Neural Layered BRDFs

Jiahui Fan
Nanjing University of Science and Technology
China
fjh@njust.edu.cn

Beibei Wang†
Nankai University,
Nanjing University of Science and Technology
China
beibei.wang@nankai.edu.cn

Miloš Hašan
Adobe Research
USA

Jian Yang†
Nanjing University of Science and Technology
China
csjyang@njust.edu.cn

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

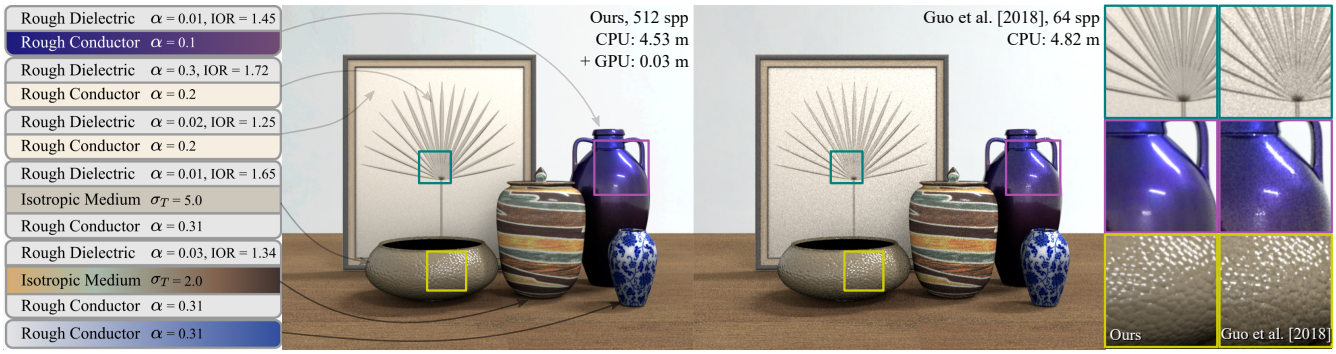

Figure 1: We present a neural latent representation for BRDFs and a BRDF layering network based on it. Our method is able to produce closely matching layered results to the Monte Carlo simulation in Guo et al. [2018] with less cost, and works well with spatially-varying parameters.

ABSTRACT

Bidirectional reflectance distribution functions (BRDFs) are pervasively used in computer graphics to produce realistic physically-based appearance. Many common materials in the real world have more than one layer, like wood, skin, car paint, and many decorative materials. However, precise simulation of layered material optics is non-trivial. The most accurate approaches rely on Monte Carlo random walks to simulate the light transport within the layers, leading to high variance and cost. Other approaches are efficient, but less accurate. In this paper, we propose to perform layering in the neural space, by compressing BRDFs into latent codes via a proposed representation neural network, and performing a learned layering operation on these latent vectors via a layering network. Our BRDF evaluation is noise-free and computationally efficient, compared to the state-of-the-art approach; it is also a first step towards a "neural algebra" of operations on BRDFs in a latent space.

†Corresponding authors. Email: beibei.wang@nankai.edu.cn.
†Corresponding authors. Email: csjyang@njust.edu.cn.
Jiahui Fan, Beibei Wang and Jian Yang are with PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, School of Computer Science and Engineering.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530732

CCS CONCEPTS

• **Computing methodologies** → **Reflectance modeling**; **Ray tracing**; **Appearance and texture representations**.

KEYWORDS

BRDF, Layering

ACM Reference Format:
Jiahui Fan, Beibei Wang†, Miloš Hašan, Jian Yang†, and Ling-Qi Yan. 2022. Neural Layered BRDFs. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3528233.3530732

1 INTRODUCTION

Bidirectional reflectance distribution functions (BRDFs) are pervasively used in computer graphics to produce vivid and realistic appearance. Layered materials are common in the real world: car paints, skin and wood are just a few examples. The BRDFs of layered materials are more complex to render than single-layer materials, since the light might bounce several times within the layers before exiting the surface. The challenge of representing and rendering layered materials has attracted a large amount of research.

Existing approaches including approximate analytic models [Weidlich and Wilkie 2007] have been proposed to solve this problem. Among them, the Monte Carlo simulated approaches are most accurate, at the cost of expensive random walk and high variance, which makes them less practical. The other approaches are either less accurate or require large memory footprints.

In recent years, neural networks for representing spatially varying bidirectional reflectance distribution functions (SVBRDFs) and bidirectional texture functions (BTFs) have received attention ([Rainer et al. 2019], [Rainer et al. 2020], [Kuznetsov et al. 2021], and [Takikawa et al. 2021]). These neural approaches mostly focus on efficient compression ([Rainer et al. 2019], [Rainer et al. 2020], [Takikawa et al. 2021]) and query ([Kuznetsov et al. 2021]). They have greatly reduced the storage overhead, successfully making high-dimensional SVBRDF and BTF data practically usable in rendering. However, none of these methods consider layered materials.

We propose to represent both analytic and measured BRDFs with a latent space, and then perform a layering operation between the latent vectors with a layering network. The layering neural network replaces the expensive random-walk, resulting in a simple, noise-free and computationally efficient evaluation scheme for layered BRDFs.

2 RELATED WORK

Neural based SVBRDF/BTF/BRDF compression. SVBRDF/BTF compression shares the same sense that compressing high dimensional data into compact representations with our method. Recently, Kuznetsov et al. [2021] represent and render a variety of material appearances at different scales, and Sztrajman et al. [2021] represent each BRDF with a decoder structure, resulting in better quality, at the cost of more storage for each BRDF. We categorize the aforementioned approaches as *specialized methods*, because one neural network only represents one material/BRDF in these works, resulting in high quality, but heavy storage overhead and difficulty for operations.

The other kind of approaches are *generalized methods*. Rainer et al. [2019] proposed a neural representation for BTFs, using a latent vector to represent each texel, but requiring to train an autoencoder per BTF and does not generalize across materials. Later, Rainer et al. [2020] introduced a unified model to represent all materials, with lower quality as a trade-off. Specifically, neither of them work well for highly specular materials, compared to our method. Some other researches focus on compressing single BRDFs, instead of SVBRDFs/BTFs into latent representation. Hu et al. [2020] utilize a convolutional neural network to compress and reconstruct measured BRDFs slices and manage to edit specific attributes in the latent space. Also, there is a concurrent work by Zheng et al. [2021], using neural processes to compactly represent measured BRDFs and classifiers to qualitatively tweak certain attributes of the BRDFs.

BRDF layering operations. Layering is an important operation for BRDFs, which has been addressed by several lines of work. Some Fourier-based methods ([Jakob et al. 2014], [Jakob 2015], and [Zeltner and Jakob 2018]) rely on expensive computation per parameter setting, which makes it difficult to handle spatially-varying textures. The Monte Carlo based methods ([Guo et al. 2018], [Gamboa et al. 2020], and [Xia et al. 2020]) are able to produce high-quality results, and support spatially-varying textures, thus we treat them as ground-truth. However, the required random walks lead to extra variance (noise). Tracking directional statistics ([Belcour 2018], [Yamaguchi et al. 2019], and [Weier and Belcour 2020]) expresses the directional statistics (e.g., mean and variance) of a layered BRDF and track the statistical summary at each step, resulting in real-time performance. However, they are a fundamentally approximate way of representing the underlying functions. Compared to all of these works, our method represents a BRDF with a latent vector, and performs the operations on the latent vectors. Our layering results are very close to Monte Carlo based approaches, avoiding the expensive random walk and additional variance (noise).

3 NEURAL BRDF REPRESENTATION AND LAYERING

In this section, we present our solution to neural BRDF representation and the layering operation. We first formulate the problem (Sec. 3.1). Then, we show how to represent BRDFs with our general-purpose BRDF decoder structure (Sec. 3.2). Finally, we introduce a layering neural network to produce layered materials (Sec. 3.3).

3.1 Overview and formulation

We focus on representing individual BRDFs and providing a layering operation on them. A BRDF is a 4D function $f(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$, where $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ are the incoming and outgoing directions on the unit hemisphere. Note that this definition can be extended to bidirectional scattering distribution functions (BSDFs) by considering full unit spheres for directions. For simplicity, we do not represent two-sided BSDFs in this paper. However, we do implicitly consider BSDFs when layering one BRDF atop another: the top BRDF is assumed be the reflective component of a BSDF, i.e., to transmit all energy that is not reflected, and this affects the layering operation.

We will start from compressing any BRDF in a compact neural form:

$$\textbf{Representation: } f(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \xrightarrow{N_{\text{rep}}} \boldsymbol{V}_f, \tag{1}$$

where $\boldsymbol{V}_f$ is known as a latent vector and N_{rep} is a neural representation projecting operator. This operator is implemented through optimization (searching for a latent vector that decodes to the input BRDF), as we introduce in later chapters.

Once the BRDFs are represented as latent vectors, we treat them as operands, and provide a layering operator that act upon them:

$$\textbf{Layering: } \{\boldsymbol{V}_{\text{top}}, \boldsymbol{V}_{\text{bottom}}, A, \sigma_T\} \xrightarrow{N_{\text{layering}}} \boldsymbol{V}_{\text{layered}}. \tag{2}$$

Above, $\boldsymbol{V}_{\text{top}}$ and $\boldsymbol{V}_{\text{bottom}}$ represent the latent vectors for BRDFs at

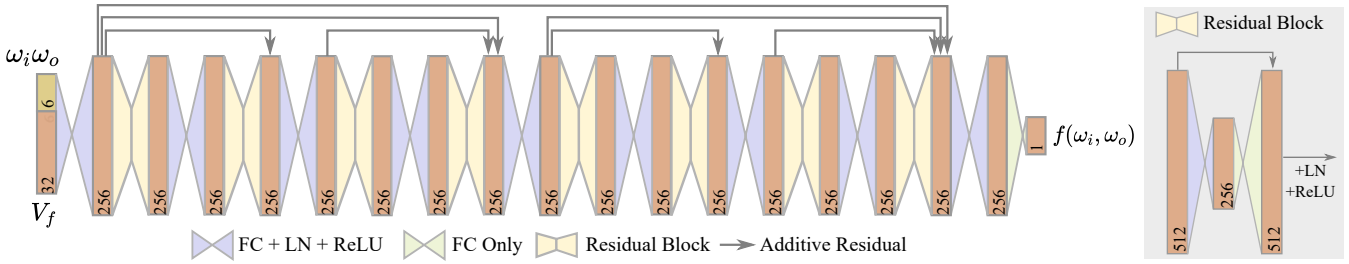

Figure 2: The structure details of the evaluation network. All residuals and skip connections are added before the normalization and activation layers.

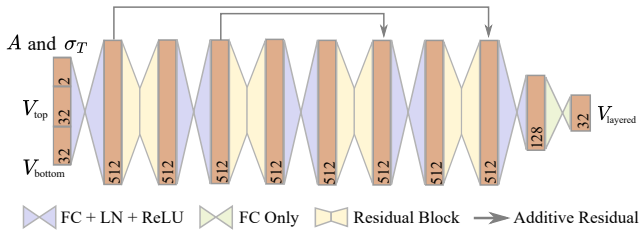

Figure 3: The structure of layering network. A and σ_T are scalars that represent the albedo and the extinction coefficients. V_{top} and V_{bottom} are the latent vectors to represent the top layer and the bottom layer respectively.

the top layer and the bottom layer, respectively. A and σ_T are the single-scattering albedo and extinction coefficients for the participating medium inserted between the two layers.

3.2 Neural BRDF representation

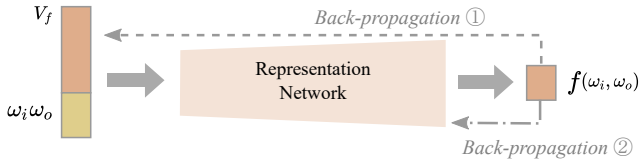

Figure 4: The high-level architecture of the neural BRDF representation. The network only includes a decoder, which is used for both BRDF evaluation and representation.

We would like to find a general-purpose neural network that is able to compress any input BRDF $f(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ into a latent vector $\boldsymbol{V_f}$. We do not want to pre-specify the discretization of the input BRDF, as there is no single satisfactory discretization. Instead, we opt to define an *evaluation* network architecture, which takes a latent vector as well as incoming and outgoing directions as input, and returns the corresponding BRDF value. To project a BRDF into the latent space, we simply optimize for a latent vector that gives back the input BRDF at any desired discretization.

We design the architecture of our Neural BRDF evaluation network as shown in Figure 4. It takes a latent vector and an incoming-outgoing pair as query, and outputs the corresponding BRDF value. The network is trained by back-propagation on a dataset of BRDFs, as detailed below.

Note that we have two back-propagation routes, one for the latent vector, the other for the weights of the evaluation network. When we are training the network, we use all values of BRDFs across the dataset, and we back-propagate the gradient through both routes ① and ②, updating the weights in the network and the latent vectors simultaneously. In this way, our network learns to use different latent vectors to represent different BRDFs. Once we have trained the evaluation network, to project any new BRDF into the latent space, we freeze the network parameters and only back-propagate to the latent vector via route ①. This projection takes about 10 to 45 seconds to converge on an RTX 2080Ti GPU. We illustrate the detailed architecture of our evaluation network in Fig. 2. We treat every channel of the RGB color space separately, and we do not clamp any high dynamic range values.

We do not require the projected BRDFs to be parametric or measured, nor do we care whether they are already layered or not. We do however make two assumptions for simplicity. First, we assume that all BRDFs are isotropic for now, mainly for the efficiency of the training dataset. Second, we assume that none of the BRDFs are normal-mapped; this operation is harder to learn and easier to achieve by altering local shading coordinates during rendering.

Our evaluation network can successfully represent BRDFs from commonly seen materials. We can also define a *latent texture*, where each texel is a latent vector representing a BRDF. In this way, we can use this latent texture to describe SVBRDFs, as will be demonstrated on more examples in Sec. 5.

Discussion. In Table 1, we compare our method against three related works, regarding the representation accuracy for specularity, ability to represent SVBRDFs, and the generality of the model. Regarding representation accuracy, both Sztrajman et al. [2021] and our method are able to handle sharp specular BRDFs, while Rainer et al. [2019] and Rainer et al. [2020] are less accurate. However, Sztrajman et al. [2021] require a decoder for each BRDF representation, which means that when a spatially varying BRDF is processed, they need to prepare all kinds of different decoders for each texel, bringing significant storage overhead. For a $4K$ resolution texture as example, Sztrajman et al. [2021] store $4K \times 4K \times 32$ floats, and decode them into $4K \times 4K \times 675$ floats. Then a dynamic loading and permuting process of the floats into network weights at run-time is also required, which is difficult for parallelization in GPU. In

Table 1: Comparison of different BRDF representation methods, considering the ability to preserve specularity, the cost storing and evaluating SVBRDFs and the cost obtaining a new BRDF representation. A quantitative comparison between Sztrajman et al. [2021] and our method is discussed in Section 3.2 below. (* need to decompress all latent vectors to decoders for evaluation.) (⋆ then compress the new decoder into a latent vector.)

Method	Spec. preservation	To store SVBRDFs	To evaluate SVBRDFs	To represent a new material
Rainer et al. [2019]	✗	one latent vector per BRDF	one decoder per SVBRDF	Train a new en/decoder
Rainer et al. [2020]	✗	one latent vector per BRDF	one decoder for all BRDFs	Run the encoder
Sztrajman et al. [2021]	✓	one latent vector per BRDF*	one decoder per BRDF	Train a new decoder⋆
Ours	✓	one latent vector per BRDF	one decoder for all BRDFs	Optimize the latent vector

contrast, our method only requires $4K \times 4K \times 32 \times 3$ floats plus a generalized decoder of $1M$ floats in total. Regarding the generality, Sztrajman et al. [2021] require network training for each BRDF, and Rainer et al. [2019] require training for each single SVBRDF, while our method only require a latent vector optimization per BRDF.

3.3 Neural BRDF layering

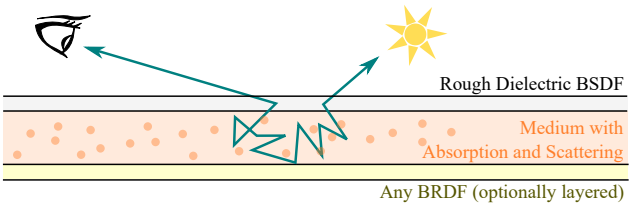

Figure 5: Our configuration for layered BRDFs includes a top layer using a rough dielectric, a bottom layer using any BRDF and a homogeneous participating media in the middle.

The layering operation on BRDFs (Figure 5) includes complex light transport interactions among the layers. The most accurate way to layer BRDFs is using a Monte Carlo random walk [Guo et al. 2018], but this is expensive, especially when there are dense volumetric media between the interfaces; the random walk also introduces variance.

We instead propose to learn the layering operation in the latent space. We consider a two-layer operation, consisting of a top layer with a rough dielectric BSDF (whose transmission component is implied as the energy complement of the top BRDF, with no energy lost in the interface itself) and a bottom layer with any BRDF, and a layer of homogeneous participating media with albedo A and extinction coefficient σ_T (assuming isotropic scattering) between the interfaces. We do not use anisotropic scattering nor do we compensate for energy loss in microfacet BRDFs, though these effects could be added to the training data. Note that our two-layer setup can handle multi-layered configurations by recursively applying the layering operation, as we discuss in Sec. 5.1.

Our layering network has a similar MLP-based structure to the evaluation network, taking two latent vectors which represents the top layer and bottom layer as input, together with the albedo and extinction coefficients of the volumetric medium. It directly outputs one target latent vector to represent the layered BRDF. Similar to the evaluation network, our layering network also deals with RGB channels independently. The detailed structure of our layering network is shown in the Fig. 3.

Table 2: Different distributions that we use to sample the parameter space of BRDFs. $\mathcal{U}(x, y)$ represents a continuous uniform distribution in the interval (x, y), and $\mathcal{V}(X)$ is a discrete uniform random variable in a finite set X.

Parameter		Sampling Function
Roughness	α	$\mathcal{U}(0.216, 1)^3$
IOR	η	$\mathcal{U}(1.05, 2)$
Fresnel	R_0	$\mathcal{U}(0, 1)$
Scattering albedo	A	$1 - \mathcal{U}(0, 1)^2$
Extinction coefficient	σ_T	$\mathcal{V}(\{0, 1, 2, 5\})$

Our representation allows several further operations beyond layering, such as interpolation, mipmapping and importance sampling in rendering. See the supplementary material for details.

4 DATA, TRAINING AND RENDERING

4.1 Dataset

We use the Mitsuba renderer [Jakob 2010] to generate the training dataset of BRDFs, consisting of rough conductors, rough dielectrics and layered BRDFs [Guo et al. 2018]. The dataset could be enriched with other types of BRDFs as needed for the application. All BRDFs have a single channel; RGB color is achieved by independently processing each channel with varying parameters. We generate 300 rough conductor BRDFs and 300 rough dielectric BRDFs, together with 12,720 layered BRDFs [Guo et al. 2018] by randomly layering them into 2 layers; additionally, we also generate 1,800 three-layer BRDFs, which means that their bottom layers are already layered BRDFs. We use the three-layer BRDFs to finetune the layering network, after it has been first trained with two layers, as described in Sec. 5.1.

Each rough dielectric BRDF has two parameters: roughness α_1 and the index of refraction (IOR) η. Each rough conductor BRDF also has two parameters: roughness α_2 and a Schlick Fresnel approximation with R_0 controlling the reflectance at 0 degrees. Therefore, each two-layer BRDF has six parameters: α_1, η, α_2, R_0, the albedo A and the extinction coefficient σ_T. In our implementation, we use the GGX model as the normal distribution function at interfaces, and only consider isotropic BRDFs, although other effects could be included. We use the Henyey-Greenstein phase function

($g = 0$) for all the medium within the layered BRDFs. All layers in layered BRDFs have unit thickness, and these appearances can be adjusted via the extinction coefficients. The sampling distributions of different parameters are shown in Table 2.

For each BRDF, we sample 25^4 pairs of incoming and outgoing directions. Since we only consider the reflection, we perform a stratified sampling along the elevation angle θ and azimuth angle φ on the upper hemisphere, where $\theta \in [0, \frac{\pi}{2})$ and $\varphi \in [0, 2\pi)$. For each sampled incoming and outgoing direction pair, we compute the BRDF value via microfacet model for rough conductor/dielectric BRDFs or Guo et al. [2018] for layered BRDFs. Then we store the incoming direction, outgoing direction and the BRDF value (without the cosine term).

Representation network. We use 12,000 two-layer BRDFs as our training set, and the other 720 two-layer BRDFs as validation. Note that although we do not use rough conductor/dielectric BRDFs to train this network, they can still be represented, because their appearances could be regarded as special cases of layered BRDFs for certain parameters.

Layering network. We project 12,720 two-layer BRDFs and their components (300 rough conductor and 300 rough dielectric BRDFs) from the dataset into the latent space with the trained representation network. Then these projected latent vectors are used directly as supervision to train the layering network, with the same proportions of training and validating set as in the representation network. The 1,800 three-layer BRDFs are later used to finetune this network.

4.2 Training

The representation network is trained first, as the layering network relies on the representation network.

Representation Network. During training this network, the shared network parameters and the latent vectors of current input BRDFs are updated simultaneously in every iteration. We calculate the L_1 loss, which better preserves color and avoids artifacts, compared to the L_2 loss, according to our experiments. The L_1 loss is defined as: $Loss = \frac{1}{N}\sum_N |f^{\text{pred}} - f^{\text{gt}}|$, where N denotes the number of BRDFs in a batch, f^{pred} and f^{gt} represents BRDF values output by our network and the ground-truth. We use learning rates 3×10^{-4} for the network weights and 1×10^{-4} for the mutable latent vectors in training set. Both learning rates decay by 0.9 after every epoch. We initialize all the latent vectors to 1, and we train the network for 50 epochs in about 40 hours on an RTX 2080Ti GPU.

Layering Network. We supervise this network with latent vectors as both inputs and outputs, via the latent vectors obtained from our trained representation network, and optimize it by the L_1 loss with the initial learning rate of 3×10^{-3}. The learning rate decays by 0.7 for every 50 epochs. It takes us about 10 hours to train this network on an RTX 2080Ti GPU for 1,000 epochs. Additionally, we finetune the layering network with 1,800 three-layer BRDFs.

4.3 Rendering pipeline

For rough conductor/dielectric BRDFs, we represent them by the latent vector using our representation network; for layered BRDFs, we perform the layering operation and get the layered latent vector; for SVBRDFs, we prepare a latent texture, where each texel stores a latent vector.

Now, we use our Neural BRDF models in rendering by path tracing. First, we store all incoming and outgoing directions and lighting values of each ray at all intersections into buffers. Second, for each buffer pixel with the Neural BRDF type, we infer the representation network for the BRDF value. Finally, we calculate the radiance of path tracing to get output images, applying specific reconstruction filters, such as Gaussian filters.

We also integrate our implementation into the MIS framework. According to the different sampling strategies (light sampling, BRDF sampling or MIS), we store all queries and the pdf information for each part above into buffers. Then we calculate the reflectance via our network and finally combine the weighted results, if needed. To get the importance sampling pdf for BRDFs under our neural representation, we train another small network to predict a pdf proxy which has a Gaussian lobe and a Lambertian lobe. See the supplementary material for more details of this network.

In order to accelerate GPU inference, we implement the inference in CUDA via NVIDIA Cutlass CUDA Templates and compile it into Python libraries. Thanks to the optimizations in Cutlass, there is no obvious increase in the time cost when the film resolution rises, as long as it does not run out of the GPU memory. We compile several libraries for different buffer sizes in advance, and dynamically decide which to use during rendering. Eventually, a single BRDF evaluation per pixel with resolution 1920×1080 costs 5 ms.

5 RESULTS

We have implemented our method inside the Mitsuba renderer [Jakob 2010] and compared our method with previous works, including Guo et al. [2018] and Belcour [2018]. Since the method by Guo et al. [2018] does not introduce any approximations other than Monte Carlo noise, we use it as the reference. All the implementations are taken from the authors' websites. All timings in this section are measured on an Ubuntu Linux workstation with an Intel Xeon E5-2650 v4 @ 2.20GHz CPU (8 cores), 64 GB of main memory and an RTX 2080Ti GPU (11 GB).

5.1 Quality validation

We first validate our layering operation, then demonstrate rendering results on more complex scenes.

Layering network. In Figure 6, we compare results from our layering network against Guo et al. [2018] (as reference with a high sampling rate) across different BRDF configurations: different roughness for the top and bottom layers and varying scattering albedos for the medium in between. Our results are close to the reference. In Figure 7, we compare our layered materials with Belcour [2018] and Guo et al. [2018]. Our method produces results close to the reference and with less noise, and our shading speed is faster than Belcour [2018] (though note that we utilize the GPU for network inference). This confirms the effectiveness of our layering network.

Our layering network can be applied recursively to obtain materials with multiple (3 or more) layers. In Figure 8, we show the results of BRDFs with more layers using our layering network, and we find our result still close to the reference. Our layering network can also work for measured BRDFs, even though they are totally unseen for the network during training stage. In Figure 9, we show the

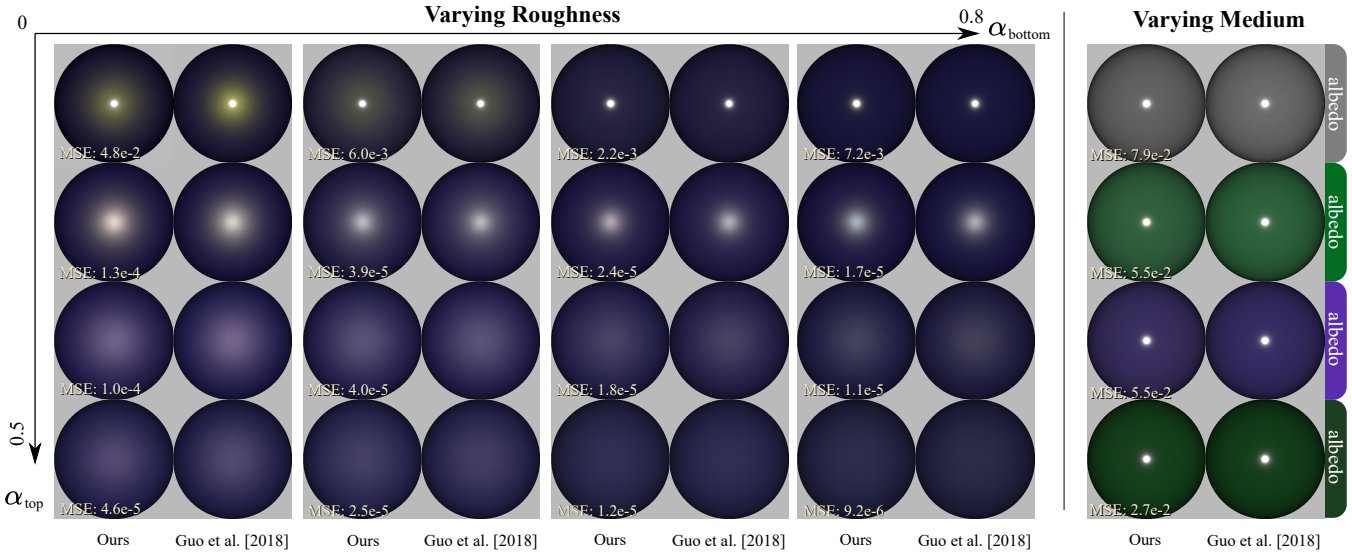

Figure 6: Comparison of outgoing radiance between our layering model and Guo et al. [2018] on varying roughness for both top and bottom layers and varying scattering albedos for the medium. Our layering network produces results close to the reference.

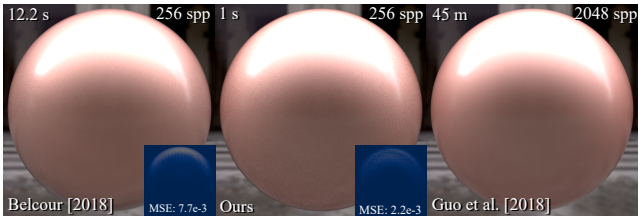

Figure 7: Comparison between Belcour [2018], our method and Guo et al. [2018] (reference) and their shading time. With equal number of samples, our method produces closer result to the reference, as compared to Belcour [2018]. All three methods have the same cost for path tracing per sample, while our shading time is shorter.

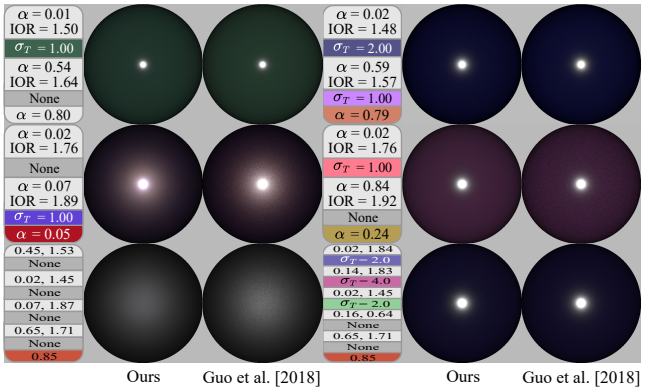

Figure 8: Comparing the visualization of outgoing radiance of our layering results and those from Guo et al. [2018] for BRDFs with multiple (3 to 6) layers. We apply our layering network consecutively from bottom to up. Though error will accumulate, our results are still close to the reference.

results of our layering network (without any fine-tuning) with measured BRDFs in MERL dataset [Matusik 2003], which demonstrates that our method can be applied effectively in unseen measured materials.

5.2 Complex scenes

In Table 3, we report the scene settings (and additional equal time MSE comparison with Guo et al. [2018]). Please also check out the accompanying video, where we show animations of the complex scenes and elaborate the detailed parameters of our layered BRDFs. Again, the cost of our method includes CPU time and GPU time, and the GPU is only used for inference of our neural networks.

Shoe. In Figure 10, we compare our method with Guo et al. [2018] on the Shoe scene under a point light and environment lighting. The surface of the Shoe is defined with a normal mapped BRDF consisting of two layers and a constant medium in between. With only 1 spp, our method is already close to noise-free (since the highlights mostly come from the point light in direct illumination), thanks to our noise-free layering and evaluation operations.

Globe and Teapot. In Figure 11, we show a Globe scene and a Teapot scene with two-layer BRDFs, and we define spatially-varying albedos of the interface between two layers. In both scenes, the top of the BRDF is a relatively smooth dielectric varnish layer and the bottom is a rough conductor. We compare our method against Guo et al. [2018] at equal time, finding that our result has much less noise. Additionally, we can arbitrarily specify spatially-varying scattering parameters (e.g., albedos) in the medium interface between layers (as well as spatially-varying roughness, shown in the accompanying video) to display various patterns. Each edit requires a re-evaluation

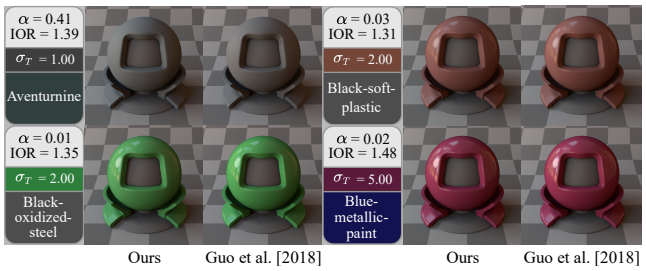

Figure 9: Comparison between the rendering results of our layering network with materials in MERL dataset [Matusik 2003] and the reference by Guo et al. [2018]. Our method can produce very close results to the reference.

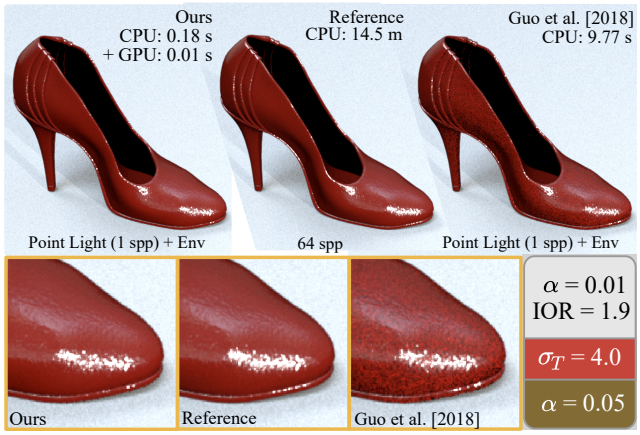

Figure 10: Comparison between our method and Guo et al. [2018] in the Shoe scene. Our method has low noise even at 1 spp (noise only comes from the environment lighting and indirect illumination).

Table 3: We tabulate the scene settings, time consumption (minutes) and error (MSE) in all scenes, compared with Guo et al. [2018] in equal-time.

Scene	Resolution	Ours		Guo et al.[2018]		Time
		Spp	MSE	Spp	MSE	
Still Life (Fig. 1)	1024 × 1024	512	$\mathbf{1.0 \times 10^{-3}}$	64	2.0×10^{-3}	4.82
Globe (Fig. 11)	1024 × 1024	256	$\mathbf{9.3 \times 10^{-4}}$	24	2.3×10^{-3}	2.42
Teapot (Fig. 11)	720 × 480	256	$\mathbf{6.0 \times 10^{-4}}$	12	5.4×10^{-3}	0.97

of our layering network, which is fast compared to the rendering itself.

Still Life. Figure 1 shows a variety of spatially-varying effects, including varying roughness, varying Fresnel, varying albedos of the interface, and normal mapping. Again, we can see that our method produces almost identical results to the reference on all these configurations within much less time.

5.3 Discussion and limitations

Our method is able to represent a large range of single- and multi-layer BRDFs with both specular and diffuse appearances, and the latent representation can be easily used to produce different effects. However, there are some approaches that we have not tried but may be potentially helpful. We also have identified some main limitations of our method and discuss the key points below.

Bidirectional Transmittance Distribution Functions (BTDFs). When training our layering network, we only use the reflection information from each BRDF layer to get the reflectance of the final layered material. When a BRDF is used as the top interface of a layering, its corresponding BTDF will be implicitly inferred by the layering network, and is never constructed explicitly. On the other hand, an extension to full neural BSDFs would be interesting and useful for other effects (e.g., translucent fabrics).

Scope/types of BRDFs. Generally, we find that BRDFs showing multiple separate lobes and highly-specular BRDFs are more difficult to predict. Currently, we do not train our model on anisotropic or normal mapped BRDFs (from individual layers). Including these types of BRDFs will further improve the practicality of our method, but is also more challenging, since the dimension of input data will further increase. While we believe our general framework could handle these effects, we leave the exploration of anisotropic and normal mapped BRDFs for future work.

Accumulated error. We have shown that when recursively applied, our layering network can be used to predict BRDFs consisting of multiple layers. However, error may accumulate as many BRDFs are layered together. This could be addressed by networks trained on a larger fixed number of BRDF layers, or exploiting neural network architectures that allow a dynamic number of inputs.

Bias and energy conservation. In practice, any neural network may produce error, which cannot be treated as unbiased in the Monte Carlo sense. The error could make the results consistently darker/brighter or introduce violation of energy conservation in rendering. This issue affects all neural rendering solutions; we have not observed artifacts caused by this, but the problem may require future research.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented a framework for neural layered BRDFs. We use a general neural network to compress BRDFs into short latent vectors, and we train a second network for the layering operation in latent space. Our representation network is able to accurately and compactly compress a wide range of BRDFs, including typical microfacet BRDFs, layered BRDFs or measured BRDFs. Our layering network avoids the expensive Monte Carlo random walk, resulting in a noise-free and computationally efficient layered material evaluation approach.

We believe that the proposed representation and layering model is novel and practical, and is a first step towards a "Neural BRDF algebra". However, this BRDF algebra is still not complete enough to encompass all common BRDFs and operations on them. Our model is currently trained on isotropic materials and media, but could be extended to anisotropic materials and media, as well as explicit handling of transmissive BTDFs. It would also be interesting to

Figure 11: Comparison between Guo et al. [2018] and our method on the Globe and Teapot scene with spatially-varying albedos of the medium between two layers. The difference images show that our method has much less error than Guo et al. [2018].

introduce normal mapping, color and roughness adjustment and other operations in the latent space.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments and suggestions. This work has been partially supported by the National Natural Science Foundation of China under grant No. 62172220, the Fundamental Research Funds for the Central Universities under grant No. 30920021133 and "111" Program under grant No. B13022. Ling-Qi Yan is supported by gift funds from Adobe, Dimension 5 and XVerse.

REFERENCES

Laurent Belcour. 2018. Efficient Rendering of Layered Materials Using an Atomic Decomposition with Statistical Operators. *ACM Trans. Graph.* 37, 4, Article 73 (July 2018), 15 pages.

Luis E. Gamboa, Adrien Gruson, and Derek Nowrouzezahrai. 2020. An Efficient Transport Estimator for Complex Layered Materials. *Computer Graphics Forum* 39, 2 (2020), 363–371. https://doi.org/10.1111/cgf.13936

Yu Guo, Miloš Hašan, and Shuang Zhao. 2018. Position-Free Monte Carlo Simulation for Arbitrary Layered BSDFs. *ACM Trans. Graph.* 37, 6, Article 279 (Dec. 2018), 14 pages.

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.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob. 2015. layerlab: A computational toolbox for layered materials. In *SIGGRAPH 2015 Courses (SIGGRAPH '15)*. ACM, New York, NY, USA. https://doi.org/10.1145/2776880.2787670

Wenzel Jakob, Eugene d'Eon, Otto Jakob, and Steve Marschner. 2014. A Comprehensive Framework for Rendering Layered Materials. *ACM Trans. Graph.* 33, 4, Article 118 (July 2014), 14 pages.

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2021. NeuMIP: Multi-Resolution Neural Materials. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4, Article 175 (July 2021), 13 pages.

Wojciech Matusik. 2003. *A data-driven reflectance model.* Ph.D. Dissertation. Massachusetts Institute of Technology.

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. 2019. Neural BTF Compression and Interpolation. *Computer Graphics Forum (Proceedings of Eurographics)* 38, 2 (March 2019).

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. 2020. Unified Neural Encoding of BTFs. *Computer Graphics Forum (Proceedings of Eurographics)* 39, 2 (June 2020). https://doi.org/10.1111/cgf.13921

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural BRDF Representation and Importance Sampling. *Computer Graphics Forum* n/a, n/a (2021). https://doi.org/10.1111/cgf.14335

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11358–11367.

Andrea Weidlich and Alexander Wilkie. 2007. Arbitrarily Layered Micro-Facet Surfaces. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia* (Perth, Australia) *(GRAPHITE '07)*. 171–178.

Philippe Weier and Laurent Belcour. 2020. Rendering Layered Materials with Anisotropic Interfaces. *Journal of Computer Graphics Techniques (JCGT)* 9, 2 (20 June 2020), 37–57. http://jcgt.org/published/0009/02/03/

Mengqi (Mandy) Xia, Bruce Walter, Christophe Hery, and Steve Marschner. 2020. Gaussian Product Sampling for Rendering Layered Materials. *Computer Graphics Forum* 39, 1 (2020), 420–435.

Tomoya Yamaguchi, Tatsuya Yatagawa, Yusuke Tokuyoshi, and Shigeo Morishima. 2019. Real-Time Rendering of Layered Materials with Anisotropic Normal Distributions. In *SIGGRAPH Asia 2019 Technical Briefs (SA '19)*. Association for Computing Machinery, New York, NY, USA, 87–90. https://doi.org/10.1145/3355088.3365165

Tizian Zeltner and Wenzel Jakob. 2018. The Layer Laboratory: A Calculus for Additive and Subtractive Composition of Anisotropic Surface Reflectance. *Transactions on Graphics (Proceedings of SIGGRAPH)* 37, 4 (July 2018), 74:1–74:14.

Chuankun Zheng, Ruzhang Zheng, Rui Wang, Shuang Zhao, and Hujun Bao. 2021. A Compact Representation of Measured BRDFs Using Neural Processes. *ACM Transactions on Graphics (TOG)* 41, 2 (2021), 1–15.