. 1	316830	12	N/A	7.4	135.4
Dragon	Fig. 6	336788	12	N/A	7.5	130.8
San Miguel	Fig. 6	920130	12	N/A	19.8	50.6
Plant	Fig. 5	184163	1	5.0	N/A	199.3

Performance. The performance numbers are summarized in Table 2. Thanks to our lightweight computation networks, we can achieve real-time performance. Note that the running time is linearly scaled by the number of valid pixels (with traced rays hitting the objects). We only count the inference time.

We also need to point out that our network inference is *indeed slower* than traditional basis functions (such as the dot product of SH coefficients) for equal storage comparison. Nevertheless, we believe that it is not easy to conduct a better comparison than equal storage, even it exposes both the advantage (better quality) and the disadvantage (slower performance) of our method. Making equal-quality comparisons or equal-quality comparisons needs to use a prohibitively large number of terms for traditional basis functions.

6.4 Discussion and limitations

Relation to precomputed radiance transfer (PRT). Though we choose all-frequency shading applications to demonstrate the capacity and the efficiency of our neural basis functions, our focus is fundamentally different from PRT research. PRT *makes use of* existing basis functions, while we *design a new type* of basis function completely in its compressed form with desired computational properties, and only use all-frequency shading to demonstrate possible uses of our neural basis functions.

Therefore, we believe that our research is orthogonal to PRT. We do not extend further discussion on PRT methods, especially those not using basis functions [Ren et al. 2013]. While we are fully aware that PRT has its own issues, we do not attempt to address them, e.g., enabling dynamic camera movement in relighting. However, since moving camera essentially requires precomputing more BRDF slices (Sec. 3), we believe our method can be extended to solving it. This involves fine-tuning with a larger set of 2D BRDF slices, and compressing them into a dictionary of latent vectors. During rendering, the corresponding latent vectors of the BRDF slices will be queried and used in follow-up computations as usual. Since projecting 2D functions onto our neural basis functions is fast enough, the precomputation can be performed in a reasonable amount of time in practice.

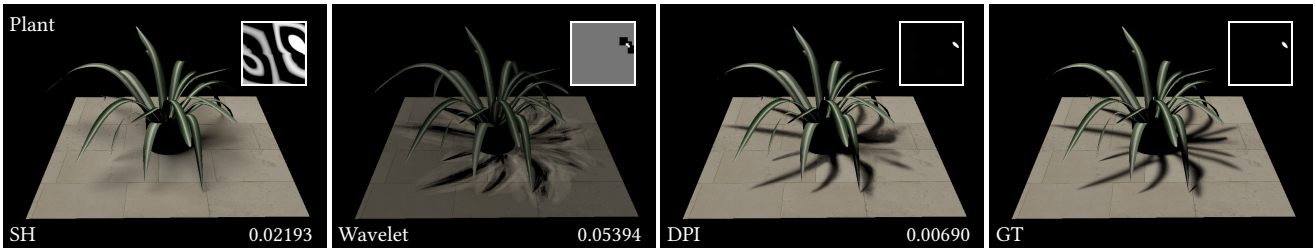

Figure 5: Comparison between our DPI network, SH and wavelet in all-frequency shading. The figure on the top right corner shows the reconstructed environment lighting of each method. Our method accurately reproduces the high-frequency shadows under a sharp spot light, while SH is failure to capture high-frequency shadows, and wavelet suffer from significant artifacts.

Figure 6: Comparison between our TPI network, wavelet and SG in dynamic relighting. Wavelet coefficients at consecutive frames may change greatly, causing severe flickering artifacts like flickering colors (red arrow) and blocky shadows (blue arrow). Rotating SG is efficient. But with insufficient lobes, it cannot accurately capture high-frequency effects such as shadows and glossy highlights. In contrast, our neural basis functions can generate high-quality images in real-time frame rates.

Artifacts. Because we use short latent vectors to compress 2D spherical functions and render images with lightweight computation neural networks, there is inevitable information loss. When we reconstruct the original spherical functions, typical artifacts such as grid patterns and blurred signals caused by neural networks are noticeable (e.g., comparing the last two rows in Fig. 3).

In addition, the computation networks (DPI, TPI, and $\mathcal{R}$ networks) produce more noticeable artifacts such as the blurred high-frequency shadows and missing specular reflection, as can be seen in our supplemental document and the accompanying video. We believe that these artifacts can be better suppressed using smarter loss functions, advanced architectures, and more data.

Generalizability. We have demonstrated that our method has a certain extent of generality by pre-training on a large dataset. But our neural basis functions do rely on fine-tuning at this stage to achieve the best quality on a specific scene. Therefore, how to further generalize our neural basis functions to completely avoid fine-tuning is an interesting research direction in the future.

7 CONCLUSION AND FUTURE WORK

We have presented neural basis functions, a learning-based technique that can significantly compress spherical functions as well as perform efficient double/triple product integrals and rotation. We train a representation network to compress an input 2D spherical function into a latent vector that acts as coefficients of implicit basis functions, and we design lightweight computation networks to perform subsequent computations completely in the latent space. With the help of these computation networks, we achieve the desired key properties of basis functions, including high compression rate, all-frequency representation, efficient rotation, and practical evaluation of double/triple product integrals, although we rely on fine-tuning on specific scenes. We show plausible results of all-frequency shading in real-time frame rates ($\geq$ 30 frames per second), but we do have a weakness in inference speed which is slower than traditional basis functions.

In the future, it is straightforward to train more specific neural networks to support new effects such as dynamic material editing [Xu et al. 2007]. It would also be interesting to extend our neural basis functions to enable more applications like affine TPI [Sun and Ramamoorthi 2009] and multiple product integrals [Sun and Mukherjee 2006]. Furthermore, as pointed out earlier, our usage of neural networks is specially tailored for rendering, so it is crucial for them to be lightweight. Therefore, we want to draw attention to this kind of lightweight neural networks used in rendering [Müller 2021]. And we hope to stimulate research on GPU architectures or low-level programming models for faster inference and training.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2020YFB1708903), the National Natural Science Foundation of China (61872223). Ling-Qi Yan is supported by gift funds from Adobe, Dimension 5 and XVerse.

REFERENCES

Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. 2019. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 221–231.

Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *arXiv preprint arXiv:2007.09892* (2020).

Brent Burley and Walt Disney Animation Studios. 2012. Physically-based shading at disney. In *ACM SIGGRAPH*, Vol. 2012. vol. 2012, 1–7.

Brian Cabral, Nelson Max, and Rebecca Springmeyer. 1987. Bidirectional Reflection Functions from Surface Bump Maps. *Computer Graphics (Proceedings of SIGGRAPH)* (1987), 273–281.

Dan A. Calian, Jean-François Lalonde, Paulo Gotardo, Tomas Simon, Iain Matthews, and Kenny Mitchell. 2018. From Faces to Outdoor Light Probes. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13341

Petrik Clarberg. 2008. Fast equal-area mapping of the (hemi) sphere using simd. *Journal of Graphics Tools* 13, 3 (2008), 53–68.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2018. Spherical cnns. *arXiv preprint arXiv:1801.10130* (2018).

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 2018. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 52–68.

Jiahui Fan, Beibei Wang, Miloš Hašan, Jian Yang, and Ling-Qi Yan. 2022. Neural Layered BRDFs. In *Proceedings of SIGGRAPH 2022*.

Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. *CoRR* abs/2103.10380 (2021). arXiv:2103.10380

Jonathan Granskog, Fabrice Rousselle, Marios Papas, and Jan Novák. 2020. Compositional Neural Scene Representations for Shading Inference. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020).

J. Guo, L. I. Mengtian, L. I. Quewei, Y. Qiang, H. U. Bingyang, Y. Guo, and L. Q. Yan. 2019. GradNet: Unsupervised Deep Screened Poisson Reconstruction for Gradient-Domain Rendering. *ACM Transactions on Graphics* 38, 6 (2019), 223.1–223.13.

Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. Frequency Domain Normal Map Filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 26, 3 (2007), 28:1–28:12.

Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. 2020. DeepBRDF: A deep representation for manipulating measured BRDF. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 157–166.

Nima Khademi Kalantari and Ravi Ramamoorthi. 2017. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017).

Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker. 2019. Inverse Rendering for Complex Indoor Scenes: Shape, Spatially-Varying Lighting and SVBRDF from a Single Image. (2019).

R Marques, C Bouville, and K Bouatouch. 2022. Gaussian Process for Radiance Functions on the S 2 S^2 Sphere. In *Computer Graphics Forum*. Wiley Online Library.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*. Springer, 405–421.

Thomas Müller. 2021. Tiny CUDA Neural Network Framework. https://github.com/nvlabs/tiny-cuda-nn.

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, H-P Seidel, and Tobias Ritschel. 2017. Deep shading: convolutional neural networks for screen space shading. In *Computer graphics forum*, Vol. 36. Wiley Online Library, 65–78.

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2003. All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 22, 3 (2003), 376–381.

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2004. Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 23, 3 (2004), 475–485.

NVIDIA. 2021. NVIDIA TensorRT. https://developer.nvidia.com/tensorrt

Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. 2010. Optix: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 29, 4 (2010), 1–13.

Gilles Rainer, Adrien Bousseau, Tobias Ritschel, and George Drettakis. 2022. Neural Precomputed Radiance Transfer. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 365–378.

Ravi Ramamoorthi and Pat Hanrahan. 2001a. A Signal-Processing Framework for Inverse Rendering. *SIGGRAPH* (2001), 117–128.

R Ramamoorthi and P Hanrahan. 2001b. An Efficient Representation for Irradiance Environment Maps. *SIGGRAPH* (2001), 497–500.

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global illumination with radiance regression functions. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Helge Rhodin, Mathieu Salzmann, and Pascal Fua. 2018. Unsupervised geometry-aware representation for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 750–767.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The state of the art in interactive global illumination. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 160–188.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 527–536.

Bo Sun and Ravi Ramamoorthi. 2009. Affine Double and Triple Product Wavelet Integrals for Rendering. *ACM Transactions on Graphics* 28, 2 (2009), 14:1–14:17.

Weifeng Sun and Amar Mukherjee. 2006. Generalized wavelet product integral for rendering dynamic glossy objects. *Acm Transactions on Graphics* 25, 3 (2006), 955–966.

Alejandro Sztrajman, Alexandros Neophytou, Tim Weyrich, and Eric Sommerlade. 2020. High-Dynamic-Range Lighting Estimation From Face Portraits. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 355–363.

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural BRDF Representation and Importance Sampling. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 332–346.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. 2020. State of the art on neural rendering. 39, 2 (2020), 701–727.

Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. 2009. All-frequency rendering of dynamic, spatially-varying reflectance. (2009), 1–10.

Tuanfeng Y Wang, Tobias Ritschel, and Niloy J Mitra. 2018. Joint material and illumination estimation from photo sets in the wild. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 22–31.

Stephen H Westin, James R Arvo, and Kenneth E Torrance. 1992. Predicting Reflectance Functions from Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH)* (1992), 255–264.

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hanggao Xin, Shaokun Zheng, Kun Xu, and Ling-Qi Yan. 2020. Lightweight Bilateral Convolutional Neural Networks for Interactive Single-bounce Diffuse Indirect Illumination. *IEEE Transactions on Visualization & Computer Graphics* 01 (2020), 1–1.

Kun Xu, Yue Gao, Yong Li, Tao Ju, and Shi-Min Hu. 2007. Real-time Homogenous Translucent Material Editing. *Computer Graphics Forum* 26, 3 (2007), 545–552.

Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic Spherical Gaussians. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 32, 6 (2013), 209:1–209:11.

Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5745–5753.

Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Miloš Hašan, and Ling-Qi Yan. 2021. Neural complex luminaires: representation and rendering. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–12.