Synthesizing Time-varying BRDFs via Latent Space

Takuto Narumoto, Hiroaki Santo, and Fumio Okura

Graduate School of Information Science and Technology, Osaka University, Japan
{narumoto.takuto, santo.hiroaki, okura}@ist.osaka-u.ac.jp

Abstract. This paper introduces a method for synthesizing time-varying bidirectional reflectance distribution functions (BRDFs) by applying learned temporal changes to static BRDFs. Achieving realistic and natural changes in material appearance over time is crucial in computer graphics and virtual reality. Existing methods employ a parametric BRDF model, and the temporal changes in BRDFs are modeled by polynomial functions that represent the transitions of the BRDF parameters. However, the limited representational capabilities of both the parametric BRDF model and the polynomial temporal model restrict the fidelity of the appearance reproduction. In this paper, to overcome this limitation, we introduce a neural embedding for BRDFs and propose a neural temporal model that represents the temporal changes of BRDFs in the latent space, which allows flexible representations of BRDFs and temporal changes. The experiments using synthetic and real-world datasets demonstrate that the flexibility of the proposed approach achieves a faithful synthesis of temporal changes in material appearance.

Keywords: texture synthesis · time-varying appearance

1 Introduction

Achieving realistic and natural temporal changes in material appearance is essential in computer graphics and virtual reality. Reflectances on a scene surface can be represented by the bidirectional reflectance distribution functions (BRDFs), which are functions of incoming and outgoing light directions and return the reflectance rates. Parametric BRDF models [5, 6, 22] have been proposed to represent reflection by a set of parameters to control the appearance. However, it is hard to manually design the transitions of the BRDF parameters to represent natural changes of the temporal appearances.

For the synthesis of time-varying BRDFs, previous works transfer the temporal transitions of the BRDF parameters to static BRDFs, utilizing a small-scale real-world time-varying BRDF dataset [10]. These methods approximate the transitions of the BRDF parameters by polynomial functions to cope with the data limitation and transfer their coefficients to a static BRDF. However, the fidelity of the synthesized BRDFs is limited due to a lack of flexibility in (1) the parametric representation of BRDFs and (2) the polynomial representation of their temporal model.

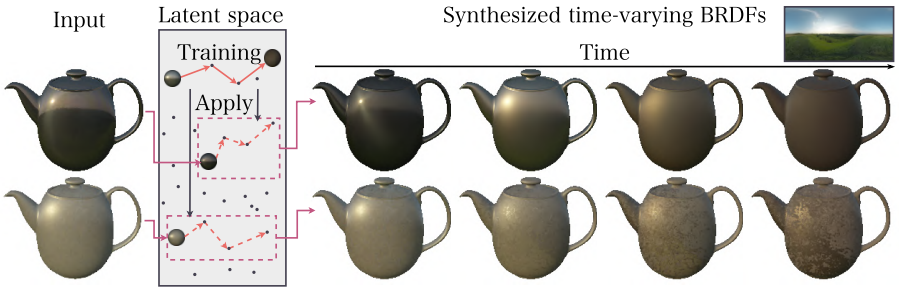


Fig. 1: Overview of the proposed method. We estimate the transitions of BRDFs over time in the latent space, enabling the faithful synthesis of time-varying BRDFs. The right-top figure illustrates the environment map used for rendering the results.

To address these issues, this paper presents a method that synthesizes time-varying BRDFs using a neural temporal model that effectively captures the temporal transition from limited resources on the time-varying BRDFs. The key to our method is to disentangle the static and temporal representations as shown in Fig. 1. Specifically, the proposed method employs a neural embedding to represent BRDFs as latent vectors, which is learned from large-scale static BRDF databases. Given the compact embedding of BRDFs, we train a lightweight neural temporal model (NTM) using the time-varying BRDF datasets, efficiently representing the transitions of the latent vectors over time. Our neural representations of BRDFs and the neural temporal model offer greater flexibility than existing parametric models, allowing the synthesis of time-varying BRDFs faithfully reflecting both input BRDFs and temporal changes.

The quantitative evaluation using synthetic data and qualitative evaluation using a real-world dataset demonstrate that the proposed method achieves a high-fidelity synthesis of BRDFs compared to the polynomial model-based method. Our implementation will be published upon acceptance.

2 Related work

In this section, we first review previous works dealing with time-varying BRDFs. We then introduce neural network-based representation methods for BRDFs.

Time-varying BRDFs Gu *et al.* [10] introduce a real-world dataset of time-varying BRDFs, which captures 26 samples with 5 different changes over time, namely burning, drying of smooth surfaces, drying of rough surfaces, rusting, and ripening. The dataset contains time-series textured BRDFs, *i.e.*, bidirectional texture functions (BTFs), where the BRDFs at each pixel are represented by the Torrance-Sparrow model [22]. Capturing these real-world datasets requires large-scale equipment (*e.g.*, a light stage) and substantial labor, making large-scale data collection difficult.

To synthesize time-varying BRDFs, [10] also proposes a model for representing the transition of the BRDF parameters, space-time appearance factorization (STAF). The STAF model represents each BRDF parameter; diffuse albedo, specular albedo, and roughness are expressed by polynomial functions with respect to timestamp t and affine projections. While the STAF model can be used for synthesizing new time-varying BRDFs by replacing the parameters of the affine projections from a different static BRDF, it suffers from spatial inconsistency due to the mismatch between the original and given parameters.

To deal with this issue, Meister *et al.* [17] have proposed the spatio-temporal BRDF (STB) model, which represents each BRDF parameter with spatially-varying polynomial functions with respect to timestep t. Based on the STB model, the example-based approach for synthesizing time-varying BTFs, Spatio-Temporal BRDFs transfer (STT), has been proposed. STT takes a time-varying BTF, fitted to the STB model, and a texture image as input and synthesizes a novel time-varying BTF, *i.e.*, determines coefficients of polynomial functions for diffuse albedo, specular albedo, and roughness. They first find the correspondences of patches between the input texture image and the diffuse albedo map of the input time-varying BTFs with timestep $t = 0$. For each corresponding patch, the synthesized STB obtains the coefficients of the specular albedo and roughness from the STB fitted to the input time-varying BTFs and the coefficients of the diffuse albedo from the input texture image with scale modification. Unlike the STAF model, the STT model incorporates spatial information. However, since the specular albedo and roughness are directly derived from the input time-varying BRDFs, transferring the temporal changes to materials with large differences regarding specularity becomes challenging.

In contrast to these data-driven methods, Kimmel *et al.* [13] propose a modeling-based simulation method of aging and weathering induced by absorbed radiation, which is derived from physical theory. The proposed method bypasses the difficulty of modeling each class of time-varying appearances by learning them from data.

Neural representation for BRDF While the measured BRDFs [7,16], which store reflectances for possible sets of lighting and viewing directions, have greater flexibility than conventional modeling-based BRDFs [4–6,18,22,26], they require a large number of parameters, making it hard to control the appearance by altering these parameters. Recent studies have explored neural representations that embed high-dimensional data into a compact latent space, which reduces the required memory footprint and allows material editing and interpolating on the latent space.

Hu *et al.* [12] have proposed the use of autoencoders to obtain a compact representation of measured BRDFs. Unlike previous factorization-based representations [3, 16], their method achieves lower reproduction error with a smaller number of parameters. Sztrajman *et al.* [21] have proposed the Neural BRDF (NBRDF) that also uses the autoencoder architecture to obtain a BRDF from samples of measured reflectances and observed angles. NBRDF trains per-material autoencoders and trains another autoencoder that models the weight

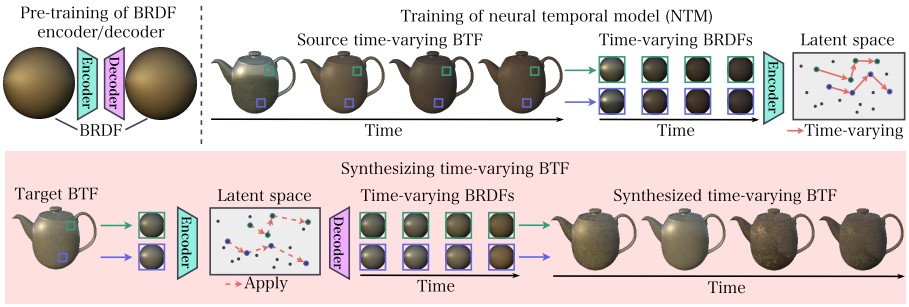


Fig. 2: Overview of the proposed method. The proposed method consists of three modules: the BRDF encoder and decoder and the neural temporal model (NTM). The encoder and decoder are pretrained on large-scale (static) BRDF datasets. We input time-varying BTFs as source material and embed per-pixel BRDFs into the latent space. The NTM then learns the transitions of latent vectors over time. To synthesize novel time-varying BTFs, we embed BRDFs in the target BTFs into the latent space. The trained NTM subsequently estimates a time series of latent vectors conditioned by the target BRDFs as an initial timestep, which are fed to the decoder to reconstruct the BRDFs.

parameters of the autoencoder, resulting in better reconstructions. However, since NBRDF needs to train the autoencoders per material, the embedding is computationally expensive. Zheng *et al.* [27] adopts the neural processes [9] in an encoder-decoder architecture to obtain a compact representation of BRDFs, denoted as BRDFNPs. BRDFNPs takes as input the sets of reflectances and incoming and outgoing light directions and embeds them into a latent vector. Fan *et al.* [8] proposes the use of the latent space of BRDFs to synthesize the layered BRDFs, denoted as neural layered BRDFs. Conventional renderer requires expensive computation to simulate the layered BRDFs; this method allows the synthesis of the layered BRDFs by combining the latent vectors of the top and bottom layer's BRDFs. The proposed method is inspired by the neural layered BRDFs and synthesizes the time-varying BRDFs in the latent space. For the embedding of BRDFs, we also propose an extended method of BRDFNPs.

3 Proposed method

Figure 2 summarizes the overview of the proposed method. We begin by training the BRDF encoder and decoder to embed a BRDF into a latent space, using large-scale data sources of static BRDFs. To model the temporal changes, we introduce the neural temporal model (NTM) representing the temporal changes in the latent space. We explain the details of each step as follows.

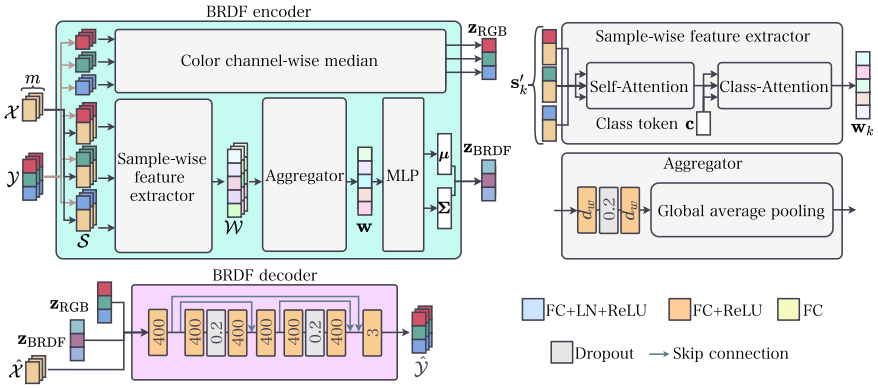


Fig. 3: Network architecture of the proposed BRDF encoder and decoder. The encoder takes the sets of angles $\mathcal{X}$ reflectances $\mathcal{Y}$ as input and outputs the latent vectors $\mathbf{z}_{\mathrm{RGB}}$ and $\mathbf{z}_{\mathrm{BRDF}}$. We input the latent vectors $\mathbf{z}_{\mathrm{RGB}}$ and $\mathbf{z}_{\mathrm{BRDF}}$ and sets of angles $\hat{\mathcal{X}}$ into the decoder and obtain the reflectances $\hat{\mathcal{Y}}$ corresponding to the input angles $\hat{\mathcal{X}}$. FC, LN, and MLP represent a fully connected layer, layer normalization [1], and multilayer perceptron, respectively.

3.1 BRDF encoder and decoder

Figure 3 shows the network architectures of the BRDF encoder and decoder. Similar to BRDFNPs [27], the BRDF encoder takes as input m sets of angles $\mathcal{X} := \{[\boldsymbol{\theta}_k \in \mathbb{R}^4]^\top\}_{k=1}^m$, where $\boldsymbol{\theta}_k$ is the incoming and outgoing light directions for the k-th sample represented by the half-vector parameterization [19,21] and the corresponding RGB reflectances $\mathcal{Y} := \left\{ \left[v_r^{(k)}, v_g^{(k)}, v_b^{(k)} \right]^\top \right\}_{k=1}^m$.

To better deal with the color inputs, we propose a branched network architecture that consists of two modules: one for extracting a color latent vector $\mathbf{z}_{\mathrm{RGB}} \in \mathbb{R}^3$ and another for a latent vector invariant to the permutation of color channels, denoted as permutation-invariant latent vector $\mathbf{z}_{\mathrm{BRDF}} \in \mathbb{R}^{d_z}$. While the color latent vector retains information on the material color, the permutation-invariant latent vector allows for efficient training without the need for data augmentation related to the permutation of color channels employed in [21].

The color latent vector $\mathbf{z}_{\mathrm{RGB}}$ is extracted by the parameter-free operation, *i.e.*, the color channel-wise median for all input reflectances. The permutation-invariant latent vectors $\mathbf{z}_{\mathrm{BRDF}}$ is extracted using attention-based neural networks. As following the BRDFNPs, the permutation-invariant latent vectors $\mathbf{z}_{\mathrm{BRDF}}$ is modeled as Gaussians, and the encoder outputs the mean $\boldsymbol{\mu} \in \mathbb{R}^{d_z}$ and the variance $\boldsymbol{\Sigma} \in \mathbb{R}^{d_z}$ of them. We first concatenate each sample from the input angles $\mathcal{X}$ and reflectances $\mathcal{Y}$ for a each color channels, resulting input samples $\mathcal{S} = \{\mathbf{s}'_k\}_{k=1}^m$ where $\mathbf{s}'_k = \left\{ \left[v_r^{(k)}, \boldsymbol{\theta}_k^\top \right]^\top, \left[v_g^{(k)}, \boldsymbol{\theta}_k^\top \right]^\top, \left[v_b^{(k)}, \boldsymbol{\theta}_k^\top \right]^\top \right\}$. We treat

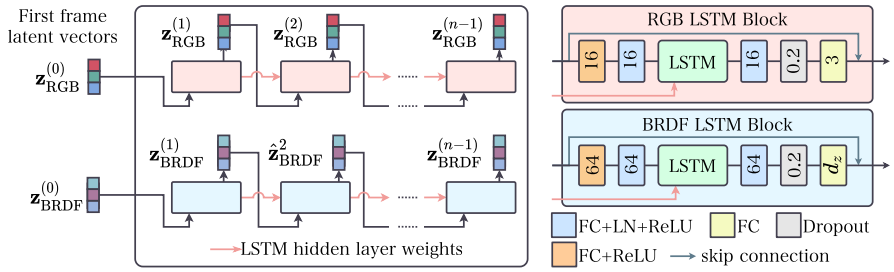


Fig. 4: The network architecture of NTM, which estimates the latent vector of the next timestep using LSTM [11]. NTM incorporates two LSTM blocks: one for the color latent vectors $\left\{\mathbf{z}^{(t)}_{\mathrm{RGB}}\right\}$ and another for the permutation-invariant latent vectors $\left\{\mathbf{z}^{(t)}_{\mathrm{BRDF}}\right\}$.

each sample $\mathbf{s}'_k$ as a permutation-invariant set of the 5-dimensional vectors and feed them into the feature extractor. This extractor consists of self-attention [24] and class-attention [23], which uses the class token $\mathbf{c}$ for the aggregation, designed to extract features that are invariant to the permutation of the color channels. Consequently, we obtain the sample-wise features $\mathcal{W} = \left\{\mathbf{w}_k \in \mathbb{R}^{d_w}\right\}_{k=1}^{m}$. We then feed the sample-wise features $\mathcal{W}$ into the aggregator, which uses the global average pooling [15] to compute the material-wise feature $\mathbf{w} \in \mathbb{R}^{d_w}$ being invariant to the number of samples m. Finally, we use the multilayer perceptron (MLP) to estimate the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\Sigma}$ of the permutation-invariant latent vectors $\mathbf{z}_{\mathrm{BRDF}}$. To handle the high-dynamic range of reflectances $\mathcal{Y}$, we apply the log transformation [27]:

$$l(\mathbf{a}) = \log\left(\mathbf{1} + \log\left(\mathbf{1} + \log\left(\mathbf{1} + \log\left(\mathbf{1} + \mathbf{a}\right)\right)\right)\right). \tag{1}$$

The decoder uses MLPs with skip connections. The decoder takes as input the concatenated latent vector $\left[\mathbf{z}^{\top}_{\mathrm{RGB}}, \mathbf{z}^{\top}_{\mathrm{BRDF}}\right]^{\top} \in \mathbb{R}^{3+d_z}$, where $\mathbf{z}_{\mathrm{BRDF}}$ is randomly sampled from the estimated normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and sets of angles $\hat{\mathcal{X}}$. It then outputs the reflectances $\hat{\mathcal{Y}}$ that corresponds to the input angles $\hat{\mathcal{X}}$.

The training of the encoder and decoder follows the same procedure used in BRDFNPs. For the training dataset, we use the MERL BRDF database [16], containing 100 measured BRDFs, and synthesized BRDFs using the Disney principled BRDF model [5], which is controlled by 11 parameters. During the training, half of the training batch is from MERL BRDFs, and the remaining is from synthesized BRDFs.

3.2 Neural Temporal Model (NTM)

The NTM aims to learn the temporal changes of BRDFs through the projection of latent vectors employing Long Short-Term Memory (LSTM) [11], which is a technique commonly used in sequential data processing. Figure 4 shows the network architecture of the proposed NTM. The NTM uses two distinct

LSTM blocks: one for the color latent vectors and another for permutation-invariant latent vectors. Given n frames time-varying BRDFs, we first compute the embedded latent vectors $\left\{\mathbf{z}_{\mathrm{RGB}}^{(t)}\right\}_{t=0}^{n-1}$ and $\left\{\mathbf{z}_{\mathrm{BRDF}}^{(t)}\right\}_{t=0}^{n-1}$ using the pre-trained BRDF encoder. The NTM takes the latent vectors $\mathbf{z}_{\mathrm{RGB}}^{(0)}$ and $\mathbf{z}_{\mathrm{BRDF}}^{(0)}$ as input and sequentially outputs the estimates of the latent vectors $\left\{\hat{\mathbf{z}}_{\mathrm{RGB}}^{(t)}\right\}_{t=1}^{n-1}$ and $\left\{\hat{\mathbf{z}}_{\mathrm{BRDF}}^{(t)}\right\}_{t=1}^{n-1}$. We then train the NTM using the following loss

$$\mathcal{L} = \frac{1}{n-1}\sum_{t=1}^{n-1}\left(\left\|l^{-1}\left(\mathbf{z}_{\mathrm{RGB}}^{(t)}\right) - l^{-1}\left(\hat{\mathbf{z}}_{\mathrm{RGB}}^{(t)}\right)\right\|_2^2 + \left\|\mathbf{z}_{\mathrm{BRDF}}^{(t)} - \hat{\mathbf{z}}_{\mathrm{BRDF}}^{(t)}\right\|_2^2\right),$$

where $l^{-1}(\cdot)$ denotes the inverse of the log transformation (Eq. (1)).

Once trained on the source time-varying BRDFs, the NTM takes the latent vectors of the target static BRDFs as input and estimates the time series of latent vectors. These latent vectors are then fed into the BRDF decoder to reconstruct the time-varying BRDFs. The synthesis is conducted in a per-pixel manner; a BTF is constructed by arranging the individual estimates.

4 Experiments

We first assess the representation capabilities of the proposed BRDF encoder and decoder. We then show the quantitative and qualitative evaluations of the synthesis of time-varying BTFs by the proposed method using synthetic and real-world datasets. In the following sections, we detail the experimental settings and present the evaluation results.

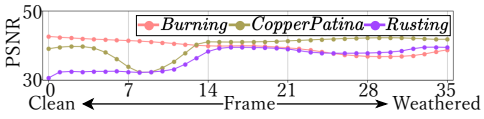


Fig. 6: Reconstruction accuracy of the proposed BRDF encoder and decoder for three time-varying BTFs in the STAF database [10].

4.1 Implementation details

The proposed method is implemented using PyTorch[1]. We train the BRDF encoder and decoder for $600,000$ iterations using the Adam optimizer [14] with the learning rate set to 1×10^{-4} and the batch size to 16. Following the original BRDFNPs, the number of samples m is set to $16,200$. The hyperparameters of the networks d_w and d_z are set to 64 and 7. The training process for the BRDF encoder and decoder requires approximately 60 hours on a single NVIDIA QUADRO RTX 8000.

For the training of the NTM, we input a time-varying BTF as input and independently feed the per-pixel time-varying BRDFs into the network. We train the NTM for $900,000$ iterations using the Adam optimizer, with the learning rate set to 5×10^{-4} and the batch size to 1024, which takes approximately 3.5 hours.

[1] PyTorch v2.0.0, `https://pytorch.org/`, last accessed on July 15, 2024.

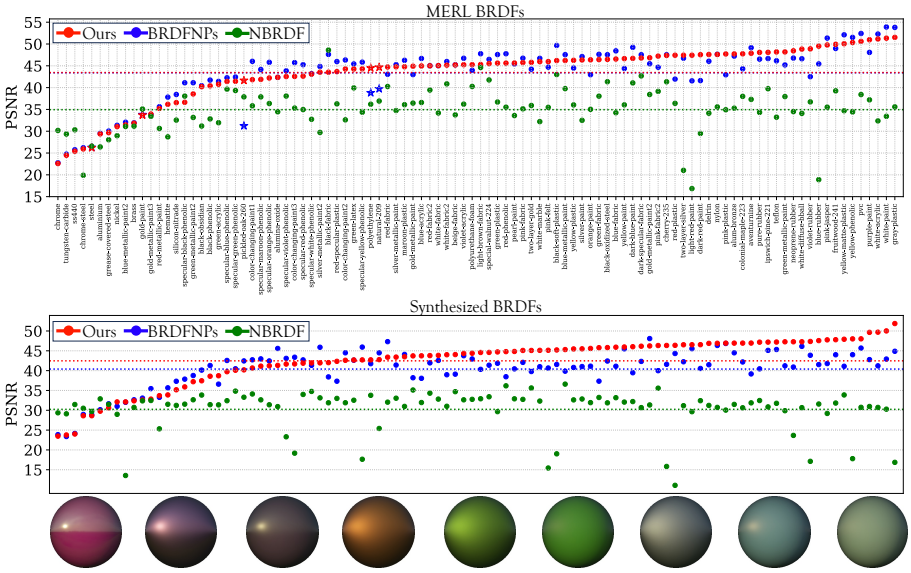


Fig. 5: Evaluation of reconstruction errors for BRDFNPs [27], NBRDF [21], and the proposed BRDF encoder and decoder. We use the MERL BRDFs [16] and our synthesized BRDFs and show the PSNRs of sphere images rendered with the estimated BRDFs. For the MERL BRDFs, five BRDFs, marked with a star indicator, are excluded from the training dataset for both our method and BRDFNPs. For the synthetic BRDFs, the ground truth images of spheres are shown below the plot. The dotted lines indicate the mean of all BRDFs for each method.

4.2 Representation capabilities of BRDF encoder and decoder

We compare our BRDF encoder and decoder with the state-of-the-art neural BRDF representation methods: BRDFNPs and NBRDF. BRDFNPs are retrained using the same dataset as ours.

Dataset and evaluation metric We use the MERL BRDFs and randomly synthesized BRDFs using the Disney principled BRDF for the evaluation. Regarding the MERL BRDFs, 5 BRDFs out of 100 are not used in the training for ours and BRDFNPs. We also use the STAF database [10] to evaluate the performance of the proposed BRDF encoder and decoder for time-varying BRDFs.

For the quantitative evaluation, we render a sphere under the environment map[2], shown in Fig. 1, using the reconstructed BRDFs and compute the peak signal-to-noise ratio (PSNR). When computing PSNR, we apply the gamma correction [2] as $\Gamma(v) = \left(1 - \mathrm{e}^{2(-\beta v)}\right)^{\frac{1}{\gamma}}$ where $\beta = 1.0$ and $\gamma = 2.8$.

Results for static BRDFs Figure 5 shows the PSNR for 100 MERL BRDFs and 100 randomly synthesized BRDFs. Ours and BRDFNPs achieve better PSNR

[2] Poly Haven, `https://polyhaven.com/`, last accessed on July 15, 2024.

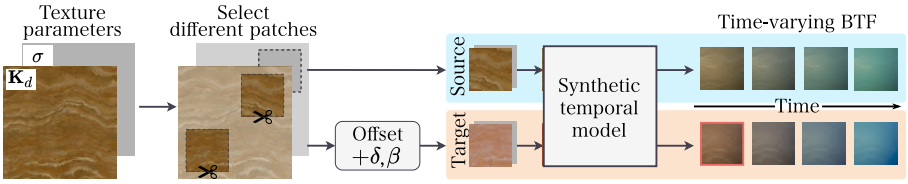


Fig. 7: Generation procedure of our synthetic time-varying BTF dataset.

than NBRDF. When comparing our method to BRDFNPs, both achieve similar accuracy on average for MERL BRDFs, but our method excels with synthesized BRDFs. As shown in the figure, our method performs similarly to BRDFNPs for materials with lower roughness but better for materials with relatively higher roughness. This is because, in the dichromatic reflection model, the specular color is modeled by the color of the incoming light, *i.e.*, white. Thus, our permutation-invariant architecture is more effective for rough materials than for specular ones.

Results for time-varying BRDFs Figure 6 shows the plots of PSNRs of the proposed BRDF encoder and decoder for three materials in the STAF database over different frames. We can see the stable predictions of the proposed method regardless of whether the materials are clean or weathered.

4.3 Novel BTFs synthesis

We compare the synthesis performance of the example-based method, STT [17], and the proposed method using two datasets: our synthetic dataset and the STAF database [10], a real-world dataset of time-varying BTFs. In the following sections, we present a detailed description of the datasets and evaluation results.

Synthetic dataset For the quantitative evaluation, we generate time-varying BRDFs by changing the BRDF parameters over time. For the BRDF model, we employ the Torrance-Sparrow model [22], the same model used in the comparison method, STT. Specifically, the Torrance-Sparrow model $b(x, y, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ is described as:

$$b(x, y, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{\mathbf{K}_d(x, y)}{\pi} + \mathbf{K}_s(x, y)\rho_s(x, y, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}),$$
$$\rho_s(x, y, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h})G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \sigma(x, y))D(\boldsymbol{\omega_h}, \sigma(x, y))}{4(\boldsymbol{\omega_i}^\top \boldsymbol{n})(\boldsymbol{\omega_o}^\top \boldsymbol{n})},$$

where (x, y), $(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$, $\boldsymbol{n}$, F, G, and D are the texture coordinates, the incoming and outgoing light directions, the normal direction, the Fresnel term, the Smith shadowing-masking function [20], and the GGX normal distribution function [25], respectively. $\mathbf{K}_d = \left[K_d^{(r)}, K_d^{(g)}, K_d^{(b)}\right]^\top \in \mathbb{R}^3$, $\mathbf{K}_s \in \left[K_s^{(r)}, K_s^{(g)}, K_s^{(b)}\right]^\top \in \mathbb{R}^3$, and $\sigma \in \mathbb{R}_+$ are diffuse albedo, specular albedo,

Table 1: Mean PSNRs over time for the proposed and comparison methods with varying the temporal parameter α and the offset parameter β. Avg. shows the average across the different settings.

		$\alpha = 0.6$					$\alpha = 0.8$					$\alpha = 1.0$					Avg.
		β=0	0.05	0.1	0.15	0.2	β=0	0.05	0.1	0.15	0.2	β=0	0.05	0.1	0.15	0.2	
S1	Ours	**39.9**	**40.2**	**41.7**	39.3	**39.4**	**40.2**	**40.5**	**40.5**	**40.6**	**40.5**	**39.2**	**39.4**	**39.3**	**41.1**	**39.3**	**40.1**
	STT [17]	39.2	40.2	40.2	**39.8**	39.1	38.1	38.9	38.9	38.5	38.0	37.1	38.0	37.7	37.5	37.1	38.5
S2	Ours	42.4	43.3	**41.6**	**39.0**	**38.0**	42.7	42.8	**40.4**	**39.0**	**37.9**	42.5	**42.8**	**41.1**	**39.4**	**38.0**	**40.7**
	STT [17]	**47.1**	**44.9**	40.3	37.4	35.6	**45.6**	**43.7**	39.7	37.1	35.4	**44.2**	42.6	38.9	36.7	35.2	40.3

and roughness, respectively, which parameterize the appearance. Given that dynamic changes in the specular albedo color result in an unnatural appearance, we use $K_s^{(r)} = K_s^{(g)} = K_s^{(b)} = K_s$, following the STAF and STB models. To mimic the temporal changes of the appearance, we define the following two temporal models:

1. Synthetic temporal model 1 (S1):

$$\begin{cases} K_d^{(r)}(x,y,t) = \frac{1}{(1+\alpha t)^2} K_d^{(r)}(x,y,0) \\ K_d^{(g)}(x,y,t) = \alpha t^2 + \frac{1}{1+\alpha t^2} K_d^{(g)}(x,y,0) \\ K_d^{(b)}(x,y,t) = \frac{1.5}{1.5-\alpha t^2} K_d^{(b)}(x,y,0) + \alpha t \\ K_s(x,y,t) \quad = K_s(x,y,0) \\ \sigma(x,y,t) \quad = \sigma(x,y,0) \end{cases}$$

2. Synthetic temporal model 2 (S2):

$$\begin{cases} K_d^{(r)}(x,y,t) = \alpha(0.9 - 0.9K_d^{(r)}(x,y,0))t + K_d^{(r)}(x,y,0) \\ K_d^{(g)}(x,y,t) = \alpha(0.4 - 0.9K_d^{(g)}(x,y,0))t + K_d^{(g)}(x,y,0) \\ K_d^{(b)}(x,y,t) = \alpha(0.15 - 0.9K_d^{(b)}(x,y,0))t + K_d^{(b)}(x,y,0) \\ K_s(x,y,t) \quad = -0.3\alpha t + K_s(x,y,0) \\ \sigma(x,y,t) \quad = 0.25\alpha\sigma(x,y,0)t^2 + 0.5(1+\alpha)\sigma(x,y,0)t + \sigma(x,y,0) \end{cases}$$

These models are controlled by the initial texture parameters and the temporal parameter α.

Using the synthetic temporal models, we generate pairs of source and target materials as illustrated in Fig. 7. The initial texture parameters are obtained from off-the-shelf static materials from ambientCG[3]. We use the same material for both the source and target, selecting different patches for each. In addition, to mimic the gap between source and target materials, we introduce the offset parameters $\boldsymbol{\delta} \in \mathbb{R}^3$ and $\beta \in \mathbb{R}$ where $\mathbf{K}_d^{(\text{target})} = \boldsymbol{\delta} + \mathbf{K}_d^{(\text{source})}$ and $\sigma^{(\text{target})} = \beta + \sigma^{(\text{source})}$.

Real-world dataset From the STAF databse [10], we use the (a) *Burning*, (b) *Copper Patina*, and (c) *Rusting* BTFs as source data. The capture time range is normalized to 0 to 1, and each data has 36 frames with 300×300 texture resolution.

[3] ambientCG, `https://ambientcg.com/`, last accessed on July 15, 2024.

Table 2: PSNRs of the final frame for the proposed and comparison methods with varying the temporal parameter α and the offset parameter β. Avg. shows the average across the different settings.

		$\alpha = 0.6$					$\alpha = 0.8$					$\alpha = 1.0$					Avg.
		β=0	0.05	0.1	0.15	0.2	β=0	0.05	0.1	0.15	0.2	β=0	0.05	0.1	0.15	0.2	
S1	Ours	**39.9**	**41.1**	**42.1**	**41.1**	**41.3**	**39.7**	**40.3**	**37.6**	**36.1**	**40.3**	**36.5**	**36.8**	**38.0**	**39.6**	**38.0**	**39.2**
	STT [17]	32.1	32.3	32.4	32.4	32.5	30.6	30.8	30.8	30.9	30.9	29.4	29.5	29.6	29.6	29.6	30.9
S2	Ours	41.5	**41.2**	**38.5**	35.5	**35.8**	**42.3**	**40.5**	**37.4**	**37.5**	**36.7**	**41.9**	**41.5**	**40.3**	**38.9**	**38.4**	**39.2**
	STT [17]	**42.9**	39.8	37.3	**35.5**	34.3	39.4	37.4	35.8	34.6	33.7	36.2	35.1	34.1	33.4	32.9	36.2

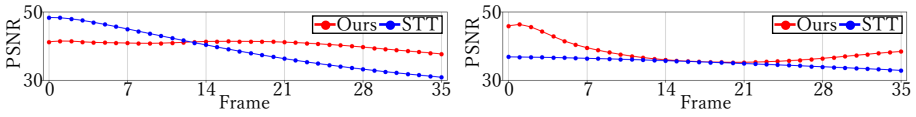
For the target material, we use three different data: (a) a *Wood* from the STAF database, (b) a *Copper* material from ambientCG, and (c) a *Steel* from the MERL BRDFs. Given that the MERL BRDFs lack spatial variation, we treat them as spatially uniform BTFs.

Experimental result for our synthetic dataset We vary the temporal parameter α and the offset of roughness β as $\alpha \in \{0.6, 0.8, 1.0\}$ and $\beta \in \{0, 0.05, 0.1, 0.15, 0.2\}$. We use $\delta = [0.1, -0.1, 0]^\top$ in synthetic temporal model 1 and $\delta = \mathbf{0}$ in synthetic temporal model 2.

Tables 1 and 2 present the mean PSNRs over time and the PSNRs of the final frame, respectively, for the proposed method and the comparison method, STT. For two scenes, we show the plots of PSNRs over time in Fig. 8 and visualizations in Figs. 9a and 9b. The PSNRs are computed in the same manner as described in Sec. 4.2. For the visualization, we render a pot scene with the estimated BTFs under the environment map shown in Fig. 1. Complete visualizations are available in the supplementary material.

Overall, the proposed method achieves better synthesis accuracy than STT. From the plots of PSNRs, we can see that STT experiences an increase in errors as time progresses. When there is a gap, particularly in specularity, *i.e.*, roughness, between the source and target materials, STT struggles to accurately reflect the target material, resulting in significant errors as shown in Fig. 9b, while STT can achieve accurate results for the scene with the small gap. In contrast, the proposed method precisely reproduces the target material. Based on the PSNRs of the final frames, the proposed method outperforms STT in almost all scenes.

Experimental result for the real-world dataset Figure 10 shows the visual results of synthesized BTFs by the proposed method and the comparison method, STT, for the three test scenes: (a), (b), and (c). In scene (a), both the proposed method and STT achieve comparable results, given that the source and target scenes share similarities in both reflectance and texture. However, in scene (b), STT fails to preserve the texture of the target BTF due to the failure of the patch matching in the source texture. In contrast, the proposed method maintains texture through its per-pixel approach. In scene (c), STT fails to accurately reproduce the specularity of the target material, as it relies solely on the specular albedo and roughness derived from the source material. In contrast, our proposed method faithfully reflects the specularity inherent in the target BTFs.



(a) S1 ($\alpha = 0.8$, $\beta = 0.1$) **(b)** S2 ($\alpha = 1.0$, $\beta = 0.2$)

Fig. 8: Plots of PSNRs over time for two scenes in our synthetic dataset.

4.4 Ablation study of permutation-invariant architecture

To confirm the effectiveness of the permutation-invariant architecture in the BRDF encoder and decoder for BTF synthesis performance, we conduct ablation studies comparing results with and without the proposed architecture, *i.e.*, ours and BRDFNPs [27]. Figure 11 shows the synthesis accuracies for the same two scenes in Fig. 8, demonstrating our consistently superior performance compared to BRDFNPs.

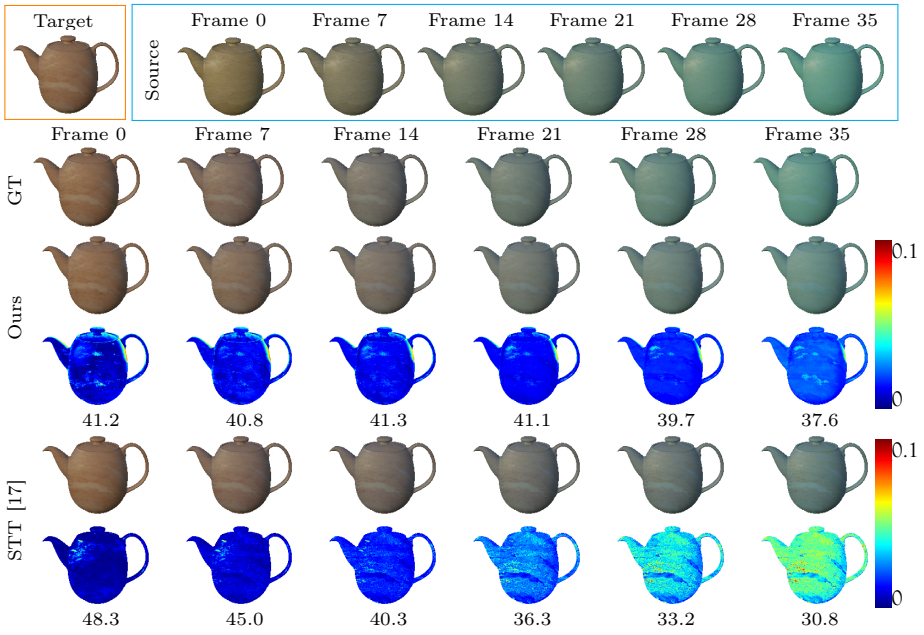
5 Discussion

This paper presents a method to synthesize time-varying BRDFs from source time-varying BRDFs and a target static BRDF. In contrast to previous methods, which suffer limited representation capabilities of parametric BRDF models and the polynomial temporal models, our proposed method achieves accurate synthesis through the use of neural embedding of BRDFs and a lightweight neural temporal model (NTM). For neural embedding, we introduce color latent and permutation-invariant latent representations, which offer efficient representation for a wide range of BRDFs. The proposed NTM represents temporal changes in BRDFs as the projections of latent vectors, which have great flexibility. Our experiments demonstrate that the proposed method excels in scenes with a large gap in reflectance properties between the source and target BRDFs. As a result, the proposed method enables us to easily generate time-varying appearances without manually designing the transitions of the BRDF parameters.

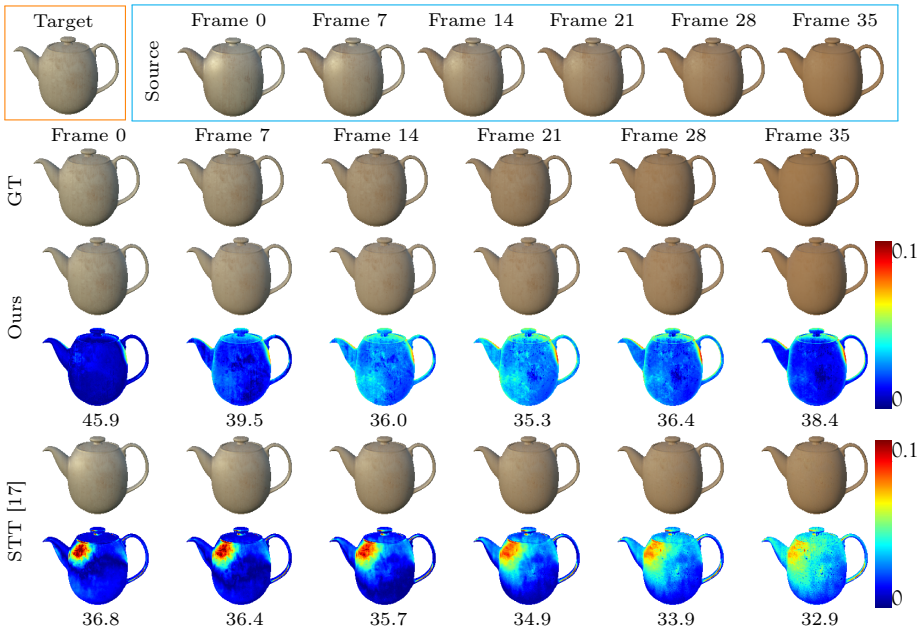
Limitation One of our limitations is that, since the proposed method trains the NTM for each specific temporal change in materials, we need to train and select the appropriate NTM according to the input target. Exploring adaptive synthesis through material classification and user instruction represents a promising direction for our future work.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers JP22K17910, JP23H05491, and JP21H03466, and JST FOREST Grant Number JPMJFR206F.

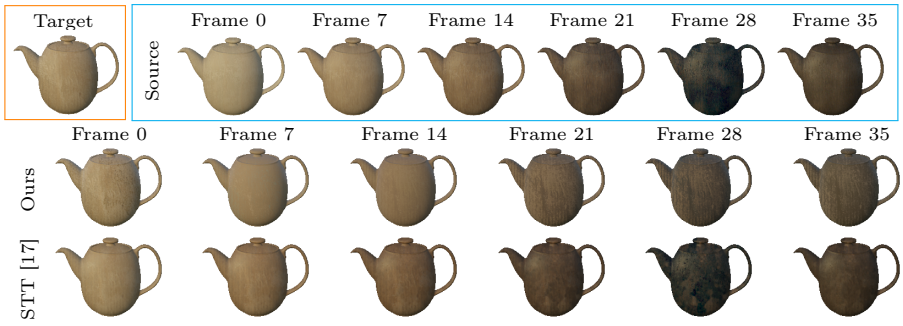


(a) synthetic temporal model 1 (S1) with $\alpha = 0.8$ and $\beta = 0.1$.

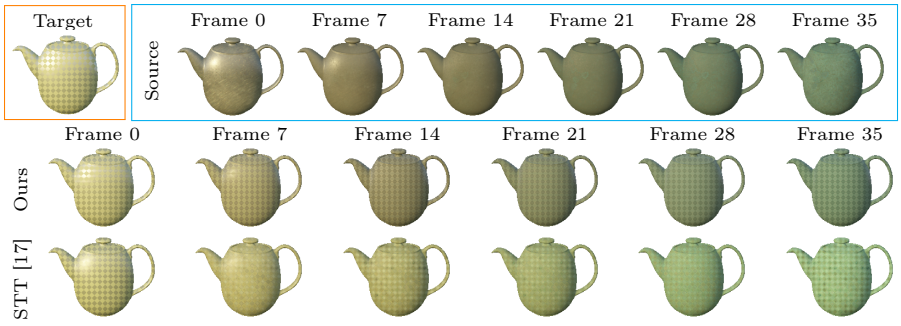


(b) synthetic temporal model 2 (S2) with $\alpha = 1.0$ and $\beta = 0.2$.

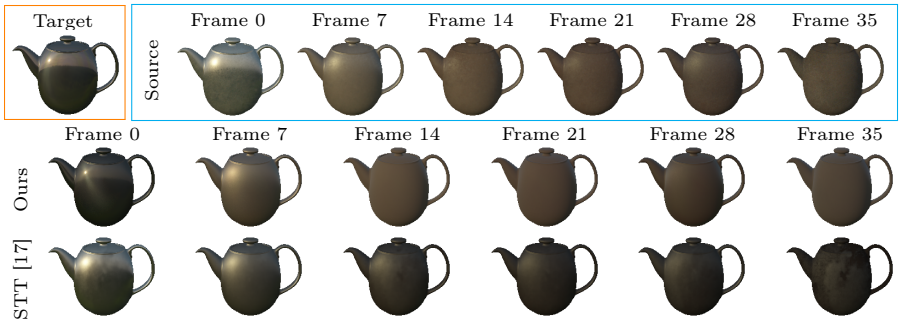
Fig. 9: Visual results for two scenes in our synthetic dataset. The top row illustrates the target and source materials, followed by rows that present the ground truth (GT) and estimated results by the proposed and comparison methods alongside their error maps. The numbers under the error maps indicate the PSNRs.



(a) Source: *Burning*, Target: *Wood* from the STAF database.

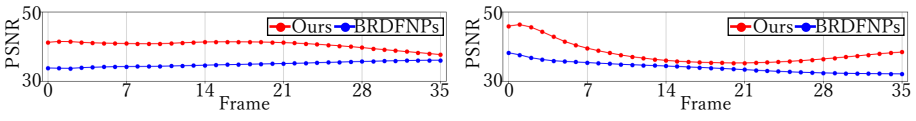


(b) Source: *Copper Patina*, Target: *Copper* from ambientCG.



(c) Source: *Rusting*, Target: *Steel* from the MERL BRDFs.

Fig. 10: Visual results for our real-world experiments. The source time-varying BTFs are obtained from the STAF database [10].



(a) S1 ($\alpha = 0.8$, $\beta = 0.1$) **(b)** S2 ($\alpha = 1.0$, $\beta = 0.2$)

Fig. 11: Plots of PSNRs with and without the proposed permutation-invariant architecture. BRDFNPs [27] is the original architecture without the proposed module.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Bagher, M.M., Snyder, J., Nowrouzezahrai, D.: A non-parametric factor microfacet model for isotropic BRDFs. ACM Trans. Graph. **35**(5), 159:1–159:16 (2016)
3. Ben-Artzi, A., Overbeck, R., Ramamoorthi, R.: Real-time BRDF editing in complex lighting. ACM Trans. Graph. **25**(3), 945–954 (2006)
4. Blinn, J.F.: Models of light reflection for computer synthesized pictures. In: ACM SIGGRAPH Computer Graphics. vol. 11, pp. 192–198 (1977)
5. Burley, B.: Physically-based shading at Disney. In: ACM Trans. Graph. (2012)
6. Cook, R.L., Torrance, K.E.: A reflectance model for computer graphics. ACM Trans. Graph. **1**(1), 7–24 (1982)
7. Dupuy, J., Jakob, W.: An adaptive parameterization for efficient material acquisition and rendering. ACM Trans. Graph. **37**(6), 274:1–274:14 (2018)
8. Fan, J., Wang, B., Hasan, M., Yang, J., Yan, L.Q.: Neural layered BRDFs. In: ACM Trans. Graph. (2022)
9. Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S., Teh, Y.W.: Neural processes. In: Int. Conf. Mach. Learn. Workshop (2018)
10. Gu, J., Tu, C.I., Ramamoorthi, R., Belhumeur, P., Matusik, W., Nayar, S.: Time-varying surface appearance: Acquisition, modeling and rendering. ACM Trans. Graph. **25**(3), 762–771 (2006)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
12. Hu, B., Guo, J., Chen, Y., Li, M., Guo, Y.: Deepbrdf: A deep representation for manipulating measured brdf. In: Comput. Graph. Forum. Wiley Online Library (2020)
13. Kimmel, B.W., Baranoski, G.V., Chen, T.F., Yim, D., Miranda, E.: Spectral appearance changes induced by light exposure. ACM Trans. Graph. **32**(1), 1–13 (2013)
14. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: Int. Conf. Learn. Represent. (2015)
15. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
16. Matusik, W., Pfister, H., Brand, M., McMillan, L.: A data-driven reflectance model. ACM Trans. Graph. **22**(3), 759–769 (2003)
17. Meister, D., Pospíšil, A., Sato, I., Bittner, J.: Spatio-temporal BRDF: Modeling and synthesis. Computers & Graphics **97**(1), 279–291 (2021)
18. Oren, M., Nayar, S.K.: Generalization of Lambert's reflectance model. In: SIGGRAPH (1994)
19. Rusinkiewicz, S.M.: A new change of variables for efficient BRDF representation. In: EGSRT (1998)
20. Smith, B.: Geometrical shadowing of a random rough surface. IEEE transactions on Antennas and Propagation **15**(5), 668–671 (1967)
21. Sztrajman, A., Rainer, G., Ritschel, T., Weyrich, T.: Neural BRDF representation and importance sampling. In: Comput. Graph. Forum. vol. 40, pp. 332–346. Wiley Online Library (2021)
22. Torrance, K.E., Sparrow, E.M.: Theory for off-specular reflection from roughened surfaces. JOSA **57**(9), 1105–1114 (1967)

23. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: Int. Conf. Comput. Vis. (2021)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inform. Process. Syst. (2017)
25. Walter, B., Marschner, S.R., Li, H., Torrance, K.E.: Microfacet models for refraction through rough surfaces. In: EGSRT (2007)
26. Ward, G.J.: Measuring and modeling anisotropic reflection. In: ACM SIGGRAPH Computer Graphics. vol. 26, pp. 265–272 (1992)
27. Zheng, C., Zheng, R., Wang, R., Zhao, S., Bao, H.: A compact representation of measured BRDFs using neural processes. ACM Trans. Graph. **41**(2), 1–15 (2021)