A Dynamic By-example BTF Synthesis Scheme

ZILIN XU, University of California, Santa Barbara, USA
ZAHRA MONTAZERI, University of Manchester, United Kingdom
BEIBEI WANG, Nanjing University, China
LING-QI YAN, University of California, Santa Barbara, USA

Fig. 1. We propose a novel by-example BTF synthesis scheme that can dynamically synthesize a non-repetitive, infinitely large BTF from a small example BTF (400 × 400). To demonstrate our method's effectiveness in both synthesis and rendering, we select four representative example BTFs (shown at the top-right corners) with unique visual effects from UBO2014 [Weinmann et al. 2014]. Each of them uses our Triple Plane with histogram-preserving blending [Heitz and Neyret 2018] and is rendered under two different lighting conditions. Our method managed to synthesize a non-repetitive BTF while faithfully capturing the complex visual effects of each BTF.

Measured Bidirectional Texture Function (BTF) can faithfully reproduce a realistic appearance but is costly to acquire and store due to its 6D nature (2D spatial and 4D angular). Therefore, it is practical and necessary for rendering to synthesize BTFs from a small example patch. While previous methods managed to produce plausible results, we find that they seldomly take into consideration the property of being *dynamic*, so a BTF must be synthesized before the rendering process, resulting in limited size, costly pre-generation and storage issues. In this paper, we propose a dynamic BTF synthesis scheme, where a BTF at any position only needs to be synthesized when being queried. Our insight is that, with the recent advances in neural dimension reduction methods, a BTF can be decomposed into disjoint low-dimensional components. We can perform dynamic synthesis only on the positional dimensions, and during rendering, recover the BTF by querying and combining these low-dimensional functions with the help of a lightweight Multilayer Perceptron (MLP). Consequently, we obtain a fully dynamic 6D BTF synthesis scheme that does not require any pre-generation, which enables efficient rendering of our infinitely large and non-repetitive BTFs on the fly. We demonstrate the effectiveness of our method through various types of BTFs taken from UBO2014 [Weinmann et al. 2014].

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Rendering, Appearance, BTF, By-example

ACM Reference Format:
Zilin Xu, Zahra Montazeri, Beibei Wang, and Ling-Qi Yan. 2024. A Dynamic By-example BTF Synthesis Scheme. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3680528.3687578

SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687578

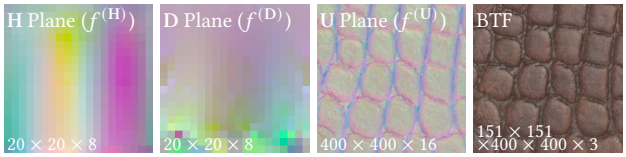

Fig. 2. The direct visualization of our feature planes. The positional feature plane shows highly semantic characteristics with detailed structures and variation information close to the BTF. The resolution of each feature plane follows Biplane [Fan et al. 2023]'s setting.

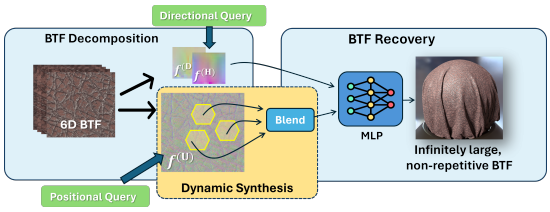

Fig. 3. Our method's pipeline. We first decompose 6D BTF into three 2D planes ($f^{(U)}$, $f^{(H)}$ and $f^{(D)}$), then perform dynamic synthesis on the positional plane as if it is a 2D texture. The synthesized reflectance is recovered via a lightweight MLP with the input of positional and directional features.

1 INTRODUCTION

Photorealistic rendering places great emphasis on accurately reproducing the appearance of real-world objects. In general, the appearance of a virtual object's surface is established by a Bidirectional Reflectance Distribution Function (BRDF). The BRDFs can be empirical, analytical, data-driven, or hybrid. While the empirical and analytical BRDF models can produce a plausible appearance, they are essentially simulations of real-world appearance and may disregard certain intricate visual effects. The data-driven appearance models are usually derived from direct measurements of a piece of a real-world object, resulting in measured Spatially Varying Bidirectional Reflectance Distribution Functions (SVBRDFs) or Bidirectional Texture Functions (BTFs) — the faithful replications of the real-world appearance. However, the data-driven materials demand a substantial amount of measured data since the SVBRDFs or BTFs are typically 6D functions with 2D spatial and 4D angular, creating difficulties in both data acquisition and utilization during rendering. Therefore, leveraging dimension reduction or decomposition is essential. Nevertheless, due to the high-dimensional nature of materials and the common assumption of far-field illumination during the measurement process, the feasible measurement area is restricted to a small patch (e.g., 400×400 texels) of the example object, as the exhaustive measurement on an entire object (e.g., a garment) is hard to accomplish. Applying a material with only such a small example patch in rendering will produce undesirable spatial repetitive patterns despite its high fidelity. Therefore, a faithful BTF synthesis scheme is important to practically bring the real-world appearance into rendering.

Previous work on material synthesis often involves creating node graphs of procedural materials, or generating G-buffers/2D parametric maps of the BRDF models (usually based on microfacet models), creating an analytical BTF representation (e.g., MATch [Shi et al. 2020]). These methods can be efficient, but still *synthetic*, thus inherently inaccurate compared to the measured BTFs because the real-world appearance has a geometric and reflectance complexity that is difficult to describe using analytic BRDFs, let alone that some of those BRDFs are further simplified (uniform, isotropic, etc.) such as PhotoMat [Zhou et al. 2023]. Moreover, those synthesis methods are usually coupled with reconstruction — they first *estimate* the appearance from some inputs, commonly one or a few photographs taken by a cellphone with a flashlight. Consequently, their quality is further compromised, often losing positional details and angular resolution. Therefore, they are unsuitable for achieving accurate appearances compared to properly using the measured BTFs.

Many BTF synthesis methods have been proposed in the past decades to generate a large BTF from a small measured BTF patch. These methods (e.g., Tong et al. [2002] and Koudelka et al. [2003]) may produce a plausible synthesis result with a better-preserved structure due to the quilting-based (i.e., similar to texture quilting [Efros and Freeman 2001]) synthesis. However, they rely on highly compressed BTF representation such as principal component analysis (PCA) and Spherical Harmonics (SH), which introduces significant quality loss. Moreover, to the best of our knowledge, all of these methods are *non-dynamic* and cannot be tiled infinitely. The large BTF must be first generated to a proper size (e.g., $4K \times 4K$) and then be used in rendering process. Even with their dimension reduction methods (e.g., PCA) applying to the BTF, pre-generating a very large BTF on the compressed low-dimension domain and then storing it is still not practical. That is why a dynamic property is in high demand, yet it has been ignored in past decades.

In this paper, we challenge the problem of *dynamically* synthesizing an infinitely large BTF without spatial repetition from a given example of measured BTF. Our insight is that with the advance of previous dimension reduction methods, it is possible to decompose a 6D BTF into the outer product ($\circ$) of 2D neural feature planes (one positional and two directional planes, possibly multi-channel). We call this decomposition approach *Triple Plane*[1]. After the decomposition, we find the decomposed 2D positional neural feature plane still exhibiting semantic information that has a similar visual appearance to the BTF, as shown in Fig. 2, which inspires us to perform texture synthesis in the position domain *as if they are textures*. In this way, the synthesized BTF can be recovered from the synthesized positional feature and the direction features via a lightweight Multilayer Perceptron (MLP). With the analysis, we reduce the BTF synthesis task into two key sub-tasks: a faithful decomposition capturing the full 6D BTF without losing details, and a synthesis method that is able to preserve content and allow dynamic queries, i.e., query without pre-generating a large texture, to steer away from heavy storage and runtime rendering cost.

Consequently, we are able to essentially generate infinitely large, non-repetitive BTF that can be directly queried during rendering without any storage overhead — all we need as input is just a measured BTF example patch. Our scheme is general and compatible

[1]We name it Triple Plane to distinguish from the Triplane method [Chan et al. 2022] that uses three intersecting planes to represent a 3D function instead of our three disjoint 2D planes that decompose a 6D BTF.

with both different neural-based BTF representation methods and texture synthesis (we specifically focus on dynamic by-example synthesis). Fig. 3 shows the overall pipeline of our scheme, and in Sec. 4, we demonstrate the high-quality BTFs we synthesized on a variety of objects with a fast, lightweight MLP. It demonstrates not only the novelty of our idea to be the first of its kind to dynamically generate BTFs on a decomposed low-dimension but also the practicality of our scheme. In summary, our main contributions are as follows:

- a novel, simple but effective approach that enables dynamic, infinitely large, and non-repetitive BTF synthesis,
- a general BTF synthesis scheme that is compatible with different neural dimensional decomposition techniques and different texture synthesis methods, and,
- a Triple Plane approach for BTF representation that decomposes the 6D BTF into neural outer products of three 2D functions and achieves high fidelity in real-time performance.

2 RELATED WORK

2.1 BTF Acquisition

BTF can be accurately measured through specialized equipment that traverses and measures each pair of the incident and outgoing angles individually (very precise, but producing a large amount of data), or it can be roughly estimated from one or a few photographs using neural networks. Neural-based BTF acquisition methods have become popular in recent years because of their convenience. However, the quality of these methods is far from comparable to measured BTFs since they are either capturing 4D isotropic BTF [Zhou et al. 2023] or estimating 2D parametric maps (e.g., normal, albedo, etc.) for the analytical appearance model [Deschaintre et al. 2018; Gao et al. 2019; Guo et al. 2021, 2020; Henzler et al. 2021; Zhou et al. 2022]. To achieve a highly realistic appearance, utilizing the measured BTFs is essential. The first measured BTF dataset (CUReT) was built by Dana et al. [1999]. There are several open-source measured BTF datasets and UBO2014 [Weinmann et al. 2014] is the most commonly used one with 400×400 spatial and 151×151 angular resolution. Unfortunately, due to the high-dimensional nature of BTF and the physical limitations of the measurement technique, the measured BTF is restricted in small size, making it unsuitable for many scenarios, as it would produce unsatisfactory repetitive tiling patterns on objects. To overcome this limitation, we propose a novel BTF synthesis approach, which generates an infinitely large without repetition from only a small example patch.

2.2 BTF Compression & Dimension Decomposition

Due to the high dimensionality of BTF, dimension decomposition methods like Principal Component Analysis (PCA) and Tensor Decomposition (TD) are essential in compressing BTF. There is a substantial amount of work attempting to apply PCA [Guthe et al. 2009; Koudelka et al. 2003; Ruiters et al. 2009] and tensor decomposition [Ruiters and Klein 2009; Vasilescu and Terzopoulos 2004] for BTF compression. With the rapid development of deep learning, neural compression became popular for material representation [Fan et al. 2022; Hu et al. 2020; Sztrajman et al. 2021; Zheng et al. 2021] because they frequently outperformed classic methods. Rainer et al. [2019] proposed the first neural BTF compression method with an auto-encoder architecture. Later, Rainer et al. [2020] introduce a unified network that projects different BTF onto a shared latent space. [Sztrajman et al. 2021]

The aforementioned works are only designed to compress the BTF. From another perspective, neural dimension decomposition methods that expose the 2D positional neural texture are more effective for BTF representation recently. NeuMIP [Kuznetsov et al. 2021] and its later extended works [Kuznetsov et al. 2022; Xue et al. 2024] can also be used for BTF compression. They have single or multiple neural textures progressively learned during training. The closest work to us is Biplane [Fan et al. 2023], which decomposes BTF into a 2D positional plane and a 2D half-vector plane. With a conditional input of a 2D difference vector, it can recover the reflectance of different BTF from a universal but large MLP. However, none of these methods completely decomposed the directional and position dimensions, i.e., the direction as a condition input to the positional feature instead of an independent feature. That means the directional information is implicitly mixed and stored with the positional features (or latent vectors). In this paper, we focus on neural dimension decomposition instead of the BTF compression methods since it can expose the 2D positional semantic information, which can be subsequently used for BTF synthesis. To achieve effective and efficient BTF decomposition, we made a modification (we call it a Triple Plane) on the top of Biplane [Fan et al. 2023] that fully separates the positional and directional dimensions and leverages a lightweight MLP to recover the reflectance for one particular BTF.

2.3 Texture Synthesis

In many applications, there is a need to bind textures to large-scale surfaces, e.g., the entire surface of a mountain. However, textures have a limited size of representational scale that is not typically large, even for $4K$ textures. Directly tiling them will cause an undesirable repetitive pattern. Therefore, the texture synthesis methods are proposed to solve this problem.

Quilting and optimization based texture synthesis methods [Barnes et al. 2009; Efros and Freeman 2001; Efros and Leung 1999; Kaspar et al. 2015; Kwatra et al. 2005] commonly produce a plausible result but involve an offline process that finds the best-matched candidate patches (or texels) from the original texture by neighborhood searching or iterative optimization. It is like "growing" the texture from the synthesized area to the unsynthesized area region by region. Thus, they usually have difficulty supporting dynamic queries, and the new texture must be synthesized before use. As mentioned in Sec. 1, the dynamic property is more desirable and practical for BTF. Therefore, we don't overly focus on such methods. In this paper, we refer to any synthesis method, including Wang tiling [Wang 1961], that needs neighboring information for synthesis as a quilting-based method for simplicity.

In another aspect, dynamic by-example texture synthesis methods [Heitz and Neyret 2018; Mikkelsen 2022] are designed to generate a (commonly infinitely large) new texture from a small texture patch called "example" or "exemplar" on the fly. New texture at any place can be instantly queried without pre-generation. It typically involves a random selection step that selects some texture patches from the example texture and a blending step that blends

those example patches together. However, due to their fully dynamic design, they have difficulty achieving the same high quality as quilting-based methods.

Thanks to our dimensional decomposition design for the BTFs, in an orthogonal approach, we leverage the previous texture synthesis method to synthesize BTFs on the position domain. To achieve dynamic property, by-example texture synthesis methods are our preferred approach, although quilting-based methods may produce better synthesized BTFs. Note that both quilting-based and by-example synthesis have their own advantages and disadvantages. We do not aim to improve the previous texture synthesis work, but we expect to find some synthesis methods that have spatial variations and do not strictly repeat, from which our method can be further improved.

2.4 BTF synthesis

Many BTF synthesis methods have been developed in the past decades. Since BTF is a 6D function, people usually first compress its angular dimension and then perform synthesis. Previous BTF synthesis methods can mainly divided into two categories. The first is the quilting-based methods [Kawasaki et al. 2005; Koudelka et al. 2003; Lefebvre and Hoppe 2006; Liu et al. 2001; Ruiters et al. 2013; Steinhausen et al. 2015; Tong et al. 2002; Zhou et al. 2005], which rely on the texture quilting which we introduced in the Sec. 2.3. The second is tiling-based methods [Leung et al. 2007; Zhang et al. 2008] that leverage the idea of Wang Tiling [Wang 1961].

Apart from them, NeuBTF [Rodriguez-Pardo et al. 2023] is a neural method that emphasizes synthesizing BTF from a given guide map. Although it briefly mentions that it supports infinitely large BTF generation by tiling a seamlessly tileable BTF (pre-generated from a seamlessly tileable guide map), it still lags behind the dynamic synthesis scheme we proposed. Because the tiling-based approach brings repetitive patterns (even for seamlessly tileable BTF), and it's impossible for their method to generate a non-repetitive, infinitely large BTF due to their pre-generation design.

Those methods may produce spatially good results, but in order to support BTF synthesis, they rely on highly compressed BTF representation, e.g., PCA [Koudelka et al. 2003; Ruiters et al. 2013; Steinhausen et al. 2015; Zhang et al. 2008; Zhou et al. 2005] or Spherical Harmonics (SH) [Kawasaki et al. 2005; Lefebvre and Hoppe 2006; Leung et al. 2007] which are well-known to lose the high-frequency in angular domain. Moreover, none of these methods is *dynamic*, and it is impossible for them to generate an infinitely large texture on the fly.

3 METHOD

3.1 Problem Analysis

Our objective is to propose an effective scheme for *dynamic*, *infinitely large* BTF synthesis, where the BTF is generated only upon query. The immediate challenge is clear: BTFs are inherently 6D functions, thus occupying a large amount of storage. Together with the complexity of measuring accurate BTFs, it is usually difficult to acquire BTFs with high spatial resolutions. Nevertheless, high-resolution BTFs are required to realistically depict not only large-scale objects, such as an entire terrain, but also small objects with abundant detail, such as leather, cloth and wood. Therefore, synthesizing high-resolution BTFs is necessary.

The key to synthesizing BTFs, as we emphasize throughout this paper, is the property of being *dynamic*. Consider a toy example: given a small texture of resolution 512^2 to start with, how to find a texel's value at a very large coordinate, such as $(100000, 100000)$, on the synthesized texture. Those methods in Sec. 2.4, either by quilting patches or growing pixels, must compute from around the small example to that large coordinate, then answer this question. In rendering, such a query can happen per shader thread. Therefore, the entire BTF must be synthesized and stored prior to rendering, unless one can afford to repeat the same query-after-synthesis process for each thread, which is impossible in practice, no matter how fast previous methods perform. Therefore, it further distinguishes the concepts between dynamic and fast/real-time. Synthesis and query are already an issue for large textures. For BTFs, this issue is further magnified, since even with heavy compression (thus significant quality loss), large BTFs still suffer more than large textures. As a result, what we need is the ability to query an arbitrary location on a synthesized BTF but without actually synthesizing it beforehand, a.k.a., the property of being dynamic.

However, dynamic synthesis only existed in 2D textures so far. Therefore, our insight is to reduce the BTF synthesis process into texture synthesis tasks, so it can benefit from the advancement of dynamic synthesis. Following this idea, we propose our full scheme that decomposes a BTF into three disjoint 2D functions, synthesizes in the 2D position domain dynamically, and reconstructs the 6D BTF. Therefore, our method supports plugging in different methods for BTF compression (as long as 2D positional semantic information is exposed, see Sec. 4.3) and texture synthesis (even including those non-dynamic, e.g., [Efros and Freeman 2001]). We clarify that we do not intend to resolve these methods' own disadvantages, but we do evaluate combinations of representative methods comparatively to determine how they fit/benefit our general scheme, as shown in Fig. 9.

3.2 Theoretical Formulation

To achieve the aforementioned goal, we have set out to accomplish three tasks: constructing a faithful dimension decomposition method for BTF which decomposes the 6D BTF into a 2D positional function and two 2D angular functions; performing texture synthesis on the position domain based on an input spatial query; finally recovering the new 6D BTF from the synthesized function and the angular functions. Fig. 3 shows our scheme's overall pipeline.

BTF & Parameterization. A $\mathbf{BTF}(\cdot)$ is a 6D function representing the reflectance (usually a RGB value) of the surface with an input of 2D spatial coordinates $\mathbf{u} \in \mathbb{R}^2$ and a pair of 2D incident (viewing) and outgoing (lighting) directions ($\boldsymbol{\omega_i} \in \mathbb{R}^2$, $\boldsymbol{\omega_o} \in \mathbb{R}^2$):

$$\mathbf{BTF}(\mathbf{u}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}). \tag{1}$$

It can be considered as a texture at which each texel is storing a 4D BRDF. Comparing to incident and outgoing directions, it is better to reparameterized the BTFs with Rusinkiewicz parameterization [Rusinkiewicz 1998] which describes the angular dimension using the half vector $\mathbf{h} \in \mathbb{R}^2$ of ($\boldsymbol{\omega_i}$, $\boldsymbol{\omega_o}$) and a difference vector

$\mathbf{d} \in \mathbb{R}^2$ between the outgoing direction $\boldsymbol{\omega_o}$ and half vector $\mathbf{h}$:

$$\mathbf{BTF}(\mathbf{u}, \mathbf{h}, \mathbf{d}). \tag{2}$$

Dimension Decomposition. Due to the 6D nature of BTFs, storing even a small patch of BTF requires substantial memory space. Utilizing compression methods, e.g., PCA is essential in reducing runtime memory usage. Our target is to find a way to decompose the 6D BTF into a conceptual "outer product" ($\circ$) of three independent 2D functions $\{f^{(i)} | i \in (\mathbf{U}, \mathbf{H}, \mathbf{D})\}$ without losing details:

$$\mathbf{BTF}(\mathbf{u}, \mathbf{h}, \mathbf{d}) = f^{(\mathbf{U})}(\mathbf{u}) \circ f^{(\mathbf{H})}(\mathbf{h}) \circ f^{(\mathbf{D})}(\mathbf{d}). \tag{3}$$

However, finding such a purely mathematical solution is difficult. Inspired by the recent advances in dimension reduction methods, we turn to seek out a data-driven approach to achieve our goal. The "outer product" of those 2D functions is achieved by a neural operator $\mathcal{N}(\cdot)$:

$$\mathbf{BTF}(\mathbf{u}, \mathbf{h}, \mathbf{d}) = \mathcal{N}(f^{(\mathbf{U})}(\mathbf{u}), f^{(\mathbf{H})}(\mathbf{h}), f^{(\mathbf{D})}(\mathbf{d})). \tag{4}$$

Generally, dimensional decomposition methods like Tensor Decomposition (TD) aim to decompose the high-dimensional functions into 1Ds. But we are decomposing the 6D BTF into a series of 2D functions. It brings convenience in performing synthesis on the position domain, which will be introduced next.

By-example Synthesis. We noticed that the decomposed positional feature plane exhibits a highly semantic characteristic with detailed structures and variation information close to the BTF, as shown in Fig. 2. It gives us the idea to perform BTF synthesis directly on the positional domain via texture synthesis methods. There are numerous available texture synthesis methods, among them, the by-example texture synthesis methods fit our goal best, which allows dynamic query and synthesis without pre-generation. Specifically, with a target query position ($\mathbf{u}^* \in \mathbb{R}^2$, typically from an infinitely large planar domain), the by-example texture synthesis process mainly involves two steps: finding multiple example patches from the example texture f corresponding to the target query; blending those example patches while keeping some statistic properties (e.g., histogram [Heitz and Neyret 2018]) of the example texture to obtain the synthesized texture f^*. Generally, the texture synthesis method can be formulated as follows:

$$f^*(\mathbf{u}^*) = \mathbf{Syn}(f, \mathbf{u}^*). \tag{5}$$

The synthesis function $\mathbf{Syn}(\cdot)$ only corresponds to the example texture f and the target query $\mathbf{u}^*$. Notably, it does not rely on the local neighborhood of the target query, eliminating the need for 'voting' from previously synthesized areas. This approach differs from quilting-based methods, which are not dynamic; however, it does not prevent the use of such methods in our framework. We assume the positional feature plane can be treated as a 2D texture:

$$f^{*(\mathbf{U})}(\mathbf{u}^*) = \mathbf{Syn}(f^{(\mathbf{U})}, \mathbf{u}^*). \tag{6}$$

BTF Recovery. After first decomposing 6D BTF into a series of 2D functions and performing synthesis in the position domain, the next thing is to recover the 6D function from the 2D functions. As illustrated in Eqn. 4. We train a lightweight MLP as the neural operator to obtain the reflectance of the synthesized BTF:

$$\mathbf{BTF}^*(\mathbf{u}^*, \mathbf{h}, \mathbf{d}) = \mathcal{N}(f^{*(\mathbf{U})}(\mathbf{u}^*), f^{(\mathbf{H})}(\mathbf{h}), f^{(\mathbf{D})}(\mathbf{d})). \tag{7}$$

Moreover, a general model is unnecessary in our case, so one MLP and the corresponding feature planes are only responsible for one specific BTF.

3.3 Implementation

BTF Decomposition. As described in Sec. 3, one of our goals is to decompose 6D BTF function into an outer product of three 2D functions. As our method is designed to be general to neural dimensional decomposition methods, we modified Biplane [Fan et al. 2023] with a 4-layer lightweight MLP and fully decomposed 2D functions $\{f^{(i)} | i \in (\mathbf{U}, \mathbf{H}, \mathbf{D})\}$. Those 2D functions are obtained by initializing three 2D discrete feature planes with learnable parameters. The specific values in the planes are progressively learned during training. This idea is inspired by Biplane [Fan et al. 2023], but we decompose 6D BTF completely into three 2D functions, and we call it a *Triple Plane*.

In practice, given a 6D query of $(\mathbf{u}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$, we first convert it onto Rusinkiewicz coordinates system [Rusinkiewicz 1998]: $(\mathbf{u}, \mathbf{h}, \mathbf{d})$. Then, we use them as texture coordinates to bilinearly fetch features from each corresponding plane.

By-example BTF Synthesis. As visualized in Fig. 2, the decomposed positional plane shows a highly semantic characteristic with detailed structures and variation information close to the original BTF. We found that directly extending the texture-based by-example synthesis methods to our positional feature "texture", i.e., feature plane $f^{(\mathbf{U})}$ produces a plausible result. Our by-example BTF synthesis scheme shows strong compatibility with different texture synthesis methods, which we will demonstrate in Sec. 4. The specific synthesis method used can be very flexible. We emphasize the histogram-preserving blending [Heitz and Neyret 2018] since we found our method can also benefit from preserving the histogram of the feature plane, though the feature plane is in a latent space.

BTF Recovery. Finally, to obtain the BTF reflectance from the decomposed feature planes, we employ a lightweight MLP $\mathcal{N}$ with 4 layers of 32 hidden nodes (except the last outputting layer):

$$\mathbf{BTF}(\mathbf{u}, \mathbf{h}, \mathbf{d}) = \mathcal{N}(f^{(\mathbf{U})}(\mathbf{u}), f^{(\mathbf{H})}(\mathbf{h}), f^{(\mathbf{D})}(\mathbf{d})), \tag{8}$$

where $f^{(\mathbf{U})}$ can be the synthesized $f^{*(\mathbf{U})}$. For simplicity, we do not strictly distinguish between these two terms. The feature planes and the lightweight MLP are jointly trained with a simple ℓ_1 loss:

$$\mathcal{L} = \ell_1(\mathcal{N}(f^{(\mathbf{U})}(\mathbf{u}), f^{(\mathbf{H})}(\mathbf{h}), f^{(\mathbf{D})}(\mathbf{d})), \mathbf{BTF}(\mathbf{u}, \mathbf{h}, \mathbf{d})). \tag{9}$$

The feature planes and the MLP are jointly optimized during training. The BTF synthesis is not performed during training. We expect our method to support synthesis automatically, even without training constraints to ensure the orthogonality of feature planes.

For more detailed implementation information, please refer to supplementary material.

4 RESULTS

Our proposed method introduces a general BTF synthesis scheme that allows flexibility in selecting each component, including neural

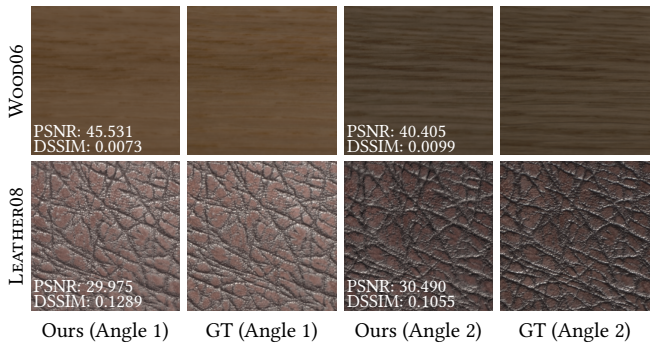

Fig. 4. Validation of our method's representation capability for different BTFs. We visualize our results along with ground truth (GT) BTFs with two different pairs of directions for each different material. We show the PSNR↑ and DSSIM↓ at the bottom-left corner.

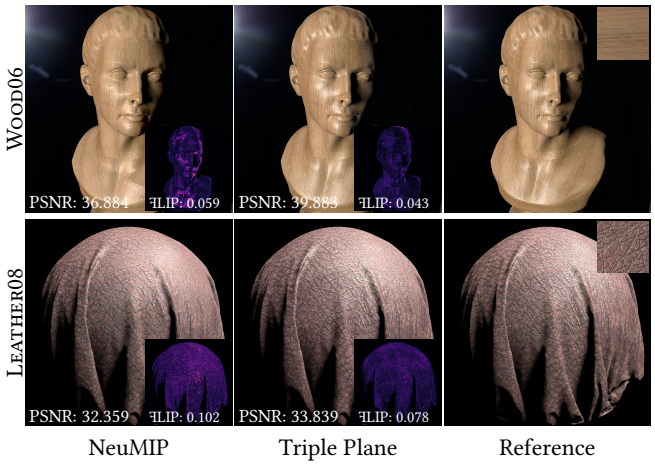

Fig. 5. Comparison with NeuMIP [Kuznetsov et al. 2021] and reference without applying synthesis, i.e., use repetitive tiling. The reference image is generated by interpolating the original BTF data. We show the PSNR↑, ꟻLIP error↓ and the error image at the bottom. Triple Plane is closer to the reference with more accurate highlights and patterns, but NeuMIP is also good. Therefore, both can be used in our scheme.

Fig. 6. We validate our by-example BTF synthesis in different scales by scaling the UV coordinate, the scaling factor is marked on the bottom-right corner. Even on a very large scale (45×), our method faithfully maintains the accurate appearance of the BTF, which demonstrates our capability of generating an infinitely large, non-repetitive BTF.

BTF representation and texture synthesis. To illustrate this versatility, we first validate each component independently by comparing various methods. Subsequently, we integrate these components to assess the overall effectiveness of our dynamic BTF synthesis approach. For the neural BTF representation, we implement a modified NeuMIP [Kuznetsov et al. 2021] without the offset module and only use the finest layer of the feature pyramid to demonstrate the advance of our Triple Plane. Note that the NeuMIP's complexity is not reduced, and keeping the finest feature layer also won't have a negative impact on its results since there is no LoD involved. As for the texture synthesis, we compare histogram-preserving blending [Heitz and Neyret 2018], another dynamic by-example synthesis method Hex-Tiling [Mikkelsen 2022] and a non-dynamic texture quilting [Efros and Freeman 2001]. Unless specifically stated, the term "ours" refers to the Triple Plane with histogram-preserving blending (if performing synthesis).

4.1 Validation and Comparison of BTF Representation

In Fig. 4, we visualize our BTF recovery results along with ground truth BTFs with two different pairs of directions for each material. As shown in the figure, our method faithfully captures the complex patterns and the highly specular reflection.

In Fig. 5, we compare our Triple Plane with a modified NeuMIP [Kuznetsov et al. 2021] using repetitive tiling, i.e., no synthesis is performed. The reference image is generated by interpolating the original BTF data. Our Triple Plane shows more accurate highlights in Wood06 and clearer stretch patterns in Leather08 compared to NeuMIP. We believe the quality improvement is because of the novel dimension decomposition scheme which has a clear separation of the position and direction dimensions.

4.2 Validation and Comparison of Texture Synthesis

Our approach treats the 2D positional feature plane similarly to standard 2D textures, enabling dynamic synthesis. We employ histogram preserving blending [Heitz and Neyret 2018] as our primary solution. In Fig. 6, we first demonstrate our method's capability of dynamically synthesizing a non-repetitive BTF on an arbitrary scale. Even on a very large scale (45× UV scaling), our method faithfully maintains the accurate appearance of the BTFs.

Our method can also benefit from other texture synthesis methods. To demonstrate that, we employ two additional texture synthesis methods. The first one is Hex-Tiling [Mikkelsen 2022], another dynamic by-example texture synthesis approach, and the second one is a non-dynamic texture quilting technique [Efros and Freeman 2001], a classic method in patch-based quilting. As shown in Fig. 7, our synthesis scheme supports all these synthesis methods, and quilting produces the most visually pleasing result because it finds the best match patches by an offline search. However, it requires a pre-generation of the synthesized BTF. The synthesized BTF is with 15× UV scaling (equivalent to 6K resolution). Even if the quilting BTF has already been decomposed into 2D functions, storing the neural textures still takes about 2GB. The histogram-preserving blending and Hex-Tiling provide a balance between dynamic query and structured quality (further discussion in Sec. 5).

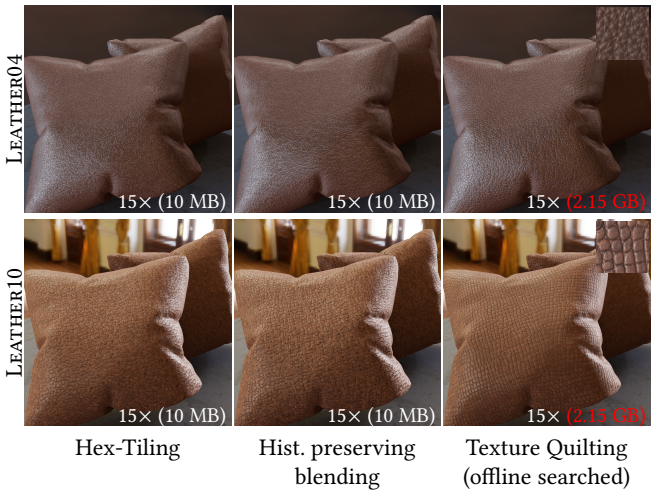

Fig. 7. Our BTF synthesis scheme is a general idea that is compatible with different texture synthesis methods. To demonstrate that, we implement another dynamic by-example texture synthesis approach Hex-Tiling [Mikkelsen 2022] (left), and a non-dynamic texture quilting [Efros and Freeman 2001] (right). With 15× UV scaling, the synthesized BTF is equivalent to having a 6K resolution. Quilting produces the most visually pleasing result because it finds the best match patches by offline searching. However, once generated, the texture can not be changed, and even after our dimensional decomposition, the synthesized texture still takes GBs storage.

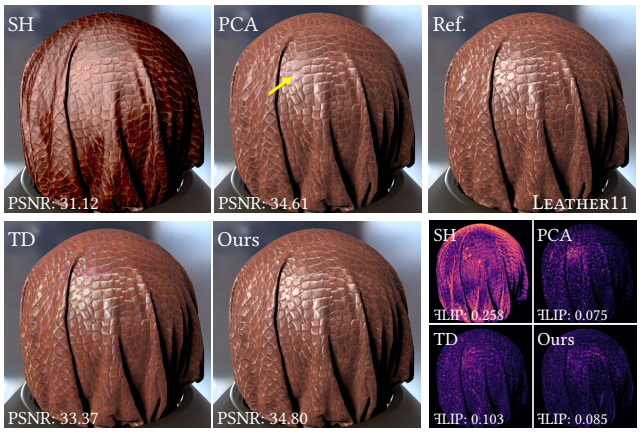

Fig. 8. Comparison with classical BTF compression methods: SH [Kawasaki et al. 2005], PCA [Koudelka et al. 2003] and Tensor Decomposition (TD) [Ruiters and Klein 2009]. To have a relatively fair comparison, we employ a similar number of coefficients for each method. For SH, we compute a discrete table of different view directions for each BTF textel. Each item in the table stores 27 SH coefficients (3 levels SH for each RGB channel) that fit the corresponding fixed view 2D BRDF. For PCA, we keep 15 eigenvalues (5 eigenvalues for each RGB channel). For TD, we basically follow the original paper's [Ruiters and Klein 2009] setting, i.e., $k_1 = k_2 = 13$ and $D_1 = D_2 = 256$, but we separately perform the decomposition on each RGB channel. The ꟻLIP error image is shown at the bottom-right corner. Our Triple Plane faithfully captures the appearance and the structure of the BTF, while SH, even only representing 2D angular distributions, loses most of the high-frequency signals. PCA fails to preserve sharp highlights. TD has visually better highlights than PCA, but the overall error is larger.

4.3 Comparison of BTF Synthesis

The existing non-neural-based BTF synthesis methods described in Sec. 2.4 are all quilting-based. Comparing with them is *unfair* to ours since we prioritize the *dynamic* property, whereas these established methods are *non-dynamic*. Non-dynamic methods may excel in structure preservation, but generating very large BTFs is impractical, as demonstrated in Fig.7.

Most existing BTF synthesis methods are derived from the quilting method, which we use for comparison in Fig. 7. These quilting methods essentially rearrange the texels without changing the original content of the BTF (i.e., no blending is performed). As our method employs a general scheme, it can utilize the same quilting approach for benchmarking, with differences only in representation capabilities—for instance, our Triple Plane versus PCA/TD/SH. Thus, in combination with Fig. 7, the comparisons of our Triple Plane with PCA, TD, and SH in Fig. 8 effectively reflect comparisons with earlier BTF synthesis techniques.

Moreover, as analyzed in Sec. 1, many of the neural-based appearance synthesis works aim at estimating 2D parametric maps, primarily reconstructing or estimating BTFs from a few images, which diverges from our objectives. Additionally, some methods (e.g., [Zhou et al. 2023]), generate partial, 4D isotropic BTFs. These are inherently different from our approach and are not suitable for direct comparison with our full 6D BTF synthesis scheme.

Based on the aforementioned considerations, we believe that the most appropriate solution is to compare with NeuMIP [Kuznetsov et al. 2021], with and without using synthesis (repetitive tiling). As shown in Fig. 9. Our Triple Plane (second and fourth columns) shows great effectiveness in both representation and synthesis. NeuMIP (first and third columns), however, notably lacks high-frequency details, resulting in unsharp highlights (Stone04) and blurred reflections (Leather03). This is possible because our dimensional decomposition method completely separates the positional dimensions from the directional dimensions, whereas NeuMIP requires angular information as a condition for input, which might result in the directional information being implicitly mixed and stored with the positional features.

It is very important to emphasize that our BTF synthesis method is the first one enabling dynamic by-example synthesis on BTFs. Therefore, there is no ground truth, or rather, the ground truth should be by-example synthesizing directly on 6D BTFs and without compression. However, this is impossible since the high-dimensional extension of the by-example texture synthesis method does not exist, and it needs exponentially increasing numbers of example patches. Even if we only perform blending in the 2D positional domain, the remaining 4D angular dimensions may contain sharp lobes that blend to ghosting artifacts unless leveraging costly methods such as optimal transport [Bonneel et al. 2011].

4.4 Performance

The evaluation time of our method via naive in-shader matrix multiplication takes 2.0 ms for 2,073,600 times queries (1920 × 1080 resolution) on an RTX 4090 GPU. It includes both neural texture

fetching cost and MLP inference cost with the overall cost scaling linearly with the number of evaluations. While the texture synthesis takes less than 0.2 ms under the same condition. Currently, our MLP inference utilizes straightforward matrix multiplication in fp32 precision without specialized optimizations. Despite this, our method achieves the reported interactive performance.

5 DISCUSSION & LIMITATIONS

There are certain limitations to our scheme. As analyzed in Sec. 3, our BTF synthesis scheme allows plug-and-play components for both representation and synthesis while we inherit the same limitations. We expect our method can directly benefit from a better choice of each component.

Representation Capability. As shown in Fig. 4 and Fig. 5, our Triple Plane produces slightly blurry results. It demonstrates that the BTF with a complex appearance will be a challenge to our method. Furthermore, without constraints on each plane to enforce their orthogonality during training, the resulting decomposition might not be very clean. A better-designed decomposition approach may help solve this problem.

Structure-persevering Synthesis. As shown in the second row of Fig. 7, with dynamic by-example texture synthesis (histogram-preserving blending [Heitz and Neyret 2018] and Hex-Tiling [Mikkelsen 2022]), one can not handle a highly structured BTF since the used texture synthesis methods can not. We hope to find a dynamic synthesis method that has spatial variations and does not strictly repeat, e.g., dynamic Wang Tiling [Wang et al. 2020] or another by-example synthesis method that better preserves the structure.

6 CONCLUSION & FUTURE WORK

In this paper, we have presented a by-example BTF synthesis scheme, allowing the dynamic synthesis of an infinitely large non-repetitive BTF from a small example BTF. We first introduce a novel dimension decomposition method (Triple Plane) that decomposes the high-dimensional 6D BTF into a series of 2D functions (including a 2D positional function and two 2D angular functions). Then, we design a simple but effective scheme that performs by-example texture synthesis on the decomposed 2D positional function as if it is a 2D texture. Finally, a lightweight MLP is employed to recover the synthesized BTF reflectance. Our results faithfully preserved the accurate appearance of the synthesized BTFs, and our method is robust in various types of BTFs.

A specifically designed importance sampling strategy is also in high demand, and one potential extension could be integrating normalizing flow [Xu et al. 2023] or histograms [Xu et al. 2023; Zhu et al. 2021]. Furthermore, in the future, it might be possible to explore a novel synthesis strategy that supports blending on structured patterns, which could further enhance our method. In addition, we anticipate that runtime performance could be greatly enhanced through targeted optimizations such as batching inference, quantization, or utilizing hardware acceleration (e.g., tensor cores).

ACKNOWLEDGMENTS

We thank Alexandr Kuznetsov and Chen Liu for their discussions and insights on the initial ideas.

REFERENCES

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* 28, 3 (2009), 24.

Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. *ACM Trans. Graph.* 30, 6 (dec 2011), 1–12. https://doi.org/10.1145/2070781.2024192

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.

Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)* 18, 1 (1999), 1–34.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–15.

Alexei A. Efros and William T. Freeman. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 341–346. https://doi.org/10.1145/383259.383296

Alexei A Efros and Thomas K Leung. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, Vol. 2. IEEE, 1033–1038.

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

Jiahui Fan, Beibei Wang, Miloš Hašan, Jian Yang, and Ling-Qi Yan. 2023. Neural Biplane Representation for BTF Rendering and Acquisition. In *Proceedings of SIGGRAPH 2023*.

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 Trans. Graph.* 38, 4 (2019), 134–1.

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 Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020. MaterialGAN: Reflectance Capture Using a Generative SVBRDF Model. *ACM Trans. Graph.* 39, 6, Article 254 (nov 2020), 13 pages. https://doi.org/10.1145/3414685.3417779

Michael Guthe, Gero Müller, Martin Schneider, and Reinhard Klein. 2009. BTF-CIELab: A perceptual difference measure for quality assessment and compression of BTFs. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 101–113.

Eric Heitz and Fabrice Neyret. 2018. High-performance by-example noise using a histogram-preserving blending operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 1–25.

Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. 2021. Generative modelling of BRDF textures from flash images. *arXiv preprint arXiv:2102.11861* (2021).

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.

Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. 2015. Self tuning texture optimization. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 349–359.

Hiroshi Kawasaki, Kyoung-Dae Seo, Yutaka Ohsawa, and Ryo Furukawa. 2005. Patch-based BTF synthesis for real-time rendering. In *IEEE International Conference on Image Processing 2005*, Vol. 1. IEEE, I–393.

Melissa L Koudelka, Sebastian Magda, Peter N Belhumeur, and David J Kriegman. 2003. Acquisition, compression, and synthesis of bidirectional texture functions. In *3rd International Workshop on Texture Analysis and Synthesis (Texture 2003)*, Vol. 59. 64.

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2021. NeuMIP: Multi-Resolution Neural Materials. *ACM Trans. Graph.* 40, 4, Article 175 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459795

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

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*. 795–802.

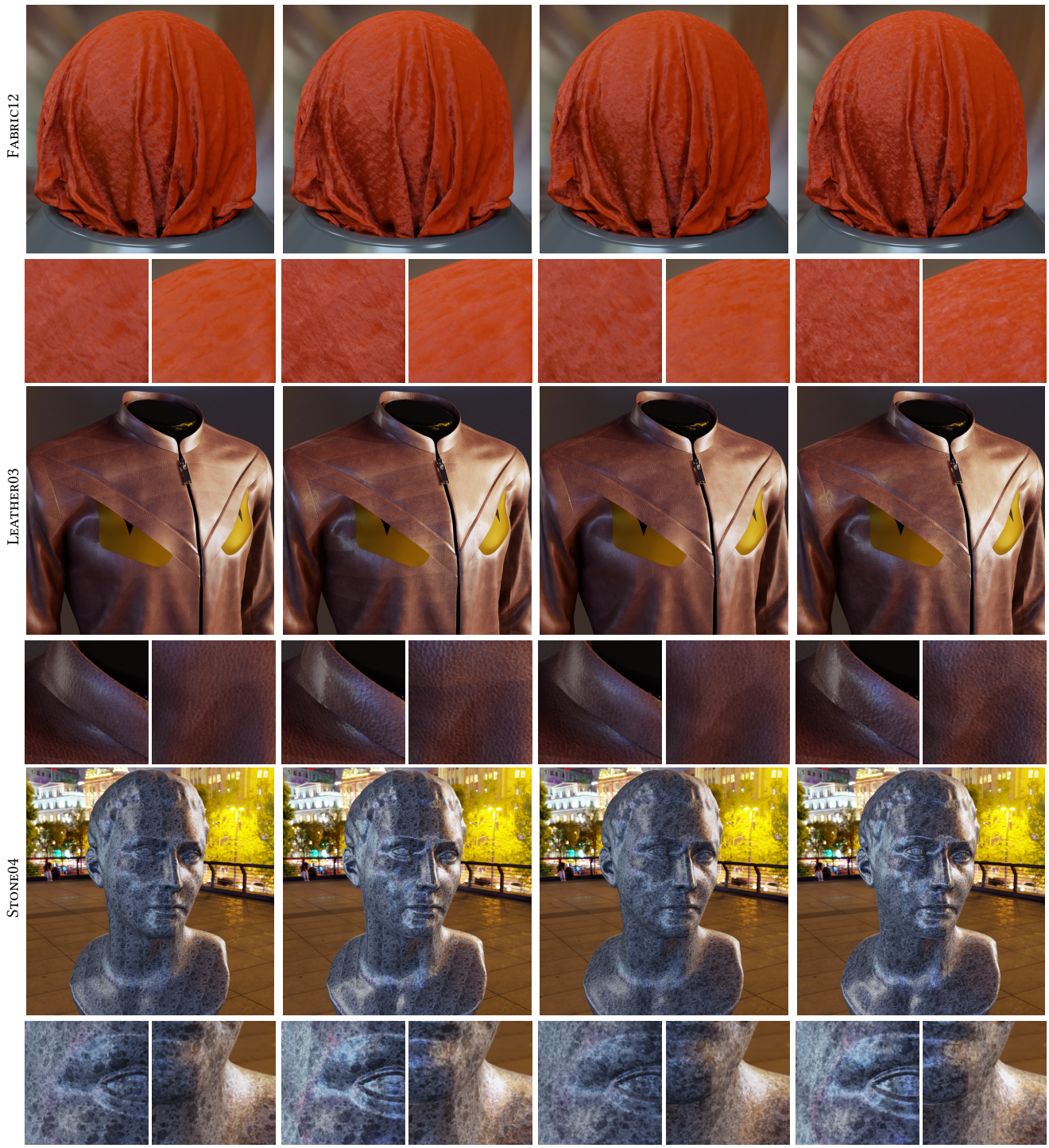

Fig. 9. We implement NeuMIP with histogram preserving blending (third column) to demonstrate our BTF synthesis scheme is general to any BTF dimension reduction method with an accessible positional feature plane. Our Triple Plane (second and fourth columns) shows great effectiveness in both representation and synthesis. NeuMIP, however, notably lacks high-frequency details, resulting in unsharp highlights (Stone04) and blurred reflections (Leather03).

Sylvain Lefebvre and Hugues Hoppe. 2006. Appearance-space texture synthesis. *ACM Trans. Graph.* 25, 3 (jul 2006), 541–548. https://doi.org/10.1145/1141911.1141921

Man-Kang Leung, Wai-Man Pang, Chi-Wing Fu, Tien-Tsin Wong, and Pheng-Ann Heng. 2007. Tileable BTF. *IEEE transactions on visualization and computer graphics* 13 (09 2007), 953–65. https://doi.org/10.1109/TVCG.2007.1034

Xinguo Liu, Yizhou Yu, and Heung-Yeung Shum. 2001. Synthesizing Bidirectional Texture Functions for Real-World Surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 97–106. https://doi.org/10.1145/383259.383269

Morten S Mikkelsen. 2022. Practical Real-Time Hex-Tiling. *Journal of Computer Graphics Techniques Vol* 11, 2 (2022).

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. 2020. Unified Neural Encoding of BTFs. *Computer Graphics Forum (Proc. Eurographics)* 39, 2 (13 July 2020), 167–178. https://doi.org/10.1111/cgf.13921

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).

Carlos Rodriguez-Pardo, Konstantinos Kazatzis, Jorge Lopez-Moreno, and Elena Garces. 2023. NeuBTF: Neural fields for BTF encoding and transfer. *Computers & Graphics* 114 (2023), 239–246.

Roland Ruiters and Reinhard Klein. 2009. BTF Compression via Sparse Tensor Decomposition. *Computer Graphics Forum* 28, 4 (2009), 1181–1188. https://doi.org/10.1111/j.1467-8659.2009.01495.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01495.x

Roland Ruiters, Martin Rump, and Reinhard Klein. 2009. Parallelized Matrix Factorization for fast BTF Compression.. In *EGPGV@ Eurographics*. 25–32.

Roland Ruiters, Christopher Schwartz, and Reinhard Klein. 2013. Example-based Interpolation and Synthesis of Bidirectional Texture Functions. *Computer Graphics Forum* 32, 2pt3 (2013), 361–370. https://doi.org/10.1111/cgf.12056 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12056

Szymon M. Rusinkiewicz. 1998. A New Change of Variables for Efficient BRDF Representation. In *Rendering Techniques '98*, George Drettakis and Nelson Max (Eds.). Springer Vienna, Vienna, 11–22.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: Differentiable Material Graphs for Procedural Material Capture. *ACM Trans. Graph.* 39, 6 (Dec. 2020), 1–15.

Heinz C Steinhausen, Rodrigo Martín, Dennis den Brok, Matthias B Hullin, and Reinhard Klein. 2015. Extrapolation of bidirectional texture functions using texture synthesis guided by photometric normals. In *Measuring, Modeling, and Reproducing Material Appearance 2015*, Vol. 9398. SPIE, 81–94.

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural brdf representation and importance sampling. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 332–346.

Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. 2002. Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces. *ACM Trans. Graph.* 21, 3 (jul 2002), 665–672. https://doi.org/10.1145/566654.566634

M Alex O Vasilescu and Demetri Terzopoulos. 2004. TensorTextures: Multilinear image-based rendering. In *ACM SIGGRAPH 2004 Papers*. 336–342.

Beibei Wang, Miloš Hašan, Nicolas Holzschuch, and Ling-Qi Yan. 2020. Example-based microstructure rendering with constant storage. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–12.

Hao Wang. 1961. Proving theorems by pattern recognition—II. *Bell system technical journal* 40, 1 (1961), 1–41.

Michael Weinmann, Juergen Gall, and Reinhard Klein. 2014. Material Classification Based on Training Data Synthesized Using a BTF Database. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 156–171.

Bing Xu, Liwen Wu, Milos Hasan, Fujun Luan, Iliyan Georgiev, Zexiang Xu, and Ravi Ramamoorthi. 2023. NeuSample: Importance Sampling for Neural Materials. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 41, 10 pages. https://doi.org/10.1145/3588432.3591524

Bowen Xue, Shuang Zhao, Henrik Wann Jensen, and Zahra Montazeri. 2024. A Hierarchical Architecture for Neural Materials. arXiv:2307.10135 [cs.GR]

Zhan Zhang, Yue Qi, and Yong Hu. 2008. Fast Synthesis and Rendering of BTF on Arbitrary Surfaces. In *11th Joint International Conference on Information Sciences*. Atlantis Press, 370–375.

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.

Kun Zhou, Peng Du, Lifeng Wang, Yasuyuki Matsushita, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. 2005. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 519–528.

Xilong Zhou, Milos Hasan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2022. Tilegen: Tileable, controllable material generation and capture. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.

Xilong Zhou, Miloš Hašan, Valentin Deschaintre, Paul Guerrero, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2023. PhotoMat: A Material Generator Learned from Single Flash Photos. In *SIGGRAPH 2023 Conference Papers*.

Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Miloš Hašan, and Ling-Qi Yan. 2021. Neural Complex Luminaires: Representation and Rendering. *ACM Trans. Graph.* 40, 4, Article 57 (jul 2021), 12 pages. https://doi.org/10.1145/3450626.3459798