DOI: 10.1111/cgf.13633
EUROGRAPHICS 2019 / P. Alliez and F. Pellacini (Guest Editors)

Volume 38 (2019), Number 2

Neural BTF Compression and Interpolation

Gilles Rainer[1] Wenzel Jakob[2] Abhijeet Ghosh[3] Tim Weyrich[1]

[1] University College London, UK
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[3]Imperial College London, UK

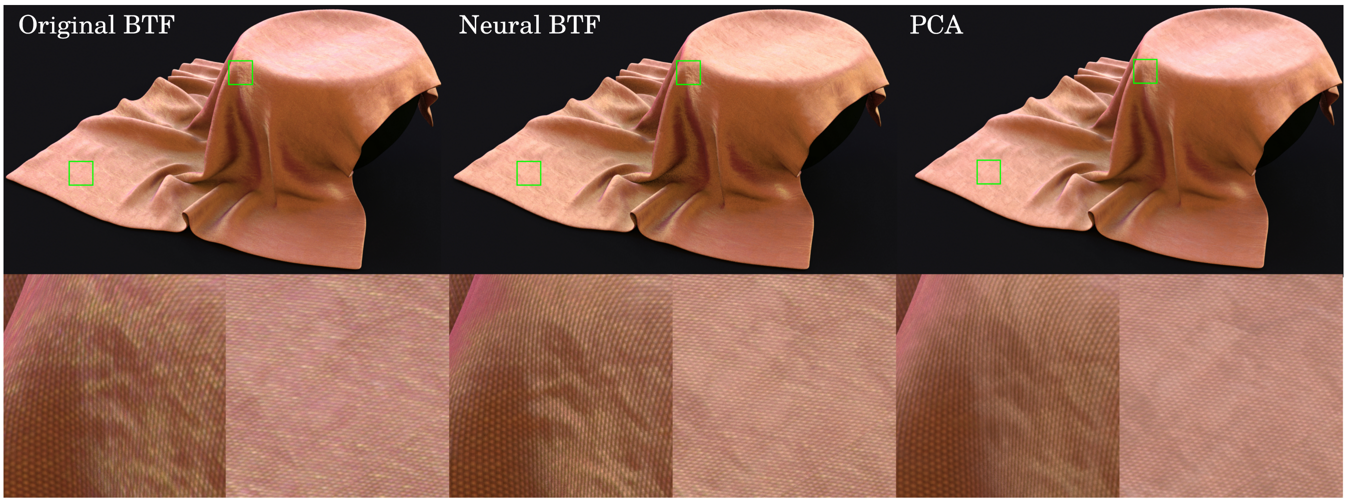

Figure 1: *BTF renderings of a challenging specular fabric (`shantung`). At the same compression ratio, our neural BTF approximation is able to capture subtle surface variations and anisotropy that are lost by principal component analysis-based compression.*

Abstract

The Bidirectional Texture Function (BTF) is a data-driven solution to render materials with complex appearance. A typical capture contains tens of thousands of images of a material sample under varying viewing and lighting conditions. While capable of faithfully recording complex light interactions in the material, the main drawback is the massive memory requirement, both for storing and rendering, making effective compression of BTF data a critical component in practical applications. Common compression schemes used in practice are based on matrix factorization techniques, which preserve the discrete format of the original dataset. While this approach generalizes well to different materials, rendering with the compressed dataset still relies on interpolating between the closest samples. Depending on the material and the angular resolution of the BTF, this can lead to blurring and ghosting artefacts. An alternative approach uses analytic model fitting to approximate the BTF data, using continuous functions that naturally interpolate well, but whose expressive range is often not wide enough to faithfully recreate materials with complex non-local lighting effects (subsurface scattering, inter-reflections, shadowing and masking...). In light of these observations, we propose a neural network-based BTF representation inspired by autoencoders: our encoder compresses each texel to a small set of latent coefficients, while our decoder additionally takes in a light and view direction and outputs a single RGB vector at a time. This allows us to continuously query reflectance values in the light and view hemispheres, eliminating the need for linear interpolation between discrete samples. We train our architecture on fabric BTFs with a challenging appearance and compare to standard PCA as a baseline. We achieve competitive compression ratios and high-quality interpolation/extrapolation without blurring or ghosting artifacts.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Reflectance modeling; Image-based rendering;*** *Neural networks; Image compression;*

1. Introduction

Photorealistic digital reproduction of real-world objects is one of the main areas of research in computer graphics. The accuracy and realism of the resulting renderings is generally limited by the faithfulness of the underlying physical model, in particular with respect to geometry and light-material interactions. For materials with complex appearance, particularly textiles, such models tend to be prohibitively complex and require a detailed description of the underlying scattering process down to interactions with individual fabric fibers. Although promising results have been achieved in specific cases, e.g., using procedural models [LZB17], it remains difficult to use this approach to reproduce the appearance of existing material samples.

Another way of rendering realistic material appearance that sidesteps the complexities of explicit modeling involves image-based acquisition methods, such as the *Bidirectional Texture Function* (BTF) proposed by Dana et al. [DvGNK99]. The BTF consists of many photographic measurements of a planar sample of a real-world material, under controlled viewing and lighting directions. The realism of the captured materials is directly related to the spatial resolution and number of angular measurements, especially for specular materials that require a dense sampling ratio. Typically, acquisition produces vast datasets with tens of thousands of textures requiring terabytes of data. The main challenge of BTF modeling, then, is to devise effective compression strategies that reduce the storage requirements to a practical amount.

Currently used local or global compression represents each BTF measurement as an entry in a large matrix or tensor that is compressed by exploiting the resulting low-rank structure, i.e. linear dependencies between different parts of the data. The most commonly used approach of *principal component analysis* (PCA)-based compression finds dependencies between spatial locations as a function of angle but does not exploit coherence in the angular dimensions themselves. None of these methods is able to consider more complex nonlinear dependencies in the high-dimensional distribution of BTF values.

This article proposes a new strategy that compresses BTF data using an asymmetric encoder-decoder network architecture. Similar to factorization-based techniques, the high-resolution textures are transformed into a low-dimensional latent representation that is decoded during rendering. In contrast to prior work, our method nonlinearly interpolates the data on the 4D angular domain and is trained to harness spatio-directional redundancies in the data. The method produces high-quality models that improve on the fidelity and compression ratio of previously used linear decompositions.

2. Related Work

The Bidirectional Texture Function introduced by Dana et al. [DvGNK99] is a 6-dimensional function of position on the material and incoming and outgoing light/view direction, commonly stored as a 4D array of two-dimensional textures. The curse of dimensionality makes dense storage of such high-dimensional data prohibitive, hence data compression is a major research topic in BTF modeling community [MBK05, FH09]. Another difficulty lies in the choice of metric to evaluate the performance of a compression algorithm – although most works use standard, perception-agnostic losses, the use of more relevant quality measures has been explored [FCGH08, JWD*14].

Matrix Factorization. Most early work on compression is based on linear matrix factorization techniques, applied to the data in 2D matrix form [KMBK03], to each sampled view direction separately [SSK03], or using decompositions into Lambertian and a specular component that are then compressed separately [KCL18]. Combining factorization with a clustering method like K-means [MMK03, TZL*02] applied to the latent representation enables the use of fewer coefficients per cluster. Other factorization techniques include hierarchical tensor decomposition methods applied to the high dimensional BTF [WWS*05, RK09] and vector quantization methods based on codebooks [KM06, HFM10, EV14].

In practice, the simplicity of standard PCA has caused it to remain the most widely used method. For instance, Weinmann et al. [WGK14] use it to compress their publicly available BTF datasets. One significant limitation to all factorization-based approaches is their relatively naïve treatment of coherence in the data, which makes no assumptions other than the existence of linear dependences. However, reflectance data is considerably more structured, which is the premise of compression methods based on analytical models.

Parametric Models. Early work on analytical models in the context of BTFs used polynomials [MGW01] and Lafortune lobes [MLH02] to model the directional dependence of each texel. Other approaches model directional variation based on the material's response to directional filter banks [TZL*02], as mixtures of parameteric models [WDR11, SPS13], spherical radial functions [TFLS11], or using a decomposition in terms of measured BRDF responses from the MERL database [WWHL07]. As a side effect, parametric methods often provide physically meaningful and potentially user-editable quantities characterizing the geometry (e.g. surface normals), surface albedo, etc. [MG09, LBAD*06].

Analytical BRDF models are generally not sufficiently expressive to capture the rich variety of local reflectance behavior observed in real-world materials, which leads to significantly higher residuals compared to factorization-based approaches. For this reason, the residual of the fit is often kept and compressed separately [MCT*05, WDR11]. Parametric methods also make additional assumptions about the data and the materials: fitting methods generally require close-to-perfect registration of the BTF data, parallax correction, as well as a clearly defined opaque material surface. Some or all of these assumptions may be violated when acquiring materials that do not do not occupy a clearly defined two-dimensional surface.

Statistical Methods. An interesting approach presented by Haindl et al. [HFA04, HF07] models the material appearance as a combination of a displacement maps with an autoregressive random field. This yields BTFs with very high compression ratios that can be expanded to any desired resolution. The two main issues with this approach are the lack of random access to texels and loss of visual fidelity for non-Lambertial materials.

Neural Networks. Several recent works have started using neural networks to render and approximate light transport [RWG*13, RDL*15]. Maximov et al. [MRF18] introduced the concept of "deep appearance maps", which use a small fully connected network as a material descriptor. Zsolnai-Fehér et al. [ZFWW18] use a neural network to render previews of materials with static scene geometry. The inverse problem has also been the focus of recent works [LDPT17, LSC18, DAD*18] that take an image as input and output estimated BRDF (or SVBRDF) parameters.

Autoencoders in particular (see [HS06, GBC16]), have received considerable attention as primitives of compression strategies: they combine the ability to learn the projection basis (like matrix factorization techniques) with the interpolative properties of analytical models that reconstruct continuous functions. The latter can be done by appending information to the latent representation, e.g. to relight scenes from new lighting directions [XSHR18], or to generate appearance parameters of a facial model based on the viewing direction [LSSS18]. Chen et al. [CWZ*18] store light field datasets as a neural network parametrized on position and direction of the viewing ray. Another example of reflectance acquisition with neural networks can be found in [KCW*18], where an autoencoder is used to decode multiplexed lighting captures of a material. However these methods are not specifically tailored to the compression of material appearance datasets, in which case both the size of latent representation as well as the network size need to be considered.

We harness two properties of autoencoders to build an effective BTF material representation: an efficient adaptive latent representation with high compression ratio in conjunction with straightforward interpolation, by making the decoder a continuous function of lighting and viewing directions.

3. Method

3.1. Input Data

The BTF $f(\mathbf{p},\lambda,\omega_i,\omega_o))$ is a seven-dimensional function of position, wavelength, and viewing angles. Partial evaluation of the BTF at a surface position $\mathbf{p}$ and wavelength λ yields a 4D function $f_{\mathbf{p},\lambda}(\omega_i,\omega_o)$ encoding the directional dependence that is known as the *apparent BRDF* (ABRDF). An important difference of ABRDFs compared to regular BRDF models is that they encode various non-local effects such as subsurface scattering.

In the discrete setting $p = (x,y) \in \mathbb{N}^2$ is a pixel coordinate, λ is a color channel of the measurement device, and the ABRDF turns into a length-n vector, with one entry for each captured combination of lighting and viewing angles. We preprocess the data in a way that is favorable for both PCA and neural compression, by applying a log transform to project the reflectance values into a perceptually more meaningful basis, and then whitening the texels by subtracting the mean of the ABRDF and dividing by the standard deviation.

PCA-based compression techniques of BTFs typically arrange all ABRDFs as columns of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where m is the number of BTF texels. The matrix is subsequently decomposed

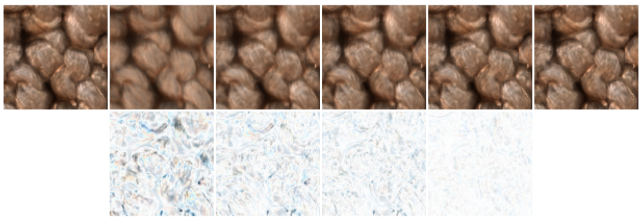
Figure 2: ***Top:*** *Input BTF and PCA approximations using 8, 16, 32, 64, 128 singular values.* ***Bottom:*** *Negative image of difference magnitude multiplied by 2.* ***Dataset:*** `Carpet05` *from [WGK14].*

using a *singular value decomposition* (SVD), which expresses the data a sum of outer products

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{1}$$

where k denotes the *rank* of the approximation. Real-world appearance data typically exhibits a numerical rank $k \ll n$, which is the foundation of this compression technique. Color is handled using a simple generalization of this scheme, which we skip here for simplicity. Figure 2 shows a series of approximations with progressively higher rank.

It is instructive to consider the shape of the resulting decomposition into $\mathbf{U}$ and $\mathbf{V}$ (the matrix $\mathbf{\Sigma}$ is normally merged into either one of them): given n viewing-lighting pairs, and m texels, the matrices $\mathbf{U}$ and $\mathbf{V}$ will have $m\times k$ and $k\times n$ entries, respectively. The size of the compressed dataset is thus related to the product of the rank k and the number of angular and spatial samples. The $\mathbf{U}$ matrix takes the role of the original photographs and can be rearranged into a set of images with k channels. The $\mathbf{V}$-matrix acts as a type of "decoder", since it converts the spatially-varying coefficients into a list of values that describe the directional reflectance behavior.

Compared to optimization-based approaches, the singular value decomposition can be computed at a moderate cost, but this assumes that all BTF data can fit into main memory. When dealing with BTFs that have a fine angular discretization (e.g. the datasets provided by Weinmann et al. [WGK14]), the decoder can become very large and have a significant effect on the overall storage footprint. The resolution of the textures must also be taken into account: for BTFs with a high spatial resolution, it may be necessary to perform the SVD on separate BTF tiles or texel clusters [MMK03], which implies having to store multiple decoders.

3.2. Neural BTF compression

Our neural BTF compression strategy is inspired by autoencoder networks [B*09]. These architectures use an *encoder* to transform a d-dimensional input vector into a latent representation of dimensionality $k \ll d$. A subsequent *decoder* network transforms the latent representation back into a d-dimensional output vector that can be compared to the input value. A simple training loss minimizes the difference between the input and output pair, and a sequence of training iterations then attempts to turn the combination of encoder and decoder into the identity function or a good approximation thereof. The original aim of this dimensional "funnel" was not

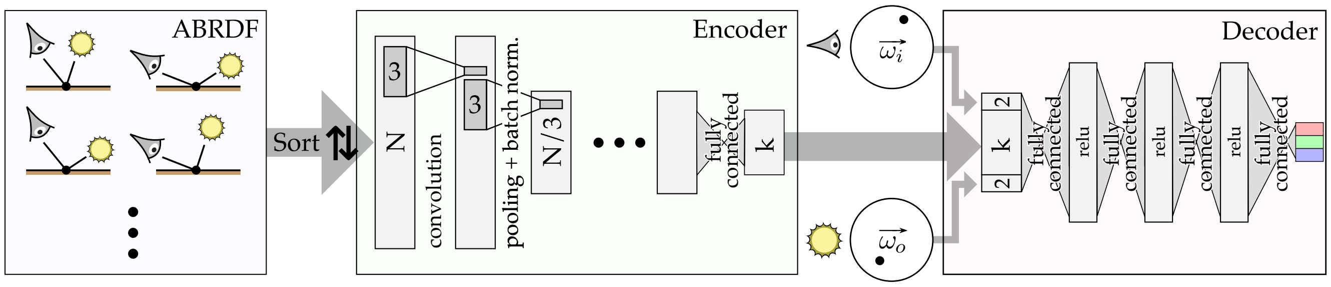

Figure 3: *Our neural BTF builds on an asymmetric encoder-decoder architecture. The encoder receives a per-pixel* apparent BRDF *(ABRDF) as input, i.e. a set of angular configurations associated with a single spatial location. These measurements are then reordered coherently (sorted by ascending spherical coordinates of light and view direction) and downsampled by a sequence of convolution layers until a fully connected layer finally produces a low-dimensional ($k = 8$) latent representation that is stored on disk. The decoder concatenates this vector with the view and camera direction projected onto their respective tangent spaces and passes them through a sequence of fully connected layers with componentwise nonlinearities. The last layer outputs a single RGB color.*

to compress data, but rather to force the network to learn the low-dimensional structure of a training dataset.

In contrast, our aim is to use an encoder to compress the input dataset and only store the latent vectors along with the decoder that is used to recover an approximation the original dataset; the encoder is no longer needed after the dataset has been transformed and can be discarded. Since we train a specific encoder and decoder for each dataset, our method is related to previous optimization-based BTF approximation methods.

Neural autoencoders not only subsume the type of dimensionality reduction enabled by PCA [BK88] but considerably exceed it due to the ability of introducing nonlinearities that can be used to recognize more complex dependences in the input data.

Many autoencoder networks in the literature are symmetric, which means that every layer of the decoder is a mirror analogue of a corresponding layer in the encoder—for instance, convolutions turn into transpose convolutions, etc. In our case, the input of the encoder is an ABRDF, in which case a symmetric decoder would also output an ABRDF. However, this is less than ideal when considering the usage in an actual rendering system: standard rendering algorithms will only query the BTF for a single angle pair (ω_i, ω_o), which means that almost all entries of the ABRDF would be computed in vain. BTF evaluation may occur millions to billions of time in a rendering, making such a wasteful approach impractical.

We are therefore interested in a simplified decoder that is small enough to run at high frequency, and which only evaluates a single angle pair that is provided as an additional input along with the latent ABRDF representation. As a consequence, the decoder becomes a regressor of the directional behavior with the added benefit that the renderer is freed from somewhat tedious aspects of BTF evaluation, such as linear interpolation and extrapolation on the spherical domain. Figure 3 illustrates our encoder and decoder, which we now discuss in turn.

Encoder network. Since the encoder is only used during training, there are no particular constraints on the architecture or its size. We rely on a sequence of 1D convolution layers with max-pooling and batch normalization to reduce the input to a sufficiently small size before transforming it into an 8-dimensional latent representation using a fully connected layer.

The number of convolutions is related to the directional resolution of the input dataset: for the Bonn dataset (22801 configurations), we repeat the illustrated downsampling layer sequence four times (each reducing the amount of data by a factor of 3×). We also acquired two new BTF datasets using a motorized gonioreflectometer. These BTFs have approximately twice the spatial resolution but only 1508 angular configurations, hence only a single downsampling layer is used.

Decoder network. Our decoder architecture is shown on the right side of Figure 3. We express the input light and view directions as a pair of 2D vectors using stereographic projection and concatenate them with the latent representation that is subsequently transformed by four layers with element-wise nonlinearities; the output of the decoder is a single RGB value. This asymmetry implies that the encoder must be evaluated n times (once for each angular configuration) to compute the final loss.

Training relies on stochastic gradient descent with a batch size of 5 and a learning rate of 0.05 and 0.1 for our and the Bonn datasets, respectively. We investigated the influence of the amount of fully connected layers in the decoder: Figure 4 shows the L2 loss (on the pre-processed test dataset) for different decoder architectures, all trained for the same number of iterations (400 epochs) on the newly captured `shantung` dataset. The loss decreases with the number of layers. Beyond 4-5 hidden layers, the improvement in the loss compared to the increase in size of the decoder is marginal. We

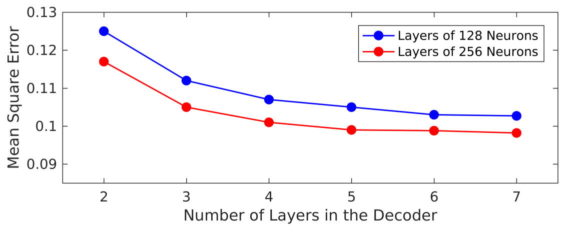

Figure 4: *Test loss for identical networks except for the number of layers in the decoder.* ***Dataset:*** `shantung` *(from our database).*

therefore use a decoder network with 4 hidden layers as a compromise between accuracy and computation time. To facilitate evaluation, we use layers with 106 neurons, which leads to a decoder that has the same number of coefficients as the **V** matrix used by PCA on our datasets. Note that PCA's decoder is considerably bigger for the Bonn dataset due to the increased angular density.

We also study the influence of the parameter k. The decoder can easily reconstruct the shape of an individual texel's reflectance function, but generalizing to all surface positions is more challenging and requires a sufficiently high-dimensional latent representation. Figure 5 shows the reconstruction error as the number

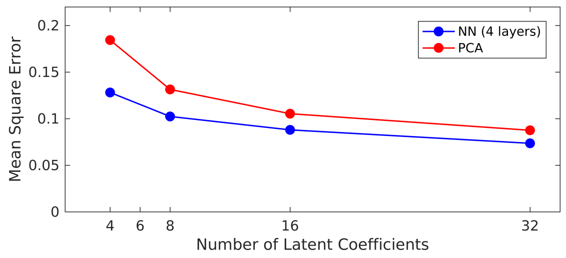

Figure 5: *Error (on the preprocessed dataset) as a function of latent coefficients, for a PCA decoder matrix and a neural decoder of the same size.* ***Dataset:*** *`shantung` (from our database).*

of latent dimensions increases, both for PCA and our approach, applied to our `shantung` dataset. Our approach performs better than PCA, achieving approximately twice the compression ratio. As the number of latent dimensions increases, the difference between our approach and PCA, when given the same storage budget, decreases. We use $k = 8$ latent coefficients in the remainder of this article which, including mean and standard deviation, adds up to approximately the same number of coefficients per texel as a simple SVBRDF model.

4. Results and Evaluation

Our evaluation is based on two sources of data: BTFs from one of our own capture setups (Figure 13) and BTFs from the publicly available datasets of [WGK14]. We specifically chose materials like shiny fabrics and wool to have a diverse set of materials with complex appearance functions. Our own datasets use a lower angular resolution (1508 light/view configurations with irregular sampling) but a higher spatial resolution (800×800 texels). The datasets from Bonn University have a very high angular resolution ($151 \times 151 = 22801$ light/view configurations) for a slightly lower spatial resolution (400×400 texels). Since the light sources used for the BTF capture at Bonn are the camera flashes, the sampling pattern on the light and view hemisphere is approximately identical. Table 1 provides numerical results for all the datasets shown in the figures, along with a breakdown of the different components of both PCA and our method that affect the compression ratio. The angular sampling of the BTF determines the size of the PCA decoder, and the spatial sampling of the BTF affects both PCA and our method's latent map size. Table 2 reports reconstruction scores on a wider range of datasets from the Bonn database [WGK14].

4.1. Compression

We compare the compression performance of our representation to that of PCA, which is the most commonly used technique used to represent compressed BTF datasets (e.g. the datasets by Weinmann et al. [WGK14]). We do not focus on optimizations, such as local PCA [MMK03]: such approaches are orthogonal and could be applied to reduce the number of latent coefficients for both PCA and our autoencoder network.

We perform comparisons on cropped datasets of 256x256 texels for our BTFs, and 100x100 texels for BTFs from Weinmann et al. [WGK14], training on the L_2 loss that is effectively also used by PCA. A marginal improvement in the reconstruction error could be obtained by training for a higher number of iterations, but we choose to stop at 400 epochs (approximately 5 hours on an NVidia GeForce GTX 980 Ti).

Table 1 shows the results of our method on multiple datasets, compared to PCA with 8 and 16 coefficients. For our datasets, which have a low angular resolution (i.e. a small PCA decoder), the compression ratio our method achieves is the same as for PCA with 8 coefficients. The improvement in reconstruction error varies depending on the dataset: The `cotton` dataset (Figure 6) has a regular structure and fairly Lambertian reflectance, which makes it easy to compress with both methods. Comparing the scores on this dataset is not particularly meaningful, as the residual is mostly noise present in the original BTF. Both PCA and our approach faithfully reconstruct the original appearance and efficiently filter out the color noise, returning a cleaner texture.

For our `shantung` material however, the appearance is much harder to encode and compress linearly due to anisotropy and specular highlights. In this case, our method significantly outperforms PCA's compression ratio. The network manages to reconstruct the high-frequency details, both in the angular and in the spatial domain. As shown in Figure 7, the results are arguably even better than the PCA reconstruction with 32 coefficients (4 times lower compression ratio), since the network manages to preserve visually pertinent features.

The datasets from Bonn University have already been precompressed using PCA with 101 coefficients—the worst case, our comparison on these datasets is favorably biased towards PCA. Nevertheless, our approach achieves an over tenfold compression

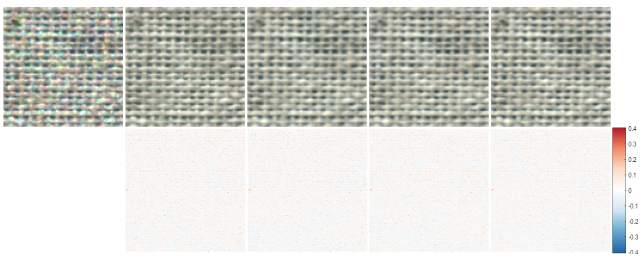

Figure 6: ***Top row (from left to right):*** *Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles: 0, 90, 182.4, 35.1.* ***Bottom row:*** *Signed L2 error of the reconstructions compared to the ground truth.* ***Dataset:*** *`cotton` from our database.*

	shantung (Ours)	cotton (Ours)	carpet05 (Bonn)	carpet07 (Bonn)	fabric04 (Bonn)	leather04 (Bonn)
Input Size	296,485	296,485	684,030	684,030	684,030	684,030
PCA(8) RMS Error	**0.0422**	**0.06357**	**0.01888**	**0.01832**	**0.0189**	**0.3513**
Decoder, Maps & Total Size	36, 542, **578**	36, 542, **578**	547, 80, **627**	547, 80, **627**	547, 80, **627**	547, 80, **627**
Compression Ratio	**512.5**	**512.5**	**1090**	**1090**	**1090**	**1090**
PCA(16) RMS Error	**0.0378**	**0.0554**	**0.01436**	**0.01357**	**0.01448**	**0.0248**
Decoder, Maps & Total Size	72, 1049, **1121**	72, 1049, **1121**	1094, 160, **1254**	1094, 160, **1254**	1094, 160, **1254**	1094, 160, **1254**
Compression Ratio	**256**	**256**	**545**	**545**	**545**	**545**
Ours(8) RMS Error	**0.0375**	**0.0598**	**0.0133**	**0.01237**	**0.0173**	**0.0272**
Decoder, Maps & Total Size	36, 542, **578**	36, 542, **578**	36, 80, **116**	36, 80, **116**	36, 80, **116**	36, 80, **116**
Compression Ratio	**512.5**	**512.5**	**5897**	**5897**	**5897**	**5897**

Table 1: *Size of the components to store (in thousands of coefficients), reconstruction error and compression ratio for PCA with 8 coefficients, PCA with 16 coefficients, and our network with 8 latent coefficients and 4 hidden linear layers of 106 neurons in the decoder.*

	carpet03	carpet12	fabric02	fabric05	felt05	felt10	leather06	leather11	stone04	stone05	wallpaper06	wallpaper11	wood01	wood06
PCA (**1:1090**)	0.01102	0.0147	0.0147	0.0265	0.0188	0.0053	0.0219	0.0639	0.4319	0.0103	0.0134	0.01448	0.00673	0.0145
Ours (**1:5897**)	0.00948	0.0118	0.0117	0.0222	0.016	0.0048	0.0179	0.0433	0.2429	0.0088	0.0118	0.01332	0.00852	0.0131

Table 2: *Root Mean Square Reconstruction Error on 2 datasets from each class of the Bonn Material Database for additional comparisons.*

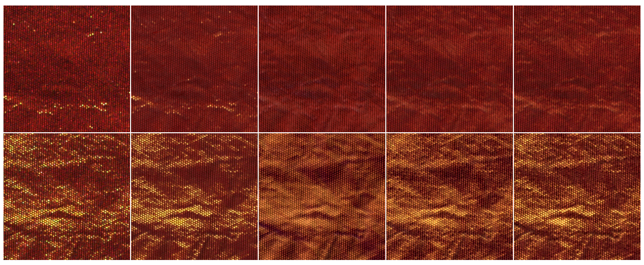

Figure 7: ***Columns from left to right:*** *Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles in the* ***top row:*** *0, 90, 182.4, 35.1.* ***Bottom row:*** *222.5, 77.2, 44.9, 33.4.* ***Dataset:*** *Shantung from our database.*

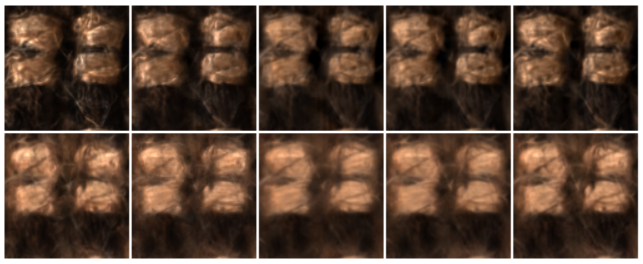

Figure 8: ***Columns from left to right:*** *Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles in the* ***top row:*** *0, 90, 180, 45.* ***Bottom row:*** *270, 30, 0, 90.* ***Dataset:*** *carpet07 from the Bonn Database.*

improvement on PCA for datasets like carpet05, carpet07, depending on how much the appearance of the texels lends itself to compression. Even in the worst cases, our approach consistently outperforms PCA on the reconstruction error by a wide margin, at an over 5-fold increase in compression.

Linear techniques like PCA tend to return the best-fitting mean solution with low-frequency variations, while the non-linearities allow our network to capture the high-frequency variations even at very high compression ratios. The lower error also translates into

	PCA (18 coeffs)	[RK09]	Our method
RMS Error	0.041	0.033	0.0404
Size (MB)	3.0	3.0	1.2

Table 3: *Comparison to [RK09] on the* Pulli *dataset.*

a considerable visible improvement: PCA tends to apply blur both in the spatial and angular domain, while our network (at the same number of latent coefficients) seems to efficiently capture details that matter perceptually and contribute to a visually more faithful reconstruction (as Figure 7 shows).

For further evaluation, we apply our method on a dataset that was featured in many previous BTF compression articles – the Pulli sample from the 2003 Bonn BTF datasets (UBO2003). Table 3 compares the performance of our method (still the same architecture) with the Sparse Tensor Decomposition from [RK09]. For the data size comparison, we store our latent maps as well as the network decoder layers as 16-bit float OpenEXR images.

4.2. Interpolation

BTF datasets consist of a texture sampled at a discrete set of light-view combinations. To obtain the appearance of the texture at new light-view directions, the standard approach entails interpolating the closest 3 directions based on a Delaunay triangulation of the measurement locations. Note that interpolation is simultaneously needed in *both* the incident and outgoing direction argument. Special treatment is furthermore needed when the chosen direction lies outside of the convex hull of the measured directions, in which case we interpolate the nearest two directions. When the dataset is compressed with PCA, this type of interpolation remains necessary.

In our approach, this part is considerably simplified: the light and view directions are both continuous parameters of the decoder network, hence no interpolation must be performed during rendering. However, we must still verify that the decoder behaves sensibly when evaluating regions that lie between measurement locations, or even outside of their convex hull. To do so, we train on subsets of the original BTFs and use the remaining textures as ground truth images for evaluation.

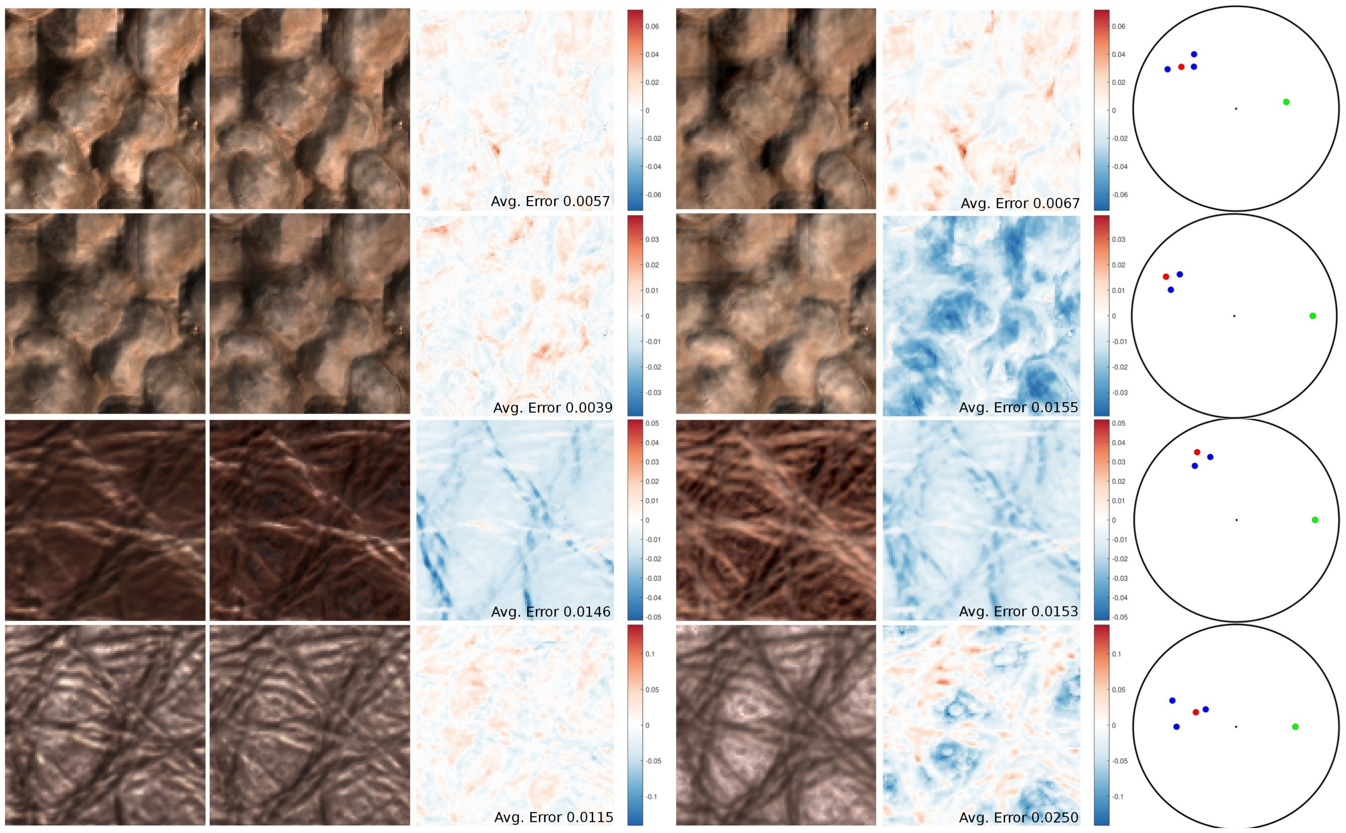

Figure 9: *Ground truth, reconstruction with our method and with PCA, with error images. Angles displayed on the unit disk: view direction (green), light direction (red), and light directions used for PCA interpolation (blue).* ***Datasets:*** `carpet05` *and* `leather04` *from [WGK14].*

Ground Truth Texture Comparisons We cross-validate the angular dependence of our network and PCA with linear interpolation against the ground-truth. The ground truth images were neither in the network training dataset nor in the matrix decomposed by PCA. We show a signed error plot, which displays the L_2 distance between the reconstructed pixel and the original value. If the norm of the original pixel is higher than the reconstructed one, the sign is positive (red), and in the opposite case the sign is negative (blue).

Figure 9 shows comparisons for 2 datasets from [WGK14]. For `carpet05`, we removed 20% of the original dataset. When the lighting direction is surrounded by nearby samples, PCA interpolation produces good results. When the lighting direction is further from the zenith however, changes in shadowing become significant even for small perturbations to the lighting direction. The blur induced by the barycentric interpolation becomes noticeable, and contrast is reduced (first row). Our network preserves these details, even when extrapolating outside the convex hull of samples (second row). Linear interpolation reverts to the two closest samples in this case, which produces an image with a lighting configuration that significantly differs from the ground truth, while our network does not encounter such difficulties and produces plausible output. The `carpet05` dataset provides the most favorable setting for interpolation from discrete samples since the material has a strong Lambertian component and a low-frequency spatial variation in height.

The `leather04` dataset at the bottom of Figure 9 is more challenging due to high-frequency normal variations and the specularity of the surface. This means that the appearance can change drastically between close light or view directions (rows 3 and 4). Here, we removed 50% of the original dataset's samples. In both examples, our network is able to successfully "hallucinate" appearance details and achieve a more faithful match to the ground truth.

Figure 10 shows results on the shantung dataset from our database, from which we removed 20% of the textures. The angular sampling in the original dataset is already much lower than in the datasets from [WGK14], which makes it even harder to interpolate linearly. Furthermore, the material has very specular glints as well as a strong anisotropic component. Not only does the interpolation from PCA miss specular highlights, it also changes the overall tone of the images, making the reconstructed textures unusable for rendering. Our network produces higher-quality results and manages to capture a reasonable part of the specular highlights while removing shot noise that was present in the input photographs.

Figures 9 and 10 only display characteristic examples for these materials, but we also generated interpolated images for many other configurations that are provided in the supplemental material.

Angular Plots Another way of evaluating the interpolation performance is to consider a single texel for a fixed viewing angle (Figure 11). We plot the dependence on the lighting direction under stereographic projection onto a disk. For both materials, extrap-

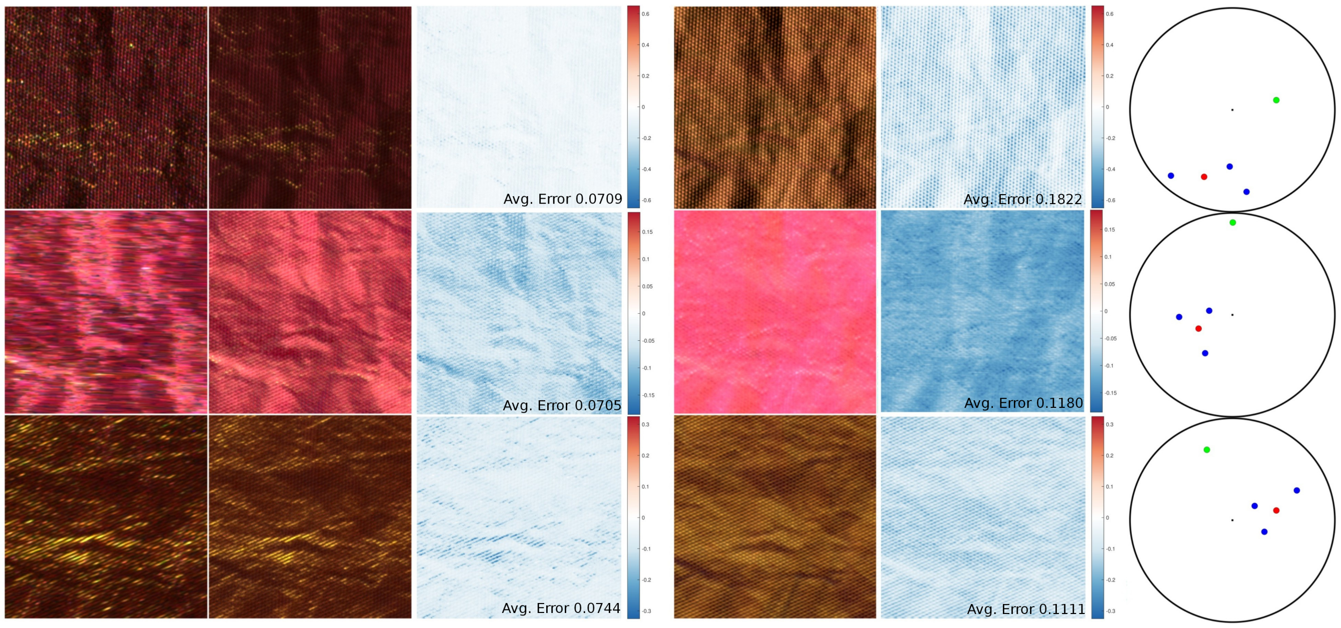

Figure 10: *Ground truth, reconstruction with our method and with PCA, with respective error images. Angles are displayed on the unit disk: view direction (green), light direction (red), and light directions used for PCA interpolation (blue).* ***Dataset:*** `shantung` *(from our database).*

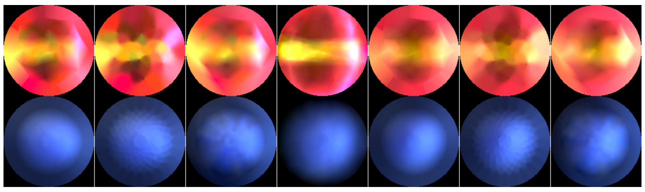

Figure 11: *Angular plots of ABRDF slices (fixed view, varying lighting) with different interpolation strategies. From left to right: Original BTF (barycentric, lumigraph, natural neighbor), neural network, PCA (barycentric, lumigraph, natural-neighbor).* ***Dataset:*** `shantung` *(top),* `fabric04` *(bottom).*

olation outside of the convex hull of samples is challenging. We test different interpolation strategies: barycentric interpolation on the triangulated set of sample directions, unstructured lumigraph blending [BBM*01], and natural neighbor interpolation [Sib81].

For BTFs with a sparse angular sampling like the `shantung` BTF, all three interpolation strategies exhibit seams and drawbacks. Only the network manages to reconstruct a familiar looking reflectance profile—the angular plot displays 2 elliptical anisotropic highlights that are typical of shiny fiber materials such as silk. Although there is no ground truth reference for this comparison, we find the angular output produced by the network the most plausible.

Rendering Comparisons We use the open source renderer Mitsuba [Jak10] to create high-resolution renderings of a cloth draped over a bowl (Figure 12). The cloth model, lighting and view conditions remain the same throughout the renderings. In the left and right columns, reflectance values are linearly interpolated using the original BTF values or the PCA reconstruction respectively. In the middle, we render with our neural network. Both PCA and our method of compression preserve the essential part of the material appearance; even to the trained eye, the differences are minimal.

Our PCA and BTF shaders are unoptimized and involve a linear sweep through the dataset to find the nearest light/view direction, making speed comparisons unfair towards PCA. Timings on our datasets: BTF 8.2 min, PCA 8.8 min, NN 11.2 min, LAMBERTIAN 2.1 min. Timings on Bonn datasets: BTF 1.2 hrs, PCA 1.1 hrs, NN 12 min, LAMBERTIAN 2.1 min. We nevertheless interpret this as indicative that the rendering performance of our neural model is at least acceptable, but we do not claim superior rendering speeds.

For additional visualisations, we refer to animations in the supplemental material. We render a square textured with a single repetition of the BTF texture, under moving point light illumination with unstructured lumigraph blending for the BTF and PCA.

5. Conclusions and Future Work

In this paper, we presented a BTF model based on a neural network. We train the network separately for each new dataset, and store the latent vectors with the decoder network as a compressed representation of the original dataset. We achieve competitive compression ratios due to our method's ability to exploit nonlinear dependencies in the dataset, as well as the decoder's continuous parametrization on lighting and viewing directions.

Our technique is especially well suited for storage and transmission of BTFs because of the small memory footprint of the network, as well as for photorealistic renderings due to the improved interpolation. In the future, it would be interesting to experiment with different error metrics for the network training (SSIM for instance) as well as different color spaces, which could lead to perceptually more accurate results. Finally, it may also be possible to perform

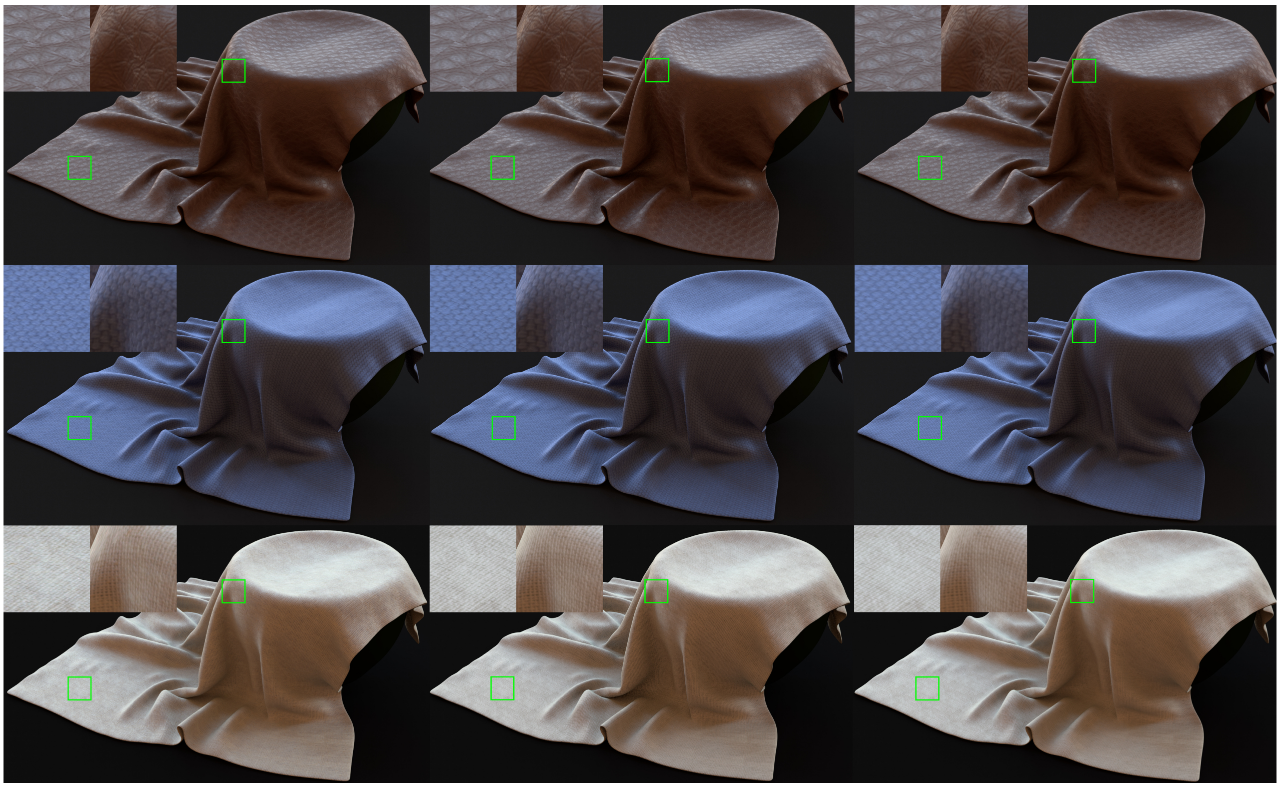

Figure 12: *Renderings with Mitsuba's pathtracer (2000x1200 pixels, 128 samples per pixel, 10 processes, uniformly sampled BSDFs, environment lighting). Left to right: original BTF, our network and PCA.* ***Datasets*** *(top to bottom):* `leather04`, `fabric04`, `cotton`.

texture synthesis on the latent maps generated by our encoder network, to generate BTFs with an arbitrarily large spatial resolution.

Acknowledgments

We would like to thank Change of Paradigm Ltd. for supporting this work, and Reinhard Klein and his team for providing data and helping with comparisons. We would also like to acknowledge the EPSRC Early Career Fellowship EP/N006259/1, and the EPSRC grant EP/K023578/1.

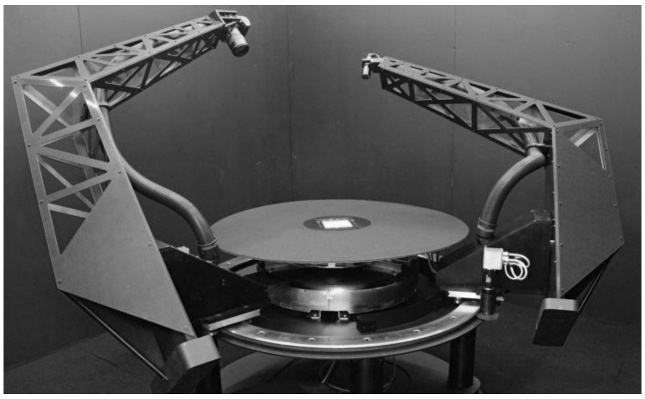

Figure 13: *Our custom four-axis gonioreflectometer uses white-LED illumination and a Canon 5DSR 50-megapixel camera to flexibly sample high-dynamic range BTFs at 75µm texel resolution (downsampled from up to 37µm pixels for orthographic views).*

References

[B*09] Bengio Y., et al.: Learning deep architectures for ai. *Foundations and trends® in Machine Learning 2*, 1 (2009), 1–127. 3

[BBM*01] Buehler C., Bosse M., McMillan L., Gortler S., Cohen M.: Unstructured lumigraph rendering. *Proc. SIGGRAPH* (2001), 425–432. 8

[BK88] Bourlard H., Kamp Y.: Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics 59*, 4 (Sep 1988), 291–294. 4

[CWZ*18] Chen A., Wu M., Zhang Y., Li N., Lu J., Gao S., Yu J.: Deep surface light fields. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 1 (July 2018), 14:1–14:17. 3

[DAD*18] Deschaintre V., Aittala M., Durand F., Drettakis G., Bousseau A.: Single-image SVBRDF capture with a rendering-aware deep network. *ACM Trans. on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 128:1–128:15. 3

[DvGNK99] Dana K. J., van Ginneken B., Nayar S. K., Koenderink J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. on Graphics 18*, 1 (Jan. 1999), 1–34. 2

[EV14] Egert P., Vlastimil H.: Parallel BTF compression with multi-level vector quantization in OpenCL. In *Pacific Graphics Short Papers* (2014), Keyser J., Kim Y. J., Wonka P., (Eds.), The Eurographics Association. 2

[FCGH08] Filip J., Chantler M. J., Green P. R., Haindl M.: A psychophysically validated metric for bidirectional texture data reduction. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* (2008), 138:1–138:11. 2

[FH09] Filip J., Haindl M.: Bidirectional texture function modeling: A state of the art survey. *IEEE Tr. Pat. An. & Mach. Intel. (PAMI) 31*, 11 (Nov. 2009), 1921–1940. 2

[GBC16] Goodfellow I., Bengio Y., Courville A.: *Deep Learning*. MIT Press, 2016. www.deeplearningbook.org. 3

[HF07] Haindl M., Filip J.: Extreme compression and modeling of bidirectional texture function. *IEEE Tr. Pat. An. & Mach. Intel. (PAMI) 29*, 10 (Oct. 2007), 1859–1865. 2

[HFA04] Haindl M., Filip J., Arnold M.: BTF image space utmost compression and modelling method. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03* (Washington, DC, USA, 2004), ICPR '04, IEEE Computer Society, pp. 194–197. 2

[HFM10] Havran V., Filip J., Myszkowski K.: Bidirectional texture function compression based on multi-level vector quantization. *Computer Graphics Forum 29*, 1 (2010), 175–190. 2

[HS06] Hinton G., Salakhutdinov R.: Reducing the dimensionality of data with neural networks. *Science (N.Y.) 313* (08 2006), 504–7. 3

[Jak10] Jakob W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 8

[JWD*14] Jarabo A., Wu H., Dorsey J., Rushmeier H., Gutierrez D.: Effects of approximate filtering on the appearance of bidirectional texture functions. *IEEE Trans. Visualization and Computer Graphics 20*, 6 (June 2014), 880–892. 2

[KCL18] Kim Y. H., Choi J., Lee K. H.: An efficient method for specular-enhanced BTF compression. *Computer & Graphics 75* (06 2018), 1–10. 2

[KCW*18] Kang K., Chen Z., Wang J., Zhou K., Wu H.: Efficient reflectance capture using an autoencoder. *ACM Trans. on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 127:1–127:10. 3

[KM06] Kawai N., Matsufuji K.: Azimuth-rotated vector quantization for BTF compression. In *ACM SIGGRAPH 2006 Research Posters* (New York, NY, USA, 2006), SIGGRAPH '06, ACM. 2

[KMBK03] Koudelka M. L., Magda S., Belhumeur P. N., Kriegman D. J.: Acquisition, compression, and synthesis of bidirectional texture functions. In *In ICCV 03 Workshop on Texture Analysis and Synthesis* (2003). 2

[LBAD*06] Lawrence J., Ben-Artzi A., DeCoro C., Matusik W., Pfister H., Ramamoorthi R., Rusinkiewicz S.: Inverse shade trees for non-parametric material representation and editing. *ACM Trans. on Graphics (Proc. SIGGRAPH) 25*, 3 (July 2006), 735–745. 2

[LDPT17] Li X., Dong Y., Peers P., Tong X.: Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Trans. on Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 45:1–45:11. 3

[LSC18] Li Z., Sunkavalli K., Chandraker M. K.: Materials for masses: SVBRDF acquisition with a single mobile phone image. *Proc. Eur. Conf. Comp. Vision (ECCV)* (2018). 3

[LSSS18] Lombardi S., Saragih J., Simon T., Sheikh Y.: Deep appearance models for face rendering. *ACM Trans. on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 68:1–68:13. 3

[LZB17] Luan F., Zhao S., Bala K.: Fiber-level on-the-fly procedural textiles. *Computer Graphics Forum (Proc. EGSR) 36*, 4 (July 2017), 123–135. 2

[MBK05] Müller G., Bendels G. H., Klein R.: Rapid synchronous acquisition of geometry and appearance of cultural heritage artefacts. In *Proceedings of the 6th International Conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage* (Aire-la-Ville, Switzerland, Switzerland, 2005), VAST'05, Eurographics Association, pp. 13–20. 2

[MCT*05] Ma W.-C., Chao S.-H., Tseng Y.-T., Chuang Y.-Y., Chang C.-F., Chen B.-Y., Ouhyoung M.: Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 187–194. 2

[MG09] Menzel N., Guthe M.: g-BRDFs: an intuitive and editable BTF representation. *Computer Graphics Forum* (2009). 2

[MGW01] Malzbender T., Gelb D., Wolters H.: Polynomial texture maps. *Proc. SIGGRAPH* (2001), 519–528. 2

[MLH02] McAllister D. K., Lastra A., Heidrich W.: Efficient rendering of spatial bi-directional reflectance distribution functions. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (Aire-la-Ville, Switzerland, Switzerland, 2002), HWWS '02, Eurographics Association, pp. 79–88. 2

[MMK03] Müller G., Meseth J., Klein R.: Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (Nov. 2003), Ertl T., Girod B., Greiner G., Niemann H., Seidel H.-P., Steinbach E., Westermann R., (Eds.), Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 271–280. 2, 3, 5

[MRF18] Maximov M., Ritschel T., Fritz M.: Deep appearance maps. *arXiv:1804.00863* (2018). 3

[RDL*15] Ren P., Dong Y., Lin S., Tong X., Guo B.: Image based relighting using neural networks. *ACM Tr. Graph. (Proc. SIGGRAPH) 34*, 4 (July 2015), 111:1–111:12. 3

[RK09] Ruiters R., Klein R.: BTF compression via sparse tensor decomposition. *Proc. Eurographics Symposium on Rendering* (2009), 1181–1188. 2, 6

[RWG*13] Ren P., Wang J., Gong M., Lin S., Tong X., Guo B.: Global illumination with radiance regression functions. *ACM Trans. on Graphics (Proc. SIGGRAPH) 32*, 4 (July 2013), 130:1–130:12. 3

[Sib81] Sibson R.: A brief description of natural neighbor interpolation (chapter 2). *V. Barnett, Interpreting Multivariate Data* (1981), 21–36. 8

[SPS13] Silva N., Paulo Santos L.: Interactive high fidelity visualization of complex materials on the gpu. *Computer & Graphics, Technical Section 37*, 7 (Nov. 2013), 809–819. 2

[SSK03] Sattler M., Sarlette R., Klein R.: Efficient and realistic visualization of cloth. *Proc. Eurographics Workshop on Rendering* (2003), 167–177. 2

[TFLS11] Tsai Y.-T., Fang K.-L., Lin W.-C., Shih Z.-C.: Modeling bidirectional texture functions with multivariate spherical radial basis functions. *IEEE Tr. Pat. An. & Mach. Intel. (PAMI) 33*, 7 (July 2011), 1356–1369. 2

[TZL*02] Tong X., Zhang J., Liu L., Wang X., Guo B., Shum H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH) 21*, 3 (July 2002), 665–672. 2

[WDR11] Wu H., Dorsey J., Rushmeier H.: A sparse parametric mixture model for BTF compression, editing and rendering. *Computer Graphics Forum (Proc. Eurographics) 30* (2011), 465–473. 2

[WGK14] Weinmann M., Gall J., Klein R.: Material classification based on training data synthesized using a BTF database. *Proc. Eur. Conf. Comp. Vision (ECCV)* (2014), 156–171. 2, 3, 5, 7

[WWHL07] Weistroffer R. P., Walcott K. R., Humphreys G., Lawrence J.: Efficient basis decomposition for scattered reflectance data. *Proc. Eurographics Symposium on Rendering* (2007), 207–218. 2

[WWS*05] Wang H., Wu Q., Shi L., Yu Y., Ahuja N.: Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Trans. on Graphics (Proc. SIGGRAPH) 24*, 3 (July 2005), 527–535. 2

[XSHR18] Xu Z., Sunkavalli K., Hadap S., Ramamoorthi R.: Deep image-based relighting from optimal sparse samples. *ACM Trans. on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 126:1–126:13. 3

[ZFWW18] Zsolnai-Fehér K., Wonka P., Wimmer M.: Gaussian material synthesis. *ACM Trans. on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 76:1–76:14. 3