Computers & Graphics 114 (2023) 239–246

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on CEIG 2023

NeuBTF: Neural fields for BTF encoding and transfer

Carlos Rodriguez-Pardo [a,b,*], Konstantinos Kazatzis [a], Jorge Lopez-Moreno [a,b,*], Elena Garces [a,b,*]

[a] SEDDI, Spain
[b] Universidad Rey Juan Carlos, Spain

ARTICLE INFO

Article history:
Received 21 May 2023
Accepted 14 June 2023
Available online 20 June 2023

Keywords:
Neural fields
Reflectance
Rendering
BTF compression

ABSTRACT

Neural material representations are becoming a popular way to represent materials for rendering. They are more expressive than analytic models and occupy less memory than tabulated BTFs. However, existing neural materials are immutable, meaning that their output for a certain query of UVs, camera, and light vector is fixed once they are trained. While this is practical when there is no need to edit the material, it can become very limiting when the fragment of the material used for training is too small or not tileable, which frequently happens when the material has been captured with a gonioreflectometer. In this paper, we propose a novel neural material representation which jointly tackles the problems of BTF compression, tiling, and extrapolation. At test time, our method uses a guidance image as input to condition the neural BTF to the structural features of this input image. Then, the neural BTF can be queried as a regular BTF using UVs, camera, and light vectors. Every component in our framework is purposefully designed to maximize BTF encoding quality at minimal parameter count and computational complexity, achieving competitive compression rates compared with previous work. We demonstrate the results of our method on a variety of synthetic and captured materials, showing its generality and capacity to learn to represent many optical properties.

1. Introduction

A common approach to modeling real-world spatially-varying materials in computer graphics is through the use of Bidirectional Texture Functions (BTFs). This type of representation models the dense optical response of the material, and is more general than analytic representations such as microfacet SVBRDF. However, BTFs can occupy large amounts of memory. Recently, neural material representations are being proposed as a learning-based alternative to tabulated BTFs, providing a more compact solution while keeping the flexibility and generality of BTFs.

Creating digital representations of real material samples requires using an optical capture device, such as a gonioreflectometer, a smartphone [1], or a flatbed scanner [2]. During the process, several choices must be made. First, it is important to select a patch of the material that contains enough spatial variability. Second, a process – automatic or manual – must be found to produce a tileable material that can be used to create seamless 3D renders. Finally, resources must be allocated for storage as needed. Making these choices when dealing with implicit or tabulated representations, such as in BTFs or neural materials, is particularly crucial. Once these representations are trained or captured, they cannot be easily modified and it is only possible to query them using the UVs, light, and camera vectors.

In this paper, we propose a novel neural material representation that addresses these issues. Unlike existing neural approaches that are immutable once trained [3–6], our model can be queried at test time with a guidance image that conditions the neural BTF to the structure provided by the guidance image. Our approach resembles synthesis by example and procedural processes, and can be used to extrapolate BTFs to large material samples, as well as to easily create tileable ones. Furthermore, our method achieves better compression rates than previous work on neural BTF representations.

To achieve this, we present a novel method that works at two steps. In the first step, we condition the neural BTF using a *guidance image* as input. To this end, we use an *autoencoder* that outputs a high-dimensional latent representation of the material, a *neural texture*, which jointly encodes reflectance and structural properties. In the second step, the UV position of the latent representation, along with the camera and light vectors, are decoded by a fully-convolutional sinusoidal decoder, a neural *renderer* to obtain the RGB values. Using a single BTF as input, we train the network end-to-end using a custom training procedure, loss function, and data augmentation policy. This policy, inspired by recent

* Corresponding authors.
E-mail addresses: carlos.rodriguezpardo.jimenez@gmail.com (C. Rodriguez-Pardo), konstantinos.kazatzis@seddi.com (K. Kazatzis), jorge@jorg3.com (J. Lopez-Moreno), elenagarces@gmail.com (E. Garces).

https://doi.org/10.1016/j.cag.2023.06.018
0097-8493/

work on attribute transfer [7], allows the autoencoder to encode the relationship between structural features and reflectance, enabling the propagation of the BTF to novel input guidances. Once trained, the novel input guidances may come from the same material, a different material, or a structural pattern. An input guidance of the same material can be used to extrapolate the BTF to larger samples or create tileable BTFs, provided the input guidance is tileable. If the input guidance is a structural pattern, the local features can be used to synthesize novel materials.

In summary, we propose the following contributions:

- The first neural BTF representation with conditional input that can be used to extrapolate BTF measurements, easily create tileable BTFs, and synthesize novel materials.
- We show how to leverage our system for rendering large-scale and tileable neural BTF generation using measurements captured with small portions of the material.
- We demonstrate that our method works with synthetic and captured materials of diverse optical properties, including colored specular or anisotropy.

We provide additional results, supplementary materials, and implementation details at our project website.

2. Related work

An accurate method for representing the optical properties of materials is through Bidirectional Texture Functions (BTFs) [8]. BTFs are 6D functions that characterize all possible combinations of incoming and outgoing light and camera directions for the 2D spatial extent of a material. Although they are successful in representing materials, they have a major drawback in terms of memory requirements. Therefore, BTF compression has been a major research topic [9]. Non-neural approaches used dimensionality reduction techniques such as Principal Component Analysis (PCA) [10–12], vector quantization [13], or clustering [14]. However, these approaches were recently surpassed by neural models [15] due to their flexibility and superior capacity to learn non-linear functions.

Neural BTFs. Rainer et al. [3] proposed the first method to use deep autoencoders to compress BTFs, surpassing PCA [12] on captured BTFs. However, this approach required training a single neural network per material. To address this limitation, a later work by the same authors [5] proposed a generalization of this idea in which a single network was able to generalize to a variety of materials. Although these methods were very effective for compressing flat materials, they had some limitations when it came to modeling materials with volume. In their work, Kuznetsov et al. [4] improved the quality of neural materials by introducing a neural offset module that captures parallax effects. Further, they method also allowed for level-of-detail though MIP mapping by training a multi-resolution neural representation. However, grazing angles and silhouette effects remained a challenge for this approach. In a subsequent work, Kuznetsov et al. [6] explicitly trained the network using queries that span surface curvatures, effectively handling these cases. Representing fur, fabrics, and grass with neural reflectance fields was explored by Baatz et al. [16] who proposed a representation that jointly models reflectance and geometry.

All of these approaches share the idea of querying neural material using UVs, camera, and lighting vectors, but do not provide any functionality for modifying the material once the network is trained. In contrast, our approach can take a guidance image as input, which conditions the output to generate material variability.

Material synthesis and tiling. Texture synthesis is a long-standing problem in the field of computer graphics. The goal is to reconstruct a larger image given a small sample, leveraging the structural content and internal statistics of the input image. This concept has been used for synthesizing single images, BTFs, and full material models. For images, the most common strategies include PatchMatch [17], texture transport [18], point processes [19,20], or neural networks [21–24]. BTF synthesis, however, has received less attention. Steinhausen et al. [25,26] extrapolated BTF captures to larger material samples using non-neural texture synthesis methods. For full materials, Li et al. [27] captured the appearance of materials by first estimating their BRDF and then synthesizing the high-resolution micro-structure from a dataset of measured SVBRDFs. Nagano et al. [28] measured microscopic patches of the skin and used a convolutional filter to propagate the measurements to a spatially-varying texture. Deschaintre et al. [29] used an autoencoder to propagate SVBRDFs to large material samples. Also recently, Rodriguez-Pardo and Garces [7] propagated any kind of visual attribute having a single image as guidance. Their approach shares some similarity to ours, although they transfer 2D image attributes, while we transfer the full BTF.

Procedural models [30–32] are nowadays very successful for generating tileable materials. Thanks to the use of a tileable template, these methods adjust the generated image to the features available in the template. As we show, our approach can also work with a binary template as input. However, guaranteeing predictable outputs given this kind of input is out of the scope of our technique, which can transfer BTF measurements having as input a guidance image of the same material.

Other neural representations in rendering. Limited to BRDFs, neural networks trained with adaptive angular sampling have been explored to enable importance sampling [33], needed for Monte Carlo integration. Deep latent representations also allow for BRDF editions. For example, Hu et al. [34] demonstrate that autoencoders can outperform classic PCA for the purpose of editing. Other applications of neural encodings in rendering are numerous. For instance, they have been used for scene prefiltering [35], where geometry and materials are simplified to accommodate the LoD of the scene using a voxel-based representation and trained latent encodings. For anisotropic microfacets, Gauthier et al. [36] propose a cascaded architecture able to adjust the material parameters to the MIP mapping level. Encoding light transport using neural networks for real-time global illumination has also been explored [37,38], showcasing promising results.

3. Method

We present an overview of our approach in Fig. 1, where we show our inference and training pipelines. Our goal is twofold: First, find a compact representation for a BTF through the use of neural networks. Second, enable the extrapolation of the BTF according to guidance images used as input. In Section 3.1 we describe our inference pipeline and neural network, and in Section 3.2 our training process. Section 4 contains specific implementation details and design of the neural networks.

3.1. Inference

Our neural network is composed of three modules: an *autoencoder* $\mathcal{A}$, a *neural texture* $\mathcal{T} \in \mathbb{R}^{H\times W\times D}$, and a *renderer* $\mathcal{R}$. The renderer $\mathcal{R}(\mathcal{T}(\mathrm{u},\mathrm{v}),\omega_o,\omega_i) = \mathrm{RGB}$ takes as input the feature vector at the (u,v) coordinates of the neural texture $\mathcal{T}$, the view ω_o and light ω_i positions, and returns an RGB value. $\mathcal{R}$ acts as a conventional BTF and can be used as such in any render engine.

An input *guidance* image, $\mathcal{G} \in \mathbb{R}^{H\times W\times 3}$, is used during inference to condition the generation of the *neural texture* $\mathcal{T}$. This

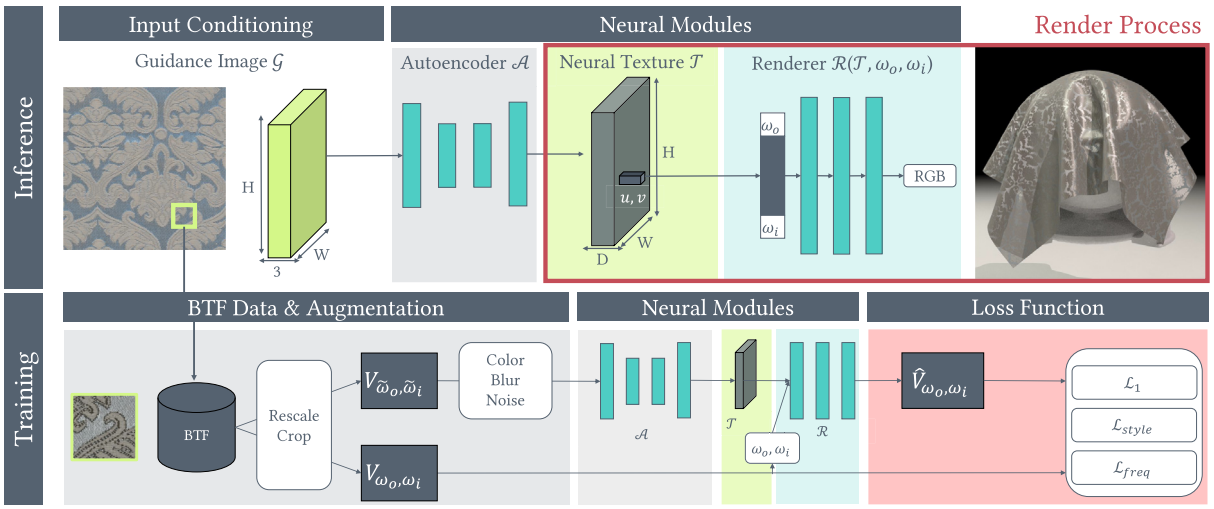

Fig. 1. An overview of our neural BTF inference and training processes. Top-Inference: Using a guidance image $\mathcal{G} \in \mathbb{R}^{H\times W\times 3}$, we use our trained autoencoder to generate the Neural Texture $\mathcal{A}(\mathcal{G}) = \mathcal{T}_\mathcal{G} \in \mathbb{R}^{H\times W\times D}$, which preserves the spatial resolution of the input image but represents a higher-dimensional learned representation. This Neural Texture $\mathcal{T}_\mathcal{G}$, along with the trained renderer $\mathcal{R}$, can be queried as a regular BTF, using UVs, and target camera and light positions for regular rendering. Bottom-Train: During training, following previous work on photometric data augmentation [7], we randomly select an input *view* $V_{\tilde{\omega}_o,\tilde{\omega}_i}$ and a target V_{ω_o,ω_i}. This allows the model to generalize to novel light or camera conditions and acts as a regularizer. To both views, we apply random rescale and cropping. Then, only to $V_{\tilde{\omega}_o,\tilde{\omega}_i}$, we randomly apply hue variations, gaussian blur, and noise, and feed it to the *autoencoder*, which returns a 2D *latent representation* of the material. A fully-convolutional *decoder* with sinusoidal activations receives both this latent space and the target ω_o, ω_i camera and light angles, and estimates $\hat{V}_{\omega_o,\omega_i}$. This output is compared with V_{ω_o,ω_i} using a multifaceted loss function.

conditioning allows us to propagate the learned reflectance to novel guidance images that can be: a larger sample of the same material, a different material, or a structural image. In the simpler case, the guidance image comes from the BTF used for training, and our process is equivalent to previous work [3,4].

The autoencoder $\mathcal{A}$ takes the guidance image $\mathcal{G} \in \mathbb{R}^{H\times W\times 3}$ as input and outputs a neural texture $\mathcal{T} \in \mathbb{R}^{H\times W\times D}$ with the same size $H \times W$ as the input guidance image but with more latent dimensions D. As a result, each pixel in the guidance image has a higher-dimensional neural representation in $\mathcal{T}$. Because it is trained without explicit supervision, this latent representation can capture the reflectance and structural patterns automatically. Kuznetsov et al. [4] also used a latent neural representation of the material, however, lacking the initial autoencoder their approach cannot synthesize novel BTFs without retraining, while our conditioning module allow us to generate novel BTFs during test time. The autoencoder and the renderer are neural networks trained jointly, using an end-to-end image-to-image approach describe below.

3.2. Training

Fig. 1 (bottom) illustrates our training process. It has two objectives: First, equivalent to regular BTF encoding, we aim to find the mapping between camera direction ω_o, light direction ω_i, and output slices of the BTF: $(\omega_o, \omega_i) \rightarrow V_{\omega_o,\omega_i} \in$ BTF. Second, we aim to condition the synthesis process with an input guidance image, $\mathcal{G}$. To this end, for training, we feed the model with images, $V_{\tilde{\omega}_o,\tilde{\omega}_i}$, which are randomly sampled from the BTF, and are subject to additional data augmentation processes. This extensive augmentation process guarantees invariance to different input variations during test time, like camera or illumination conditions, while keeps consistency of the outputs.

Loss function design. Our loss function, that compares ground truth slices V_{ω_o,ω_i} with generated ones $\mathcal{R}(\mathcal{A}(f(V_{\tilde{\omega}_o,\tilde{\omega}_i})), \omega_o, \omega_i)$, is a weighted sum of three terms: a pixel-wise loss, a style loss, and a frequency loss,

$$\mathcal{L} = \lambda_{\mathcal{L}_1}\mathcal{L}_1 + \lambda_{style}\mathcal{L}_{style} + \lambda_{freq}\mathcal{L}_{freq} \quad (1)$$

The main driver of our loss is the pixel-wise norm $\mathcal{L}_1$. $\mathcal{L}_1$ produces sharper results than higher-order alternatives, such as $\mathcal{L}_2$ [7,39]. Following [4], we apply a $log(x + 1)$ compression to improve the model results on high dynamic range. This compression is only done to the pixel-wise component of the loss function. Inspired by recent work on texture synthesis, capture and transfer [2,24, 32,40,41], we introduce a $\mathcal{L}_{style}$ loss to help the model generate higher quality and sharper results. Further, to mitigate the spectral bias of convolutional neural networks and help ameliorate the results further, we also introduce a *focal frequency loss* into our learning framework [42]. This combination of loss functions proves effective for our problem, without the need for complex adversarial losses which could reduce efficiency or destabilize training.

Data augmentation. We train our models using a comprehensive data augmentation policy aimed at achieving high quality reflectance propagation, increasing performance and generalization, and allowing for generation of multiple resolution materials at test time. We build upon recent work on material transfer [7] and use images of the material taken under different illumination and viewing conditions as inputs to our autoencoder. This helps it generalize to novel capture setups, which allows for multiple applications we describe on Section 6. In particular, we use every image available on the input BTF, selected uniformly at random for each element in each batch during training. As in [7], we also use random rescaling, which helps the model generalize to new scales, and build neural materials of multiple resolutions at test time, as we describe on Section 6.3. Inspired by recent work on image synthesis [7,24,43], we use random cropping, which helps generalization by effectively increasing the dataset size. Finally, we extend the color augmentation policy in [7] with random hue changes across the entire color wheel, and introduce random Gaussian noise and blurs to the input images, to help it generalize further, as proposed in [2].

4. Model design and implementation

We provide extensive implementation details for model sizes, training and data generation on the supplementary material.

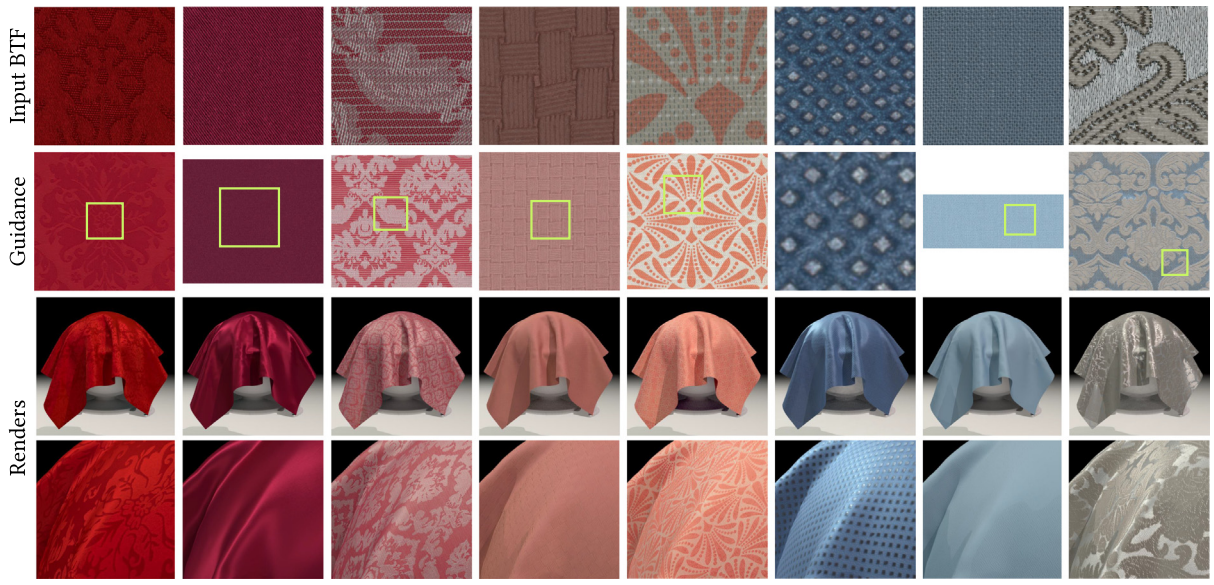

Fig. 2. Some tileable neural materials achieved with our method. On the top row, we show a slice of the BTF used to train a NeuBTF representation. With the tileable guidance images shown on the second row, we propagate the neural texture using our autoencoders. These neural textures can be rendered to generate realistic images (third). We provide closeups on the bottom row. In the cases where the guidance image covers a larger area than the training crop, we highlight the training surface area as a green inset.

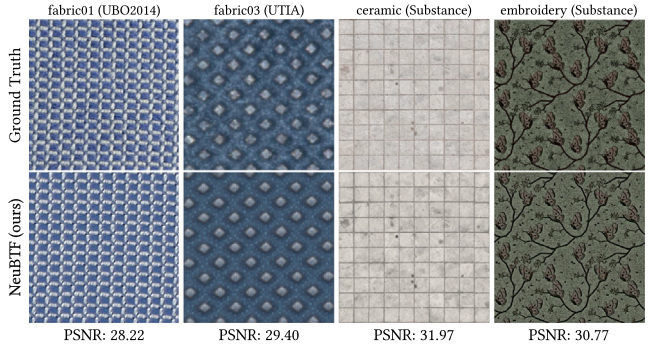

Fig. 3. Qualitative results on a variety of BTFs, from different sources. From left to right, we show results on a material from the UBO [12] BTF dataset, the UTIA [54] BTF dataset and two synthetic materials rendered from Substance SVBRDFs.

Autoencoder. For the *autoencoder*, we use a lightweight U-Net [44] with a few modifications to tailor it for our problem. Inspired by recent work on CNN design, we leverage ConvNext [45] Blocks across our model, with depth-wise convolutions using 5×5 kernels. We empirically observe that ConvNext blocks achieve higher quality structural editions at a lower parameter count than vanilla U-Net blocks. To further help convergence and preserve details in the input images, we use residual connections [2,46,47] in every convolutional block of the model. We use 1×1 convolutions on the skip connections and residual scaling [48]. As in [45], we use Layer Normalization [49] and GELU non-linearities [50]. On the bottleneck of the model, we introduce an attention module [51] to help the model learn longer-range dependencies. To avoid checkerboard artifacts [52], we use nearest neighbors interpolation for upsampling. Inspired by recent work on tileable material generation [32], we use *circular padding* throughout the model. We initialize its weights using *orthogonal initialization* [53], which helps avoiding exploding gradients.

Renderer. For the *renderer*, we build upon SIREN [55] MLPs, with additional modifications to enhance its performance for our problem. We use 1×1 convolutions instead of vanilla linear layers, to allow for end-to-end training using 2D images. Further, we introduce Layer Normalization [49] before each sinusoidal non-linearity, which stabilizes training. Finally, inspired by [56], we use residual connections [46], to help preserve the information of the input vector across the decoder layers. Model weight initialization follows [55]. With sinusoidal activations, we observe significantly higher reconstruction quality and training dynamics than with ReLU [57] MLPs, which are common for BTF compression [3–5]. Because the network is fully-convolutional, it can take as input feature vectors of any size. This is very convenient for our use cases when the input guidance image have a size different from the size of the original BTF used to train it. The renderer can be evaluated very efficiently in GPU, at an average of $2.514e^{-4} \pm 4.48e^{-5}$ ms per sample.

5. Evaluation

5.1. Qualitative and quantitative analysis

We evaluate our method on materials from different sources including acquired BTFs from [12,54], and rendered BTFs from procedurally generated and scanned SVBRDFs. In Fig. 2, we show examples of the results of NeuBTF for a variety of materials with highly complex structures and reflectance properties, like colored specular (first column) or anisotropy (last). We show some additional results in Fig. 3 for materials of different datasets. As shown, our model achieves high quality reconstructions regardless on the type of data source. In Table 1, we show the reconstruction error for the same materials, averaged across the full directional space, for a variety of pixel-wise and perceptual metrics.

Finally, in Fig. 4, we show a colored visualization for a few channels of the latent neural texture $\mathcal{T}$ found for a variety of materials. Because the values for the neural texture are unbounded, to each channel $c \in \mathcal{T}$, we standardize them to 0 mean and unit variance, and apply a $sigmoid(c) = \frac{1}{e^{1-c}+1}$ non-linearity to make the maps comparable. Without any explicit training, the models learn to separate distinct parts of the material. For example, the model finds distinct latent spaces for *warp* and *weft* yarns on woven fabrics, or separation between color and

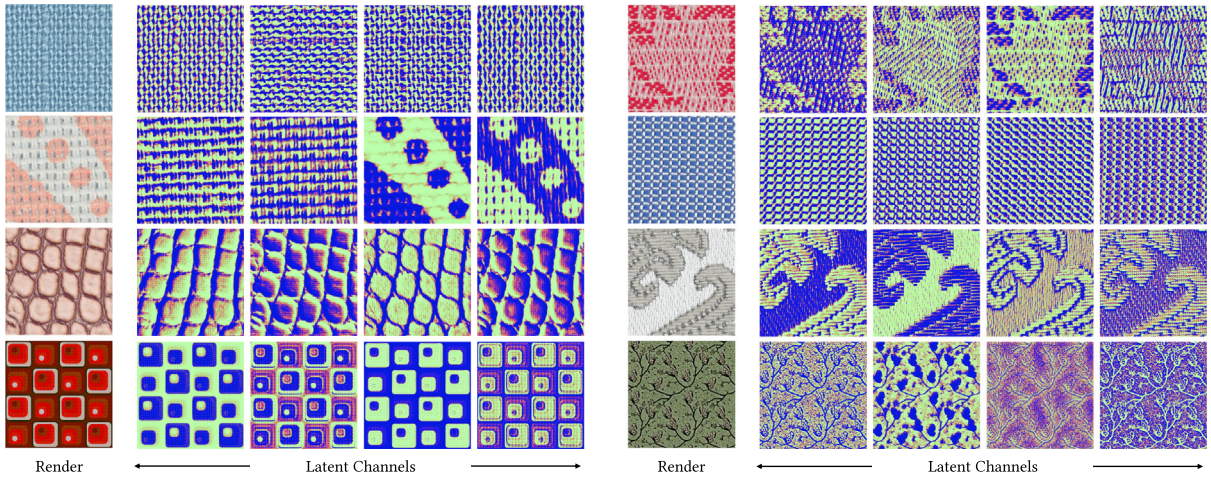

Fig. 4. A selection of latent channels learned by NeuBTF for a variety of materials. We use a colorspace to help visualization. Without explicit supervision, the model internally learns semantically meaningful latent spaces. For instance, in the second example on the left, the two leftmost latent spaces encode geometry, while the other two encode the two distinct colors of the printed pattern over the yarns.

Table 1
Average (± std.) reconstruction error across the full dimensional space for materials of different datasets, measured using pixel-wise and perceptual metrics.

Material	fabric01 [12]	fabric03 [54]	ceramic (*Subst.*)	embroidery (*Subst.*)
PSNR ↑	26.58±1.820	27.73±4.711	29.19±2.111	28.79±1.831
SSIM [58] ↑	0.710±0.108	0.729±0.142	0.819±0.110	0.652±0.156
LPIPS [59] ↓	0.451±0.041	0.404±0.054	0.270±0.075	0.341±0.135
FLIP [60] ↓	0.391±0.047	0.426±0.105	0.365±0.065	0.391±0.115

Table 2
Number of trainable parameters in the decoders and amount of latent texture channels for different neural BTF compression algorithms. We use Torchinfo [61] for this analysis. Note that exact comparisons are challenging, as [4] optimizes a multi-level texture pyramid and [5] learns a latent vector which can encode novel materials. For neither our method nor [3,5], we count the parameters in the encoders, as they are not needed for using the materials on rendering systems.

Method	**NeuBTF (ours)**	NeuMIP [4]	Rainer 2019 [3]	Rainer 2020 [5]
Decoder Parameters	**3011**	3332	35725	38269
Texture Channels	**14**	**14**	**14**	38

geometric patterns. This disentanglement provides clues on why the material propagation is possible, and suggests potential future research directions for fine-grained neural material edition.

5.2. Compression comparisons with previous work

In Table 2, we show the number of trainable parameters on the decoders of different neural BTF compression algorithms. As shown, our model is competitive with previous work in terms of trainable parameters. This is achieved as we use more complex loss functions than previous work, which help regularize the models, and because our sinusoidal MLP achieves higher quality reconstructions for natural signals than ReLU MLPs, as shown in [55]. NeuMIP [4] uses smaller MLPs, however, they require an additional decoder for their *neural offset* module, which helps them encode parallax effects (See Fig. 5), for which our model struggles. Our decoder has one order of magnitude fewer parameters than [3,5], however, the method in [5] provides the benefit of fast encoding of new materials, while ours requires a different model for each new material.

5.3. Limitations

As we show in Fig. 5, our model struggles with materials with strong displacement. While our method provides accurate encodings on viewing angles close to the material surface, it cannot accurately encode grazing angles for such extreme cases. Displacement maps translate the geometric position of the points over the surface, breaking the underlying assumptions behind our neural texture. NeuMIP [4] solves this issue by explicitly modeling parallax effects with a *neural offset* module. While we did not observe that such extension was needed for acquired BTF data, like the UBO2014 [62] dataset, introducing a similar module into our editable neural material framework is an interesting future research direction to increase its generality.

6. Applications

6.1. Reflectance propagation and tileable neural BTFs

Many material reflectance acquisition devices are limited in the surface dimensions they can digitize. This hinders their applicability to many real-world materials, which exhibit variations that cannot be captured at such small scales. Further, in many applications like SVBRDF acquisition, obtaining larger samples of the material improves realism and helps tileable texture synthesis. In this context, previous work on BTF reflectance compression inherit the surface area limitations of the capture devices used to generate their training data. Our method can easily be applied for reflectance propagation. We build upon the work of Rodriguez-Pardo and Garces [7] and leverage our *encoder* to propagate the neural texture optimized using a small portion of the material

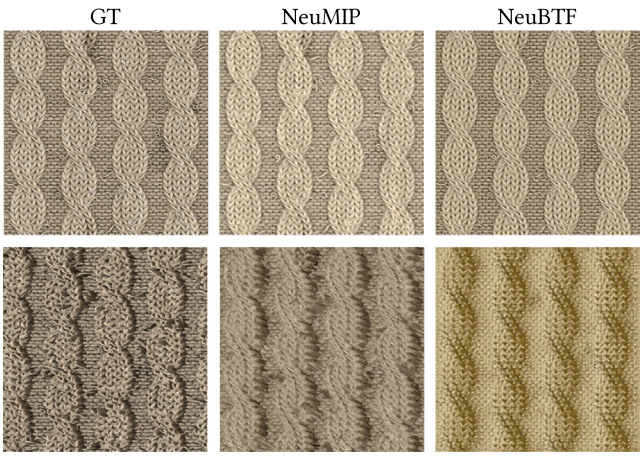

Fig. 5. A failure case of our method. Compared to NeuMIP [4], which explicitly models parallax effects, our model struggles to accurately encode materials with strong displacements, as this synthetic cable knit from Substance3D. For this type of materials, NeuBTF accurately encodes orthogonal viewing angles (top row), however, it struggles at grazing angles (bottom row).

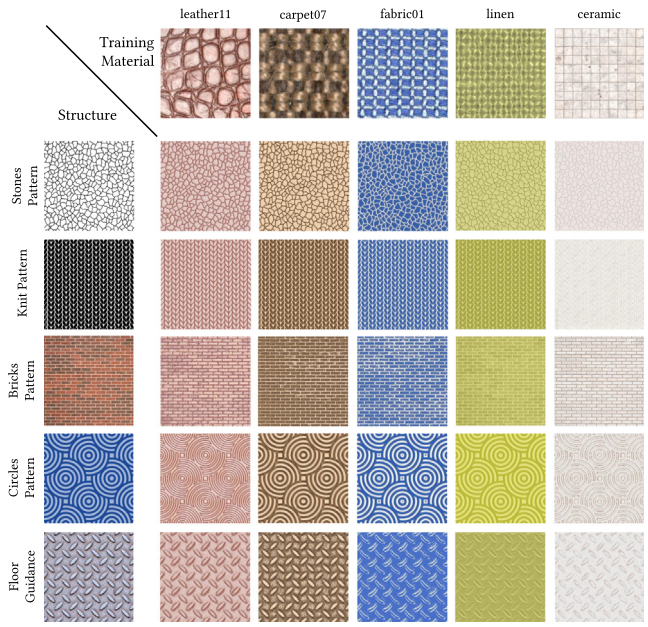

Fig. 6. Examples of structural editions allowed by our method on a variety of materials. On the leftmost column, we show *guidance images*, which represent structures into which we transfer the neural BTF measurements illustrated on the top row. As shown, our method can effectively propagate BTF measurements into many material and structure types, using as guidances either synthetic or real images. The first three materials (*leather11, carpet07, fabric01*) are taken from *UBO 2014* [62], the *linen* material is rendered from a captured SVBRDF, while the *ceramic* material is rendered from an artistic material taken from *Substance3D*. We show renders generated using $\theta_v = \theta_l = 0$. Additional results are provided on the supplementary material.

(e.g. a 1×1 cm capture) to a larger portion of the same material, represented with a *guidance image* captured using a commodity device like a flatbed scanner. Because our model is trained using a large amount of lighting conditions, as in [7], the propagation is invariant to how the images are illuminated. We show results of such pipeline in Fig. 2. For instance, on the last row, we show an anisotropic and specular SILVER JACQUARD fabric, for which we generated a BTF by rendering a 1×1 cm SVBRDF. This small crop cannot represent the complex pattern in the fabric, which we show on the *guidance image*, which covers a 10×10 cm area. Using our encoder, we propagate the neural material to this guidance image, generating a new, high-resolution, latent space which we can render, enabling realistic material representations with a reduced digitization cost. This propagated neural material has a 2000×2000 texels resolution, and it requires no re-training during test time.

Relatedly, our propagation framework can also be easily leveraged for generating tileable BTFs. Given any *guidance image* of the material, we can generate a tileable version of it, either using manual editions by artists or automatic algorithms [24,63–65]. With this tileable input guidance, we can use our autoencoder $\mathcal{A}$ to propagate the neural texture $\mathcal{T}$, effectively generating tileable BTFs, as we show in Fig. 2. This propagation algorithm can leverage state-of-the-art algorithms for tileable texture synthesis without any modification of our material model or training framework. Tileable BTFs were not achievable with previous approximations and this simple pipeline has the potential of enabling novel applications of this type of material representation in rendering scenarios.

6.2. Structural material edition

Besides propagating BTF measurements to larger portions of the material, NeuBTF allows for generating novel materials using structural editions. Given a trained NeuBTF and a guidance image representing some particular target structure, we can propagate the neural texture to this guidance image, generating high-quality neural materials which preserve the structure of the guidance image and the reflectance properties of the trained neural material. This pipeline allows for easily generating multiple different neural materials without the need for retraining. As we show in Fig. 6, this propagation method works for many types of input guidances, including vector black and white images, procedurally generated textures, or real photographs of materials and textures. We show results on acquired BTFs and from synthetic BTFs, rendered from scanned and manually generated SVBRDFs. As shown, our propagation frameworks provides high-quality material editions, even for very challenging cases, like the *circles pattern*.

6.3. Multi resolution neural materials

Another useful application enabled by our method is the generation of materials at different resolutions. Unlike previous work [4], which explicitly optimizes a pyramid of levels of detail during training, we can generate materials at any resolution at test time without introducing any additional complexities to our material representation. Because we train our models using random rescales as a data augmentation policy, they are equivariant to rescales of its input guidance images $G \downarrow$: $\mathcal{R}(\mathcal{A}(\mathcal{G})) \downarrow = \mathcal{R}(\mathcal{A}(\mathcal{G} \downarrow))$. As such, we can generate any continuous resolution for a particular BTF by downsampling the guidance image to the target resolution and propagating its neural texture, as we illustrate in Fig. 7. Note that this algorithm only guarantees accurate results for the rescaling ranges that we use during data augmentation.

7. Conclusions

We have presented a learning based representation for material reflectance which provides efficient encoding and powerful propagation capabilities. Our method introduces input conditioning into neural BTF representations. This allows for multiple applications which were not possible with previous neural

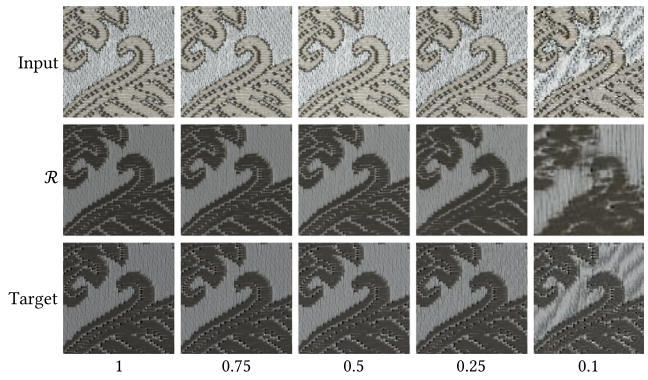

Fig. 7. Our method naturally enables for the generation of different resolutions for the neural materials. We show the input to the autoencoder (top row), the rendered material at $\theta_c = 75, \theta_l = 60, \phi_c = \phi_l = 0$ (middle row), and the ground truth image at those positions (bottom), at different resolutions (columns). We achieve this by downsampling the guidance image $\mathcal{G}$ fed into the autoencoder module, which returns an accurately downsampled latent space, thanks to our data augmentation policy applied during training. The rightmost column lies beyond the ranges in which we train the model, however, the results are still somewhat plausible.

models, including BTF extrapolation, tiling and novel material synthesis through structure propagation. Our method builds upon recent work on neural fields, network design and data augmentation, showing competitive compression capabilities with previous work on neural BTF representation. Through multiple analyses, we have shown the capabilities of our method on a variety of materials with different reflectance properties, including anisotropy or specularity, as well as effectively handling either synthetic and acquired BTFs.

Our method can be extended in several ways. The most immediate extension is to allow for materials with strong parallax effects due to displacement mapping or curvature, as in [4,6]. Our representation is limited to opaque materials. Extending them to handle translucent or holed surfaces would increase their realism in materials like thin fabrics or meshes. Further, our method could be extended to allow for hyperspectral BTF data [66], but captured data is scarce. Besides, recent work on neural BRDF representations [33,37,67] and generative models [68] suggests a promising research direction: Learning to sample from neural BTFs, using invertible neural networks. While these may introduce challenging complexities to the models, they could provide efficient representations for Monte Carlo rendering using importance sampling. Further, building upon recent work on SVBRDF capture [2,32], BRDF sampling [69] and BTF compression [5], it could be possible to learn a prior over neural BTFs with a generative model. This should help in capturing more efficiently the data needed for generating these assets, as well as generating new materials and interpolating between them. Finally, editing semantic and reflectance properties in neural fields is an active area of research [70–74]. While our method introduces structural edition into neural BTF representations, it is not capable of editing particular semantic properties, such as albedo or specularity. Extending our edition capabilities to more fine-grained parameters is an interesting research avenue. We hope our method inspires future research on neural material representations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

Elena Garces was partially supported by a Juan de la Cierva - Incorporacion Fellowship (IJC2020-044192-I). This publication is part of the project TaiLOR, CPP2021-008842 funded by MCIN/AEI/10.13039/501100011033 and the NextGenerationEU/PRTR programs.

References

[1] Guo Y, Smith C, Hašan M, Sunkavalli K, Zhao S. Materialgan: Reflectance capture using a generative svbrdf model. 2020, arXiv preprint arXiv:201000114.
[2] Rodriguez-Pardo C, Dominguez-Elvira H, Pascual-Hernandez D, Garces E. Umat: Uncertainty-aware single image high resolution material capture. In: Proc of computer vision and pattern recognition. 2023.
[3] Rainer G, Jakob W, Ghosh A, Weyrich T. Neural btf compression and interpolation. In: Computer graphics forum, vol. 38, Wiley Online Library; 2019, p. 235–44.
[4] Kuznetsov A. Neumip: Multi-resolution neural materials. ACM Trans Graph 2021;40(4).
[5] Rainer G, Ghosh A, Jakob W, Weyrich T. Unified neural encoding of btfs. In: Computer graphics forum, vol. 39, Wiley Online Library;; 2020, p. 167–78.
[6] Kuznetsov A, Wang X, Mullia K, Luan F, Xu Z, Hasan M, et al. Rendering neural materials on curved surfaces. In: ACM SIGGRAPH 2022 conference proceedings. 2022, p. 1–9.
[7] Rodriguez-Pardo C, Garces E. Neural photometry-guided visual attribute transfer. IEEE Trans Vis Comput Graphics 2021;29(3):1818–30.
[8] Dana KJ. Brdf/btf measurement device. In: Proceedings eighth IEEE international conference on computer vision. Vol. 2. IEEE; 2001, p. 460–6.
[9] Filip J, Haindl M. Bidirectional texture function modeling: A state of the art survey. IEEE Trans Pattern Anal Mach Intell 2008;31(11):1921–40.
[10] Koudelka ML, Magda S, Belhumeur PN, Kriegman DJ. Acquisition, compression, and synthesis of bidirectional texture functions. In: 3rd International workshop on texture analysis and synthesis. 2003, p. 59–64.
[11] Müller G, Meseth J, Klein R. Compression and real-time rendering of measured btfs using local pca. In: VMV. 2003, p. 271–9.
[12] Weinmann M, Gall J, Klein R. Material classification based on training data synthesized using a btf database. In: European conference in computer vision. Springer; 2014, p. 156–71.
[13] Havran V, Filip J, Myszkowski K. Bidirectional texture function compression based on multi-level vector quantization. In: Computer graphics forum, vol. 29, Wiley Online Library; 2010, p. 175–90.
[14] Tong X, Zhang J, Liu L, Wang X, Guo B, Shum HY. Synthesis of bidirectional texture functions on arbitrary surfaces. ACM Trans Graph 2002;21(3):665–72.
[15] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
[16] Baatz H, Granskog J, Papas M, Rousselle F, Novák J. Nerf-tex: Neural reflectance field textures. In: Computer graphics forum. vol. 41, Wiley Online Library; 2022, p. 287–301.
[17] Diamanti O, Barnes C, Paris S, Shechtman E, Sorkine-Hornung O. Synthesis of complex image appearance from limited exemplars. ACM Trans Graph 2015;34(2):1–14.
[18] Aittala M, Weyrich T, Lehtinen J. Two-shot svbrdf capture for stationary materials. ACM Trans Graph 2015;34(4):1–13.
[19] Guehl P, Allègre R, Dischler JM, Benes B, Galin E. Semi-procedural textures using point process texture basis functions. In: Computer graphics forum, vol. 39, Wiley Online Library; 2020, p. 159–71.
[20] Lefebvre S, Hoppe H. Appearance-space texture synthesis. ACM Trans Graph 2006;25(3):541–8.
[21] Elad M, Milanfar P. Style transfer via texture synthesis. IEEE Trans Image Process 2017;26(5):2338–51.
[22] Zhou Y, Zhu Z, Bai X, Lischinski D, Cohen-Or D, Huang H. Non-stationary texture synthesis by adversarial expansion. ACM Trans Graph 2018;37(4):1–13.
[23] Frühstück A, Alhashim I, Wonka P. Tilegan: Synthesis of large-scale non-homogeneous textures. ACM Trans Graph 2019;38(4):1–11.
[24] Rodriguez-Pardo C, Garces E. Seamlessgan: Self-supervised synthesis of tileable texture maps. IEEE Trans Vis Comput Graphics 2022.
[25] Steinhausen HC, den Brok D, Hullin MB, Klein R. Extrapolating large-scale material btfs under cross-device constraints. In: Bommes D, Ritschel T, Schultz T, editors. Vision, modeling & visualization. The Eurographics Association; 2015, p. 143–50.

[26] Steinhausen HC, Martín R, den Brok D, Hullin MB, Klein R. Extrapolation of bidirectional texture functions using texture synthesis guided by photometric normals. In: measuring, modeling, and reproducing material appearance II (SPIE 9398), vol. 9398, 2015.
[27] Lin Y, Peers P, Ghosh A. On-site example-based material appearance acquisition. In: Computer graphics forum. vol. 38, Wiley Online Library; 2019, p. 15–25.
[28] Nagano K, Fyffe G, Alexander O, Barbic J, Li H, Ghosh A, et al. Skin microstructure deformation with displacement map convolution. ACM Trans Graph 2015;34(4). 109-1, https://dl.acm.org/doi/10.1145/2766894.
[29] Deschaintre V, Drettakis G, Bousseau A. Guided fine-tuning for large-scale material transfer. In: Computer graphics forum, vol. 39, Wiley Online Library; 2020, p. 91–105.
[30] Hu Y, Hašan M, Guerrero P, Rushmeier H, Deschaintre V. Controlling material appearance by examples. In: Computer graphics forum, vol. 41, Wiley Online Library; 2022, p. 117–28.
[31] Zhou X, Hašan M, Deschaintre V, Guerrero P, Sunkavalli K, Kalantari NK. A semi-procedural convolutional material prior. In: Computer graphics forum, Wiley Online Library; 2023.
[32] Zhou X, Hasan M, Deschaintre V, Guerrero P, Sunkavalli K, N.K. Kalantari. Tilegen: Tileable, controllable material generation and capture. In: SIGGRAPH Asia 2022 conference papers. 2022, p. 1–9.
[33] Sztrajman A, Rainer G, Ritschel T, Weyrich T. Neural brdf representation and importance sampling. In: Computer graphics forum, vol. 40, Wiley Online Library; 2021, p. 332–46.
[34] Hu B, Guo J, Chen Y, Li M, Guo Y. Deepbrdf: A deep representation for manipulating measured brdf. In: Computer graphics forum, vol. 39, Wiley Online Library; 2020, p. 157–66.
[35] Bako S, Sen P, Kaplanyan A. Deep appearance prefiltering. ACM Trans Graph 2023;42(2):1–23.
[36] Gauthier A, Faury R, Levallois J, Thonat T, Thiery JM, Boubekeur T. Mipnet: Neural normal-to-anisotropic-roughness mip mapping. ACM Trans Graph 2022;41(6):1–12.
[37] Rainer G, Bousseau A, Ritschel T, Drettakis G. Neural precomputed radiance transfer. In: Computer graphics forum, vol. 41, Wiley Online Library; 2022, p. 365–78.
[38] Gao D, Mu H, Xu K. Neural global illumination: Interactive indirect illumination prediction under dynamic area lights. IEEE Trans Vis Comput Graphics 2022.
[39] Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 1125–34.
[40] Mardani M, Liu G, Dundar A, Liu S, Tao A, Catanzaro B. Neural ffts for universal texture image synthesis. Adv Neural Inf Process Syst 2020;33:14081–92.
[41] Aittala M, Weyrich T, Lehtinen J. Practical svbrdf capture in the frequency domain. ACM Trans Graphics (ToG) 2013;32(4):110–1.
[42] Jiang L, Dai B, Wu W, Loy CC. Focal frequency loss for image reconstruction and synthesis. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 13919–29.
[43] Texler O, Futschik D, Kučera M, Jamriška O, Sochorová Šárka, Chai M, et al. Interactive video stylization using few-shot patch-based training. ACM Trans Graph 2020;39(4):73.
[44] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9 2015, proceedings, Part III 18. Springer; 2015, p. 234–41.
[45] Liu Z, Mao H, Wu CY, Feichtenhofer C, Darrell T, Xie S. A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 11976–86.
[46] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.
[47] Diakogiannis FI, Waldner F, Caccetta P, Wu C. Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. ISPRS J Photogramm Remote Sens 2020;162:94–114.
[48] Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 8110–9.
[49] Ba JL, Kiros JR, Hinton GE. Layer normalization. 2016, arXiv preprint arXiv:160706450.
[50] Hendrycks D, Gimpel K. Gaussian error linear units (gelus). 2016, arXiv preprint arXiv:160608415.
[51] Woo S, Park J, Lee JY, Kweon IS. Cbam: Convolutional block attention module. In: Proceedings of the European conference on computer vision. 2018, p. 3–19.
[52] Odena A, Dumoulin V, Olah C. Deconvolution and checkerboard artifacts. Distill 2016.
[53] Arpit D, Campos V, Bengio Y. How to initialize your network? Robust initialization for weightnorm & resnets. Adv Neural Inf Process Syst 2019;32.
[54] Haindl M, Filip J VR. Digital material appearance: The curse of tera-bytes. ERCIM News 2012;(90):49–50.
[55] Sitzmann V, Martel J, Bergman A, Lindell D, Wetzstein G. Implicit neural representations with periodic activation functions. Adv Neural Inf Process Syst 2020;33:7462–73.
[56] Mehta I, Gharbi M, Barnes C, Shechtman E, Ramamoorthi R, Chandraker M. Modulated periodic activations for generalizable local functional representations. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 14214–23.
[57] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning. 2010, p. 807–14.
[58] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: From error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.
[59] Zhang R, Isola P, Efros AA, Shechtman E, Wang O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 586–95.
[60] Andersson P, Nilsson J, Akenine-Möller T, Oskarsson M, Åström K, Fairchild MD. Flip: A difference evaluator for alternating images. Proc ACM Comput Graph Interact Tech 2020;3(2):1–15, https://dl.acm.org/doi/10.1145/3406183.
[61] Yep T. Torchinfo. 2020, URL https://github.com/TylerYep/torchinfo.
[62] Reinhard K, Martin R, Michael W, Ralf S, Christopher S. Design and implementation of practical bidirectional texture function measurement devices focusing on the developments at the University of Bonn. Sensors 2014;14(5). http://dx.doi.org/10.3390/s140507753.
[63] Moritz J, James S, Haines TS, Ritschel T, Weyrich T. Texture stationarization: Turning photos into tileable textures. In: Computer graphics forum. vol. 36, Wiley Online Library; 2017, p. 177–88.
[64] Li Z, Shafiei M, Ramamoorthi R, Sunkavalli K, Chandraker M. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 2475–84.
[65] Rodriguez-Pardo C, Suja S, Pascual D, Lopez-Moreno J, Garces E. Automatic extraction and synthesis of regular repeatable patterns. Comput Graphics 2019;83:33–41.
[66] Rump M, Sarlette R, Klein R. Groundtruth data for multispectral bidirectional texture functions. In: Conference on colour in graphics, imaging, and vision; 2010. Society for Imaging Science and Technology; 2010, p. 326–31.
[67] Fan J, Wang B, Hašan M, Yang J, Yan LQ. Neural layered brdfs. In: Proceedings of SIGGRAPH 2022. 2022.
[68] Müller T, McWilliams B, Rousselle F, Gross M, Novák J. Neural importance sampling. ACM Trans Graphics (ToG) 2019;38(5):1–19.
[69] Nielsen JB, Jensen HW, Ramamoorthi R. On optimal, minimal brdf sampling for reflectance acquisition. ACM Trans Graph 2015;34(6):1–11.
[70] Wu Q, Tan J, Xu K. Palettenerf: Palette-based color editing for nerfs. 2022, arXiv preprint arXiv:221212871.
[71] Liu S, Zhang X, Zhang Z, Zhang R, Zhu JY, Russell B. Editing conditional radiance fields. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 5773–83.
[72] Wang C, Chai M, He M, Chen D, Liao J. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 3835–44.
[73] Ye W, Chen S, Bao C, Bao H, Pollefeys M, Cui Z, et al. Intrinsicnerf: Learning intrinsic neural radiance fields for editable novel view synthesis. 2022, arXiv preprint arXiv:221000647.
[74] Haque A, Tancik M, Efros A, Holynski A, Kanazawa A. Instruct-nerf2nerf: Editing 3d scenes with instructions. 2023, arXiv preprint arXiv:230312789.