Neural Biplane Representation for BTF Rendering and Acquisition

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

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

Miloš Hašan
Adobe Research
USA
milos.hasan@gmail.com

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 biplane model for the representation, compression, and rendering of bidirectional texture functions (BTFs). One key application of our model is a lightweight pipeline for BTF acquisition. In this scene, we showcase a variety of BTFs, including those from the UBO2014 dataset [Weinmann et al. 2014], synthetic analytical BTFs, and real-world materials that were captured using a cell phone with a collocated flash. Our model demonstrates efficient compression and faithful rendering of BTFs, regardless of whether they were obtained through heavyweight or lightweight capture methods or synthetic data. The insets for each material show the raw data and the visualization of our biplane representation.

†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, Nanjing University of Science and Technology.

SIGGRAPH '23 Conference Proceedings, August 6–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08…$15.00
https://doi.org/10.1145/3588432.3591505

ABSTRACT

Bidirectional Texture Functions (BTFs) are able to represent complex materials with greater generality than traditional analytical models. This holds true for both measured real materials and synthetic ones. Recent advancements in neural BTF representations have significantly reduced storage costs, making them more practical for use in rendering. These representations typically combine spatial feature (latent) textures with neural decoders that handle angular dimensions per spatial location. However, these models have yet to combine fast compression and inference, accuracy, and generality. In this paper, we propose a biplane representation for BTFs, which uses a feature texture in the half-vector domain as well as the spatial domain. This allows the learned representation

to encode high-frequency details in both the spatial and angular domains. Our decoder is small yet general, meaning it is trained once and fixed. Additionally, we optionally combine this representation with a neural offset module for parallax and masking effects. Our model can represent a broad range of BTFs and has fast compression and inference due to its lightweight architecture. Furthermore, it enables a simple way to capture BTF data. By taking about 20 cell phone photos with a collocated camera and flash, our model can plausibly recover the entire BTF, despite never observing function values with differing view and light directions. We demonstrate the effectiveness of our model in the acquisition of many measured materials, including challenging materials such as fabrics.

CCS CONCEPTS

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

KEYWORDS

BTF, neural representation, BTF acquisition

ACM Reference Format:
Jiahui Fan, Beibei Wang†, Miloš Hašan, Jian Yang†, and Ling-Qi Yan. 2023. Neural Biplane Representation for BTF Rendering and Acquisition. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 6–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3588432.3591505

1 INTRODUCTION

To produce vivid and realistic appearances in computer graphics, high-quality materials are among the key requirements. Commonly used material models include analytical models (typically based on microfacet theory) as well as data-driven materials. A common formalization of data-driven appearance is the bidirectional texture function (BTF), which is a 6-dimensional function consisting of two spatial dimensions and four angular dimensions (two each for the incoming and outgoing light). BTFs can characterize a more general appearance than analytical bidirectional reflectance distribution functions (BRDFs), both for measured data and challenging synthetic data. This is particularly true when reproducing non-local effects such as parallax, self-shadowing, and interreflections caused by complex heightfield or non-heightfield geometry (e.g., yarns and fibers in fabrics).

Due to the high dimensionality of BTFs, using them in practical rendering applications is not trivial. The simplest way is employing discrete tables, leading to prohibitive storage costs, which have been addressed through classical compression techniques such as (clustered) PCA [Weinmann et al. 2014] and dictionary learning [Tongbuasirilai et al. 2022]. In recent years, neural networks for representing BTFs have received attention. They mainly focus on efficient compression [Rainer et al. 2019; Rainer et al. 2020; Takikawa et al. 2021] and efficient rendering with parallax effects [Kuznetsov et al. 2021], some also handle silhouette effects [Kuznetsov et al. 2022], while others support material layering [Fan et al. 2022]. They have greatly reduced the storage overhead, making high-dimensional BTF data practically usable in rendering. However, none of these methods offer a combination of high representation accuracy (especially for more specular materials), compressing and rendering speed, and the compression ratio.

The acquisition of real BTFs is a challenge in itself. Exhaustive BTF measurement [Dana et al. 1999] requires thousands of photographs under controlled viewing and lighting directions; this is time-consuming and needs complex setups that have only been built in a small number of labs. Some efforts have been made to make the captured images sparser [Brok et al. 2017], but still, more than ten cameras and hundreds of LEDs are required.

In this paper, we propose a novel neural representation for BTFs. We notice that BTFs are simplified when parallax and normal mapping effects are factored out. Although they remain 6-dimensional, they become approximately 4-dimensional functions of a 2D spatial coordinate $\mathbf{u}$ and a 2D half-vector coordinate $\mathbf{h}$, with a correction that can be effectively learned by a neural model. Furthermore, the spatial and half-vector domains are amenable to a further decomposition. Inspired by the *triplane* representation [Chan et al. 2022], we design a *biplane* feature representation for this problem: one plane (a 2D feature texture) for the half-vector domain and another plane to encode the spatial domain. Look-ups from these two feature maps are concatenated together with the difference vector, and are then decoded with a small but universal multilayer perceptron (MLP). Furthermore, we optionally combine the biplane representation with an offset network [Kuznetsov et al. 2021] to handle parallax effects, leading to further enhanced realism.

The explicit feature grid characterization of the half-vector domain enables richer information in this domain (e.g., sharp specular lobes, controllable highlight falloff), allowing our general MLP decoder to remain small. In contrast, previous methods relied on neural networks to fully handle high frequencies in the angular domain, which resulted in reduced accuracy or the need for a much larger network. The spatial and half-vector planes are optimized per material, which ensures good quality while keeping optimization cost relatively low (since the MLP is small and remains fixed while compressing a given material). Finally, compressing a single BTF of resolution 512×512 takes approximately 5 minutes.

Our representation also allows for efficient light-weight BTF acquisition. By taking about 20 collocated camera-flash images with a cell phone, our model can approximately recover the entire BTF. Despite never observing BTF data points with differing view and light directions, and only observing a subset of half-angle space, our pretrained model encodes prior information about plausible behavior of the unobserved directions, which enables this application. We demonstrated our model's ability for acquisition on several measured materials, including challenging materials like fabrics. In summary, our model for BTF representation and rendering can be effectively used with several different sources of data: heavy exhaustive capture, lightweight capture or synthetic materials with parallax and displacement effects.

2 RELATED WORK

Neural based BTF/BRDF compression. Recently, neural networks have been introduced for BTF/BRDF compression. We categorize these methods into two groups: *specialized methods* that only represent an individual material or BRDF with an over-fitting network,

and *generalized methods* that are capable of representing universal materials.

Among the specialized methods, Kuznetsov et al. [2021] represent a wide range of material appearances at different scales and support parallax effects, which are later improved by Kuznetsov et al. [2022] to handle silhouette effects. By representing each BRDF with a decoder structure, Sztrajman et al. [2021] produce higher quality on specular materials, at the cost of more storage.

The generalized methods usually require large networks to express the universal materials. Rainer et al. [2019] use an encoder-decoder structure per BTF, and it does not generalize across materials, which is improved by Rainer et al. [2020] to represent all materials at the cost of lower quality. Other researches have focused on compressing single BRDFs, like Hu et al. [2020] and Zheng et al. [2021], rather than representing BTFs. Fan et al. [2022] proposed a decoder-only structure that can represent universal BRDFs with high quality. However, due to the large MLP design, their model has a long compression and evaluation time. Also, none of these methods natively handle parallax effects, and they all require dense BTF data as inputs.

Exhaustive BTF capture. Classic acquisition of BTFs [Dana et al. 1999] requires thousands of textures to be captured under controlled viewing and lighting directions. The capture relies on complex equipment, such as gonioreflectometer or gantry setups [Haindl et al. 2012; Sattler et al. 2003], kaleidoscopes [Han and Perlin 2003; Ihrke et al. 2012] or camera arrays ([Müller et al. 2005; Schwartz et al. 2013, 2011]). Some efforts make the required captured data sparser [Brok et al. 2017], but still need more than ten cameras and hundreds of LEDs. In addition to the complexity of the equipment, the capture process is also time-consuming. A recent detailed overview of setups for capturing material appearance can be found in Guarnera et al. [2016]. Our model can effectively compress the data from such exhaustive capture pipelines, and we also propose a lightweight BTF acquisition approach that only needs a cell phone to capture around 20 images.

Lightweight neural SVBRDF acquisition. Lightweight appearance capture with much fewer photos has also been extensively researched. Several works even use a single image as input. Please refer to Guarnera et al. [2016] and Gao et al. [2019] for more comprehensive introductions. Some of these methods focus on recovering SVBRDF maps of stationary textured materials [Aittala et al. 2016, 2015; Zhao et al. 2020] or procedural material parameters [Guo et al. 2020a], while others engage in per-pixel recovery, such as Deschaintre et al. [2018; 2019] and Guo et al. [2021]. Recovered SVBRDF maps are sometimes polluted with highlight burn-in, because of the lighting and BRDF ambiguities, especially for single-image reconstructions. In recent years, generative models [Gao et al. 2019; Guo et al. 2020b; Henzler et al. 2021] are introduced as priors to recover the SVBRDF maps. They partially alleviate the burn-in issue, but all of these methods rely on an analytical BRDF model (e.g., a Cook-Torrance-like microfacet model) whose parameters they approximate. Unlike these methods, our model is not restricted to a specific analytic material model, which allows for a better fit with the real material data. However, it requires more input photos (we found 20 to be sufficient). Additionally, the reconstruction of shapes can also be achieved jointly with the recovery of SVBRDFs by several works [Boss et al. 2020; Li et al. 2018].

3 METHOD

A BTF can be represented as a 6-dimensional function $\rho(\mathbf{u}, \omega_i, \omega_o)$, where $\mathbf{u}$ is the spatial location, and ω_i, ω_o are the incoming and outgoing directions, respectively. Each texel of a BTF can be considered as a generalized BRDF, sometimes termed the apparent BRDF (ABRDF). ABRDFs may or may not satisfy physical BRDF properties, but have the same dimensionality and are parameterized with (ω_i, ω_o).

For clarity, here we define several operations that we will introduce in the following sections of our paper. The *training* procedure refers to the training of our network, i.e., the universal MLP. The *compression* process is that, after the network is trained, we use the network to represent a given BTF by feature planes, and this is achieved by an *optimization* operation. Note that during compression, the weights of the trained network are frozen, and only the biplane features are optimized.

3.1 Design decisions

We will first analyze properties of different previous neural representations of BTFs, and then show the insight of our design. Specifically, we care about *accuracy*, *compression speed*, *evaluation speed*, *network generality*, and *compression ratio*. In Table 1, we compare related works in terms of these properties.

The main factors affecting these goals include the network structure (decoder-only optimization or encoder-decoder), network complexity and the input data layout, which can be dense (regularly sampled) or sparse (randomly sampled). Fan et al. [2022] represent an ABRDF through optimization, and employ a large general MLP with dense input data. Their model is able to handle sharp specular materials effectively. Kuznetsov et al. [2021] use sparse random input data and a small MLP that is over-fitted to each individual material. While not as accurate as Fan et al. [2022] in representing the high specularity, they do support parallax effects through a neural offset module and enable fast rendering. Rainer et al. [2019] use dense input data, while Rainer et al. [2020] can optionally accept sparser input at the cost of accuracy. Both methods train encoders that compress BTFs quickly; however, the representation accuracy is more limited.

Compression speed is important; hours or days are typically not acceptable. For example, Fan et al. [2022] would cost 10 days to represent a BTF with 400×400 resolution due to the optimization and dense input BRDF data.

Rainer et al. [2019] and Rainer et al. [2020] are faster, as they utilize a learned encoder and do not need optimization for each individual material. However, for high-quality reconstruction, their models need dense BTF data as input, leading to limited compression speed when they consume all input queries. Kuznetsov et al. [2021] support relatively fast optimization, thanks to the sparse input data (random queries) and the small MLP.

Evaluation speed mainly depends on the decoder network size. We measure the evaluation speed by counting the number of floating point operations (FLOPs) in each network. Kuznetsov et al. [2021] have the smallest network size among all these approaches, leading

Table 1: A comparison of various BTF representation methods is presented in terms of accuracy, network structure, input data structure, compressing speed, evaluation speed, model generality and compression ratio. Model generality refers to the ability to represent a wide range of materials with a single network. $^{\diamond}$: Can be optionally sparse, but at the cost of accuracy. *: The model by Fan et al. [2022] can handle sharp specular materials quite well, but does not support parallax effects. *: This is an average count for a BTF of 400×400 spatial resolution, as we describe in Section 3.1 and 3.2.

Method	Accuracy	Network type	Data	Comp. speed	Eval. cost (FLOPs)	Universal decoder	Latent dim.
Rainer et al. [2019]	● ●	Encoder-decoder	Dense	● ●	3.5M	✗	8
Rainer et al. [2020]	●	Encoder-decoder	Dense$^{\diamond}$	● ●	3.8M	✓	32
Kuznetsov et al. [2021]	● ● ● ●	Optimization	Sparse	● ● ●	0.2M	✗	7
Fan et al. [2022]	● ● ● ●*	Optimization	Dense	●	105M	✓	96
Ours	● ● ●	Optimization	Sparse	● ● ●	13.5M	✓	18*

to the fastest rendering speed. The methods of Rainer et al. [2019] and Rainer et al. [2020] have more parameters and are slower than Kuznetsov's model. Our model is slightly slower than Rainer's models. The slowest is Fan et al. [2022]'s model due to their large MLP required to handle the sharp specularity.

Network generality means its ability to represent universal BTFs, as opposed to over-fitted to a specific BTF. The models by Kuznetsov et al. [2021; 2022] are specialized for each BTF, as is that of Rainer et al. [2019].

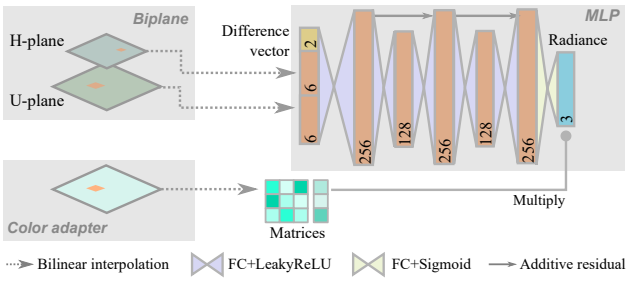

Figure 2: The structure of our representation network. Our network consists of three main components: (i) the biplane feature textures, which are two 3D tensors, individually encodes the spatial and the half-vector domains, (ii) an MLP, comprising six layers with bottleneck structures and skip connections, and (iii) a color adapter that consists of two matrices to adjust the color range of the MLP outputs.

We also compare the compression ratio by comparing the latent feature dimensionality of these methods. All listed approaches require a feature texture to represent all texels of the BTF, but the lengths (numbers of channels) of their feature vectors are different. Kuznetsov et al. [2021] use 7-channel features, achieving the best compression ratio in theory, and Rainer et al. [2019] use 8-channels (though depending on the alignment constraints of hardware memory may make 7 not smaller than 8). Our model includes multiple feature components, and one of their sizes is independent of the BTF's spatial resolution. Thus, on average, we use 18 numbers for each texel (for 400×400 texels). The latent vector sizes of Rainer et al. [2020] and Fan et al. [2022] are 32 and 96 for each BRDF, respectively.

Based on the above analysis, to design a general and practical neural BTF representation, we should follow several rules:

- a small but universal network structure to ensure high-performance evaluation and generality,
- support for sparse input data to enable more efficient compression, and
- an optimization approach instead of a learned encoder, particularly when the input data is sparse.

Following these rules, our design is an optimization-based framework with sparse data as input and a small general MLP. The main question is how to design the architecture to make it as accurate as possible, while keeping the generality at the same time.

3.2 Biplane BTF representation

We will start from the architecture by Fan et al. [2022] (NLBRDF) and then improve upon its shortcomings.

In NLBRDF, each ABRDF is represented as a latent vector V. This latent vector is decoded with a universal MLP $\Theta(V|\omega_i, \omega_o)$ for given light and view directions. Each BTF can be individually regarded as many ABRDFs, and represented with a latent texture, by projecting the ABRDF at each texel to the latent space through optimization:

$$\rho(\mathbf{u}, \omega_i, \omega_o) = \Theta(V_{\mathrm{u}}|\omega_i, \omega_o), \tag{1}$$

where V_{u} represents the latent feature vector at texel $\mathbf{u}$.

Thanks to the universal MLP, this model is general and provides a prior for plausible 4-dimensional ABRDFs. However, to represent the ABRDFs accurately (including sharp specularities), the universal MLP needs to be quite large (about forty layers with 1 million parameters). This is a critical factor causing long compression and evaluation time. Another important observation is that, this method does not consider the shared information among different ABRDFs within a BTF, since the neural network is trained on all ABRDFs from a dataset without considering which ones may plausibly be part of a single BTF. To overcome these issues, we make two decisions: first, we need to make the function decoded by the MLP simpler rather than a full 4-dimensional ABRDF; second, we need to consider the correlation between different ABRDFs of a single BTF in the spatial domain.

We also observe that a 6-dimensional BTF $\rho(\mathbf{u}, \omega_i, \omega_o)$ can be reparameterized with the Rusinkiewicz parameterization [Rusinkie

wicz 1998] using the half and difference vectors, and the difference vector $\mathbf{d}$ has less influence than the spatial and half-vector dimensions. Therefore, we reparameterize the ABRDF to $\rho(\mathbf{u}, \mathbf{h}, \mathbf{d})$, describe the spatial and half-vector dimensions over feature spaces and leave the difference vector dimension for the MLP to approximate. The MLP acts as a decoder for the difference vector combined with the features. Since we decide to include the spatial domain in the BTF representation, here, the features include both a spatial feature and a half-vector feature. Therefore, the formulation of our model is:

$$\rho(\mathbf{u}, \mathbf{h}, \mathbf{d}) = \Theta(V_\mathrm{u}, V_\mathrm{h}|\mathbf{d}) \tag{2}$$

where V_h represents the half-vector feature vector and V_u represents the spatial feature vector.

Finally, we introduce the design of our model, as shown in Figure 2. We propose a *biplane* representation using two feature textures (planes): an H-plane for the half-vector domain and a U-plane for the spatial domain:

$$\begin{cases} \text{H-plane:} & \mathcal{H} = [V_\mathrm{h}]_{r\times r} \in \mathbb{R}^{L_\mathrm{h}} \\ \text{U-plane:} & \mathcal{U} = [V_\mathrm{u}]_{u\times v} \in \mathbb{R}^{L_\mathrm{u}}, \end{cases} \tag{3}$$

where r is the half-vector resolution (set as 20 in practice), (u, v) is the spatial resolution of the BTF, and $L_\mathrm{h}, L_\mathrm{u}$ are feature sizes of the half-vector feature and the spatial feature, respectively. We use the projected hemisphere for half vectors. Note that, since the half-vector feature is shared among all texels, the storage is saved by using a small resolution for the H-plane regardless of the BTF's large spatial resolution.

Our network takes the look-ups of these feature planes as input. Given a query $(\mathbf{u}, \mathbf{h}, \mathbf{d})$, we bilinearly interpolate each feature plane to get an H-feature V_h and a U-feature V_u. These two features are concatenated and fed into a small MLP, together with the difference vector $\mathbf{d}$. The MLP outputs the final reflectance value for the given query. Our architecture remains small: the MLP has 6 layers, and the feature sizes for the spatial domain and half-vector domain are both 6.

During training on a dataset of BTFs, the features of two planes are learned at the same time as the MLP parameters. Once the network is trained, it can represent any new BTF; to do this, the parameters of the MLP are frozen, and only the two feature planes are optimized. Thus, our network is general and can represent any BTF, though ideally the initial training BTFs should cover the types of materials we would like to eventually represent. Note that since the MLP is small, the optimization of a new BTF is efficient.

3.3 Color Adapter

Our current network is already able to represent a variety of BTFs. However, we observed some color bias when representing unseen BTFs. The main reason for the color bias is the limited measured BTF dataset for training, which is not capable of covering all real-world materials, especially for color variations.

To overcome this issue, we propose a *color adapter* module to enhance the capability of representing various colors in our representation network by extending the range of colors in the latent space. For that, we perform a linear transform on the predicted value from the MLP, using a weight component and a bias component. The weight component A_u is an $\mathbb{R}^{3\times3}$ matrix and the bias component B_u is an $\mathbb{R}^{3\times1}$ matrix. Therefore, our final representation is expressed as follows:

$$\rho(\mathbf{u}, \mathbf{h}, \mathbf{d}) = A_\mathrm{u}\Theta(V_\mathrm{u}, V_\mathrm{h}|\mathbf{d}) + B_\mathrm{u}. \tag{4}$$

We apply this color adapter module only during BTF compression and not during training. In practice, we found that simultaneously optimizing the feature planes and the adapter leads to noticeable reflectance distribution differences. Thus, we perform a two-step optimization strategy: fit the grayscale intensity first, and then fine-tune the color. For that, in the first step, we freeze the adapter with an average initialization, and only optimize the feature planes. Then, in the second step, we only optimize the color adapter to adjust the colors. Note that, although the loss is computed on an averaged value in the first step, the output of the MLP is still 3-channel. Thanks to the two-step optimization strategy, the reflectance distribution, as well as the color, is plausible.

3.4 Offset module for parallax effects

To further enhance the representation of non-planar materials, we optionally use an offset module to enable parallax effects, similar to Kuznetsov et al. [2021].

The basic idea of the offset module is mapping a texture-space location $\mathbf{u}$ to a new location, given a viewing direction as additional input. Here, we directly learn the offset vector. This mapping is learned similarly to the representation model but with fewer dimensions:

$$\Delta\mathbf{u}(\mathbf{u}, \omega_o) = \Theta_\mathrm{off}(V_\mathrm{u}^\mathrm{off}|\omega_o), \tag{5}$$

where Θ_off is an MLP (4 layers of 32 hidden units), $V_\mathrm{u}^\mathrm{off}$ represents the spatial feature (6 channels) with the same resolution as the U-plane and ω_o is the view direction.

We do not train a general MLP for the offset module, since the network is small, and over-fitting each BTF is very fast. Given a BTF, we jointly optimize the offset module and the biplane features of the representation network. More specifically, given a query $(\mathbf{u}, \mathbf{h}, \mathbf{d})$, we first apply the offset module to get the new location $\mathbf{u}_\mathrm{new}$, and feed $(\mathbf{u}_\mathrm{new}, \mathbf{h}, \mathbf{d})$ to the representation network to produce the final radiance. This way, the biplane and offset texture features are optimized, and the offset MLP is learned. Note that the optimization for each BTF is computationally efficient (5 mins), and the storage is acceptable.

To summarize, a BTF is represented by two feature planes and a color adapter in the representation network, and one feature plane together with a tiny MLP in the offset network (if it is needed).

3.5 BTF Acquisition

We further propose a lightweight BTF acquisition approach. We use a hand-held cell phone to take 20 images of a flat sample with the collocated flash as the light source, thus assuming all view and light directions are identical.

The problem becomes optimizing a BTF that matches the input images. We formulate the problem as follows:

$$\min\left(\sum_{j=1}^{j=N}(G(I_j(\mathbf{u})) - \rho_t(\mathbf{u}, \mathbf{h}_j, \mathbf{d}_j))\right), \tag{6}$$

where N is the image count, I is the captured images, G represents the energy falloff by distance, and ρ_t represents the targeting BTF.

Since each image is taken w.r.t. a light and view direction, we use $\mathbf{h}_j$ and $\mathbf{d}_j$ to indicate the relative directions between each pixel at position $\mathbf{u}$ and the camera for image I_j. Note that by collocated captures, all the difference vectors are always zero, which we believe is challenging for common acquisition methods.

A simple way is to optimize Equation 6 directly by treating these images as queries to the representation network and directly optimizing the two feature planes with no additional regularization. However, the collocated images lack of directional information, resulting in undefined areas for unseen directions in the H-plane. Therefore, inspired by Den Brok et al. [2015], we treat it as a linear combination of the H-planes from the seen BTFs in our training dataset, instead of optimizing the H-plane directly. This can be reformulated as:

$$\mathcal{H}_t = \sum_{i=1}^{i=M} w_i \mathcal{H}_i, \tag{7}$$

where M is the number of known BTFs in training dataset and w_i is the weight of ith BTF's H-plane $\mathcal{H}_i$. In this way, we are learning the weights of these known H-planes to construct the target BTF. This linear combination resolves the issues with direct optimization, as shown in Section 5.2.

4 IMPLEMENTATION DETAILS

4.1 Data preparation

To train our network, we use the measured BTF data from UBO2014 dataset [Weinmann et al. 2014], which includes 84 BTFs of 400×400 texels from seven different categories. Particularly, in Figure 7, we use Rainer et al. [2020]'s training set split for comparison, which is a 91% proportion of the UBO2014 dataset. For each BTF, we resample it into 6.4×10^7 random queries. For each query, both the incoming-outgoing direction and the UV location are independently and randomly determined.

4.2 Network training

We implement our model and train our representation network using the AdamW optimizer in PyTorch [Paszke et al. 2019]. For each training step, we simultaneously train the shared network weights and individual feature planes of 4 BTFs, by taking 160,000 random queries from each of them in a batch. We start from learning rate 1×10^{-3} for feature planes and 3×10^{-4} for network weights and use an exponential learning rate scheduler for both with a 0.9 decay ratio per epoch. The training phase of 30 epochs took us 18 hours on an RTX 2080Ti GPU (11 GB).

4.3 BTF compression and acquisition

To compress a BTF, we perform a two-stage optimization strategy, as we described in Section 3.3. In the first step (15 epochs), we initialize the biplane and the optional offset texture (to zeros) and optionally the offset network. We optimize all the feature planes (learning rate 1×10^{-2}) and simultaneously train the optional offset network from scratch (learning rate 3×10^{-3}). Additionally, we apply a 2D Gaussian blurring to all feature planes, with a decaying kernel size from 20 to 0. After that, in the second step (5 epochs), we only optimize the color adapter (learning rate 1×10^{-2}).

For the lightweight BTF acquisition application, we capture about 20 pictures and process each calibrated and rectified picture into 400×400 collocated queries. We set the batch size to 40,000, run for 35 + 15 epochs instead of 15 + 5, and keep the other settings. In practice, we use a printed frame with markers to calibrate our camera captures, and then calculate all pixels' actual distance from the camera and view/light directions by calibrated results. We capture 20 images for a sample. On average, the capture and data preparation of each BTF material costs 10 mins and the optimization costs 3.5 mins each.

4.4 BTF rendering

We have implemented our method with the Mitsuba renderer [Jakob 2010] and PyTorch [Paszke et al. 2019]. We use the Mitsuba renderer to get the shading points (UV, incoming and outgoing directions) together with the direct lighting, and store them in buffers. Then we compute the BRDF value by evaluating our BTF representation network in PyTorch on the GPU, and this value is later multiplied with the stored lighting to generate the final rendered images. A 1920×1080 rendering (1 spp) takes about 1 second by our implementation.

For rendering with complex illumination, we use multiple importance sampling in our rendering pipeline. For that, the buffers also store the contributions from light sampling and BRDF sampling together with sampling weights and probability distribution functions (pdfs). For simplicity, we use Lambertian sampling for all BTFs; a straightforward extension would be to predict the weight and width of an additional Gaussian or Blinn-Phong lobe [Fan et al. 2022; Sztrajman et al. 2021].

5 RESULTS

In this section, we compare our method with previous works, including Rainer et al. [2020] and Fan et al. [2022]. All the implementations are taken from the authors' websites.

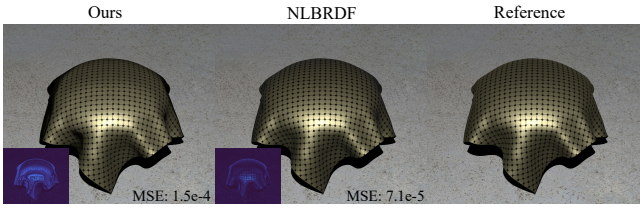

Figure 3: Comparison between our model and NLBRDF [Fan et al. 2022] on synthetic BTF. NLBRDF produces higher quality at the cost of extremely long compression time (8 days), while our model only costs 5 minutes.

5.1 Quality validation

We first validate our representation and offset networks, and then illustrate our BTF capturing quality.

Synthetic BTF data. We compare our representation networks (with and without an offset module), Rainer et al. [2020] and ground truth computed by path tracing of the synthetic material structure in Figure 6. Rainer et al. [2020] show an over-blurring appearance

in the first two rows with apparent parallax effects, since their method was not designed to handle these. A color bias is visible in the bottom row. Even without the offset module, our model produces higher quality than Rainer et al. [2020]. Introducing the offset module allows more noticeable parallax effects and matches the reference better.

In Figure 3, we compare our model with NLBRDF [Fan et al. 2022] on synthetic data. To represent such a BTF, NLBRDF needs 200 hours, while our model only needs 5 minutes. Regarding the storage cost, our model is 5× smaller than NLBRDF. On the other hand, NLBRDF produces higher quality than ours, while our result has darker edges, since there are not enough samples at grazing angles by the random sampling. Note that we do not observe this darkening effect on other examples besides this rough conductor material.

Real BTF data. We also validate the effectiveness of our model on real BTF data, by comparing our representation network with Rainer et al. [2020] in Figure 7. We use their pre-trained model to generate their results. For each BTF, we use only about 400 random samples per texel, while Rainer et al. [2020] use all 22,801 uniform queries from the dataset. By comparison, we find that with even sparser data input, our model still outperforms Rainer et al. [2020] on both seen and unseen data regarding rendering quality. Our model significantly reduces the required data and has a higher representation ability.

BTF capturing. Our representation network allows a lightweight BTF acquisition, by capturing 20 images only with a cell phone. In Figure 8, we show five examples, including fabrics and leathers. The rendered results with the captured materials are faithful under novel view/light conditions.

Complex scenes. In Figure 1, we show a DecorativeSet scene with a variety of materials, including captured BTFs with our capturing method, BTFs from UBO2014 dataset and synthetic BTFs. We use our representation network to express all these materials.

5.2 Ablation study

Linear combination for BTF capturing. In our BTF acquisition, we use a linear combination strategy to learn the H-plane features rather than optimizing them directly. We validate the influence of the linear combination strategy in Figure 4. Learning the features directly leads to a discontinuous feature texture, showing highlight artifacts in the rendered images. On the contrary, introducing the linear combination results in a continuous feature texture and produces more plausible rendered results.

Color adapter. The color adapter is a critical component in our representation network. We show its impact in Figure 5 on two synthetic materials, including specular and diffuse ones. In both cases, the results without adapter have noticeable color bias, while the adapter overcomes this issue and better matches the references.

5.3 Discussion and limitations

Expressiveness of our representation network. Our representation network can represent various materials, from synthetic to measured data. However, our model shows less accuracy than Fan et al. [2022] on high-specular materials, since our network is designed to be smaller as a trade-off. For the generality, we use a larger MLP than Kuznetsov et al. [2021] to make our model universal, which also limits the efficiency of our network. Additionally, there are some exotic materials with angularly-varying colors that our model fails to represent, as we show in Figure 9. That is because there are no such examples in the UBO2014 dataset [Weinmann et al. 2014]. One potential solution could be adding some synthetic materials into our training dataset and retrain the network.

Acquisition ability. Our universal representation network enables an easy BTF acquisition from real-life materials by cell phone

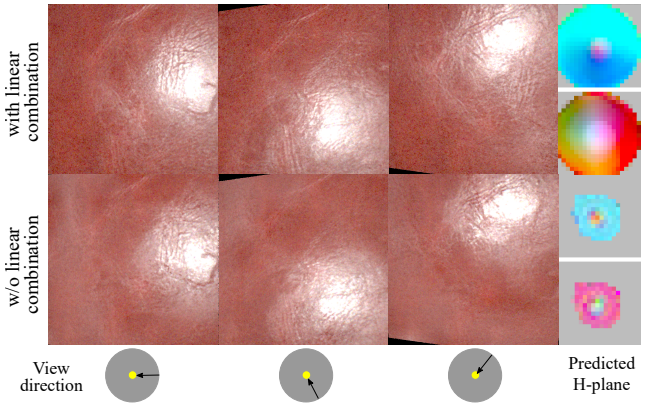

Figure 4: The impact of the linear combination strategy in our BTF capturing. Directly optimizing the features of the H-plane (without linear combination) results in incomplete discontinuous feature textures (rightmost column) and prominent artifacts around the highlights, while optimizing the coefficients of the linear combination produces more reliable results. We arrange the H-planes (6 channels) into 2 RGB images for visualization. The view directions are shown in the bottom row.

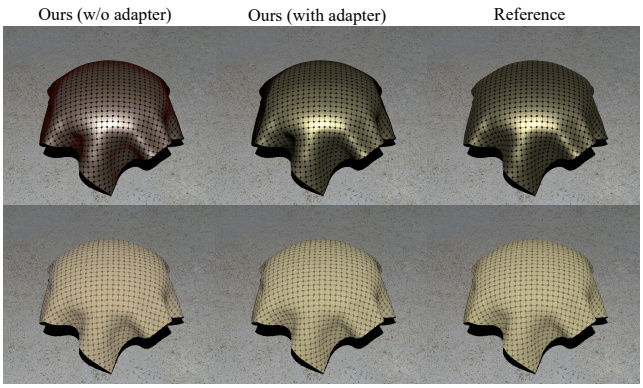

Figure 5: The influence of our color adapter. The rendered results without the color adapter (left column) show obvious color bias, which is solved by the proposed color adapter (middle column).

cameras. However, although our model can handle synthetic BTFs with heightfield and parallax effects, it does not take effects in the BTF acquisition. We find it difficult for the offset module to converge with the insufficient information in collocated captures.

Diversity of our training dataset. We train our model on the UBO-2014 BTF dataset, and our BTF acquisition pipeline also relies on the priority that learned from this collection of BTFs. We believe that those BTFs cover a good range of common materials, however, it is far from every one of them. For some exotic materials, we may need to extend our training dataset, so that our model can represent them as well.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented a new BTF representation model. Each BTF is represented by two feature planes, together with a universal MLP shared by all materials. By considering both angular and spatial domains as feature spaces, our model is able to represent BTFs more accurately with faster compression and evaluation speed than previous work. We also support an offset network to handle parallax effects, enhancing realism. Our compact representation also allows for a lightweight BTF acquisition application. By taking 20 collocated images with a cell phone, our approach can plausibly recover the entire BTF; this could become a lightweight tool for capturing BTFs, and would not be possible without our universal MLP. In summary, our new biplane representation offers effective compression and rendering for BTF data from multiple sources: exhaustive heavy capture, lightweight capture and synthetic NeuMIP-style materials [Kuznetsov et al. 2021].

There are still many potential future researching directions. Regarding the representation ability, combining our representation network with other advanced material capture networks (e.g., MaterialGAN [Guo et al. 2020b]) may improve recovery quality. From the perspective of material diversity and compatibility, representing the glinty materials with a neural network is also promising, since the traditional compression method (e.g., tensor decomposition) has shown high accuracy for representing such high-frequency effects [Deng et al. 2022]. Another interesting potential work is combining neural material with popular text-to-texture approaches [Chen et al. 2022] to achieve a simpler pipeline for neural texture acquisition.

ACKNOWLEDGMENTS

We thank the reviewers for the valuable comments. This work has been partially supported by the National Key R&D Program of China under grant No. 2022ZD0116305, National Natural Science Foundation of China under grant No. 62172220 and "111" Program under grant No. B13022.

REFERENCES

Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance Modeling by Neural Texture Synthesis. *ACM Trans. Graph.* 35, 4 (2016), 1–13.

Miika Aittala, Tim Weyrich, Jaakko Lehtinen, et al. 2015. Two-shot SVBRDF capture for stationary materials. *ACM Trans. Graph.* 34, 4 (2015), 110–1.

Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. 2020. Two-shot spatially-varying brdf and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3982–3991.

Dennis den Brok, Michael Weinmann, and Reinhard Klein. 2017. Towards Sparse and Multiplexed Acquisition of Material BTFs. In *Workshop on Material Appearance Modeling*, Reinhard Klein and Holly Rushmeier (Eds.). The Eurographics Association. https://doi.org/10.2312/mam.20171326

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16123–16133.

Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. 2022. TANGO: Text-driven Photorealistic and Robust 3D Stylization via Lighting Decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. 1999. Reflectance and Texture of Real-World Surfaces. *ACM Trans. Graph.* 18, 1 (jan 1999), 1–34. https://doi.org/10.1145/300776.300778

Dennis Den Brok, Michael Weinmann, and Reinhard Klein. 2015. Linear Models for Material BTFs and Possible Applications.. In *Material Appearance Modeling*.

Hong Deng, Yang Liu, Beibei Wang, Jian Yang, Lei Ma, Nicolas Holzschuch, and Ling-Qi Yan. 2022. Constant-Cost Spatio-Angular Prefiltering of Glinty Appearance Using Tensor Decomposition. *ACM Transactions on Graphics* 41, 2 (2022), 22:1–22:17.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-Image SVBRDF Capture with a Rendering-aware Deep Network. *ACM Trans. Graph.* 37, 4 (2018), 1–15.

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible svbrdf capture with a multi-image deep network. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 1–13.

Jiahui Fan, Beibei Wang, Milos Hasan, Jian Yang, and Ling-Qi Yan. 2022. Neural Layered BRDFs. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3528233.3530732

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 134.

D. Guarnera, G.C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. 2016. BRDF Representation and Acquisition. *Computer Graphics Forum* 35, 2 (2016), 625–650. https://doi.org/10.1111/cgf.12867 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12867

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-aware Two-stream Network for Single-image SVBRDF Acquisition. *ACM Trans. Graph.* 40, 4 (2021), 1–14.

Yu Guo, Miloš Hašan, Lingqi Yan, and Shuang Zhao. 2020a. A bayesian inference framework for procedural material parameter estimation. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 255–266.

Y. Guo, C. Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020b. MaterialGAN: Reflectance Capture using a Generative SVBRDF Model. *ArXiv* abs/2010.00114 (2020).

Michal Haindl, Jiří Filip, and Vavra Radomir. 2012. Digital Material Appearance: the Curse of Tera-Bytes. *ERCIM News* (01 2012), 49–50.

Jefferson Y. Han and Ken Perlin. 2003. Measuring Bidirectional Texture Reflectance with a Kaleidoscope. *ACM Trans. Graph.* 22, 3 (jul 2003), 741–748. https://doi.org/10.1145/882262.882341

Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. 2021. Generative Modelling of BRDF Textures from Flash Images. *ACM Trans. Graph.* 40, 6 (2021), 1–13.

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.

Ivo Ihrke, Ilya Reshetouski, Alkhazur Manakov, Art Tevs, Michael Wand, and Hans-Peter Seidel. 2012. A kaleidoscopic approach to surround geometry and reflectance acquisition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 29–36. https://doi.org/10.1109/CVPRW.2012.6239347

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

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.

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2022. Rendering Neural Materials on Curved Surfaces. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 9, 9 pages. https://doi.org/10.1145/3528233.3530721

Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. 2005. Acquisition, Synthesis, and Rendering of Bidirectional Texture Functions. *Computer Graphics Forum* 24, 1 (2005), 83–109. https://doi.org/10.1111/j.1467-8659.2005.00830.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2005.00830.x

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

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

Szymon M Rusinkie wicz. 1998. A new change of variables for efficient BRDF representation. *Rendering techniques* 98 (1998), 11–22.

Mirko Sattler, Ralf Sarlette, and Reinhard Klein. 2003. Efficient and Realistic Visualization of Cloth. In *Proceedings of the 14th Eurographics Workshop on Rendering* (Leuven, Belgium) *(EGRW '03)*. Eurographics Association, Goslar, DEU, 167–177.

Christopher Schwartz, Ralf Sarlette, Michael Weinmann, and Reinhard Klein. 2013. DOME II: A Parallelized BTF Acquisition System. In *Proceedings of the Eurographics 2013 Workshop on Material Appearance Modeling: Issues and Acquisition* (Zaragoza, Spain) *(MAM '13)*. Eurographics Association, Goslar, DEU, 25–31.

C. Schwartz, M. Weinmann, R. Ruiters, and R. Klein. 2011. Integrated High-Quality Acquisition of Geometry and Appearance for Cultural Heritage. In *Proceedings of the 12th International Conference on Virtual Reality, Archaeology and Cultural Heritage* (Prato, Italy) *(VAST'11)*. Eurographics Association, Goslar, DEU, 25–32.

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.

Tanaboon Tongbuasirilai, Jonas Unger, Christine Guillemot, and Ehsan Miandji. 2022. A Sparse Non-parametric BRDF Model. *ACM Transactions on Graphics* 41, 5 (2022), 1–18.

Michael Weinmann, Juergen Gall, and Reinhard Klein. 2014. Material Classification Based on Training Data Synthesized Using a BTF Database. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*. Springer Internatifonal Publishing, 156–171.

Yezi Zhao, Beibei Wang, Yanning Xu, Zheng Zeng, L. Wang, and N. Holzschuch. 2020. Joint SVBRDF Recovery and Synthesis From a Single Image using an Unsupervised Generative Adversarial Network. In *EGSR*. Wiley.

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.

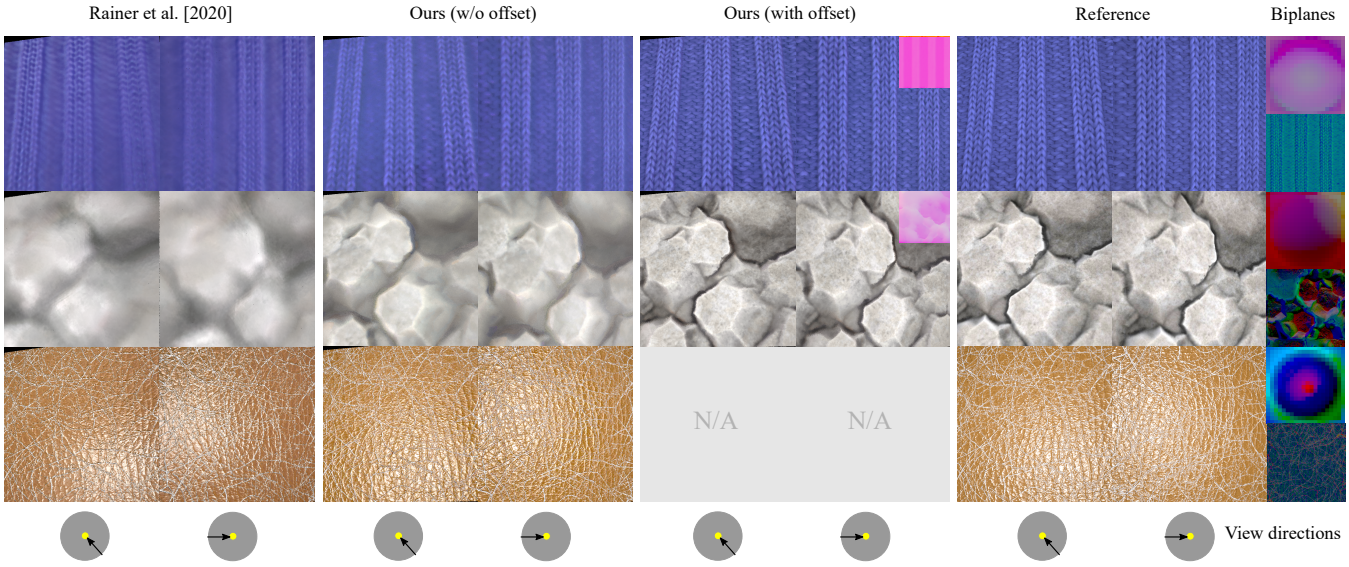

Figure 6: Comparison among our model (without and with offset module), Rainer et al. [2020] and the ground truth on synthetic BTFs. The BTF data is collected by performing path tracing on simulated heightfields in renderers. For BTFs with complex parallax effects (the first and second rows), Rainer et al. [2020] exhibit blurry results, while our model (without offset) gives layouts more clearly. Our model (with offset) provides the closest match to the reference. For flat material (third row), Rainer et al. [2020] show a noticeable color bias, while our model (without offset) has a good agreement with the reference. Note that for BTFs without a heightfield, we do not use the offset module for compression and prediction.

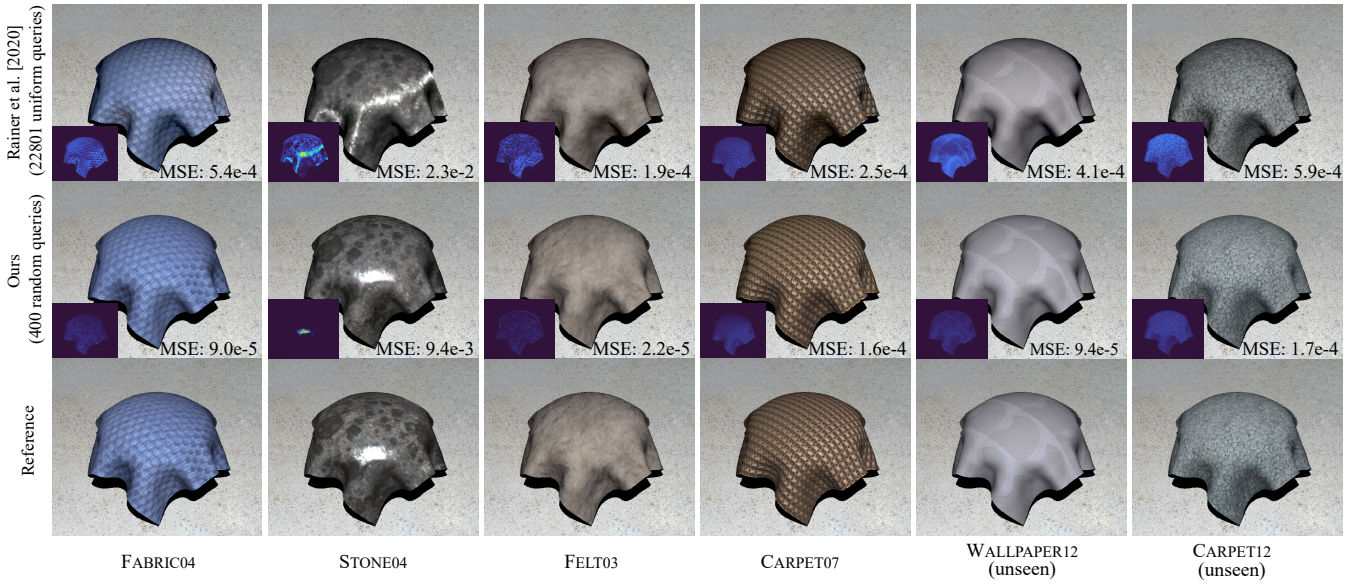

Figure 7: Comparison among our model, Rainer et al. [2020] and the references on real BTF data from UBO2014 dataset [2014]. Our model shows much higher quality than Rainer et al. [2020] both visually and quantitatively, even with sparser BTF data.

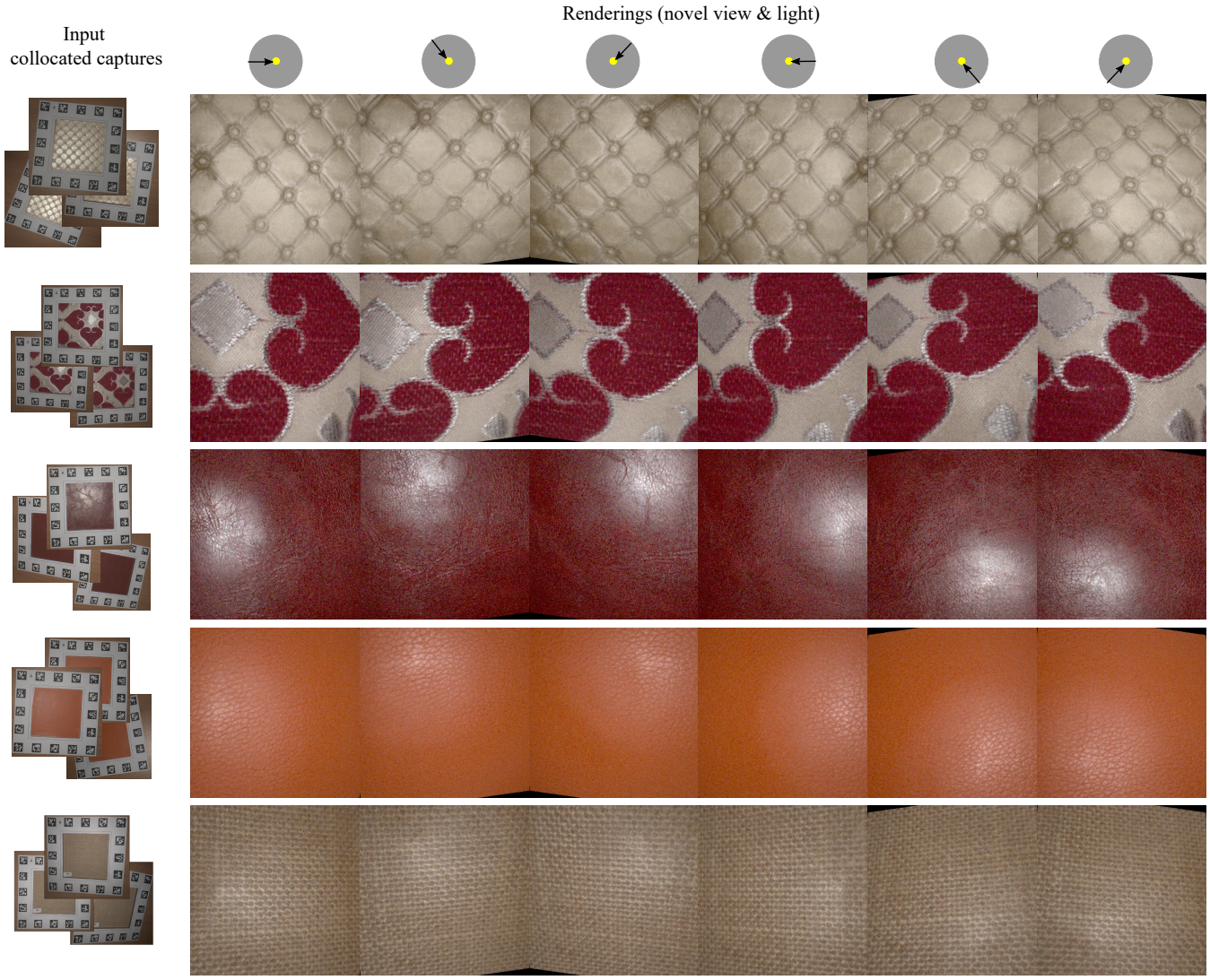

Figure 8: Results of BTF acquisition on several materials (e.g., leathers and fabrics). We capture 20 images with a cell phone for each material, then recover these materials with our representation network. With such a sparse capture, our model can recover reliable materials, thanks to the prior information learned from the dataset.

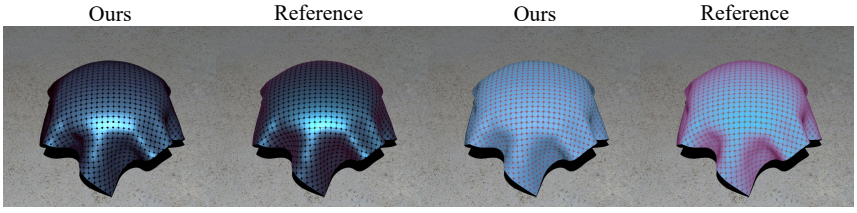

Figure 9: Results of our representation network on materials with angularly varying colors. Because the appearances of such materials are way far from those in our training dataset, our representation network cannot handle them well.