DOI: 10.1111/cgf.13921
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)
Volume 39 (2020), Number 2

Unified Neural Encoding of BTFs

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

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

Figure 1: (Left) *We train a network to encode/decode BTF texels using 77 materials from the Bonn BTF database [WGK14].* **(Center)** *Each texel's appearance is projected into a shared, 32-dimensional encoding space.* **(Right)** *We show renderings for an* ***unseen*** *material from each of the 7 classes in the database (bottom row of the database), rendered directly from the latent projection.*

Abstract

Realistic rendering using discrete reflectance measurements is challenging, because arbitrary directions on the light and view hemispheres are queried at render time, incurring large memory requirements and the need for interpolation. This explains the desire for compact and continuously parametrized models akin to analytic BRDFs; however, fitting BRDF parameters to complex data such as BTF texels can prove challenging, as models tend to describe restricted function spaces that cannot encompass real-world behavior. Recent advances in this area have increasingly relied on neural representations that are trained to reproduce acquired reflectance data. The associated training process is extremely costly and must typically be repeated for each material. Inspired by autoencoders, we propose a unified network architecture that is trained on a variety of materials, and which projects reflectance measurements to a shared latent parameter space. Similarly to SVBRDF fitting, real-world materials are represented by parameter maps, and the decoder network is analog to the analytic BRDF expression (also parametrized on light and view directions for practical rendering application). With this approach, encoding and decoding materials becomes a simple matter of evaluating the network. We train and validate on BTF datasets of the University of Bonn, but there are no prerequisites on either the number of angular reflectance samples, or the sample positions. Additionally, we show that the latent space is well-behaved and can be sampled from, for applications such as mipmapping and texture synthesis.

CCS Concepts

• ***Computer Graphics*** → *Rendering;* • ***Material Appearance*** → *BTFs & Neural Models;*

1. Introduction

One of the main goals in Computer Graphics is to create photorealistic renderings that appear plausible to the human eye. Beyond scene geometry, a lot of the visual plausibility comes from the realism of the rendered materials. While many analytic reflectance models can create realistic-looking materials, this is only one part of the challenge. The bigger part of the difficulty lies in matching the appearance of existing, real-world materials.

To tackle this issue, many research efforts have been focused on building devices that can take measurements of a given material's appearance. Typically, this means capturing calibrated photographs of a material sample at a number of different combinations of viewing and lighting direction (see [DvGNK99]). Rendering from such a discrete set of textures is quite impractical as it integrates suboptimally with the rest of the pipeline: at rendering time, we want to be able to query the appearance at an arbitrary view-light configuration, which has to be interpolated from the discrete set of measurements.

The obvious solution to this is to transform the discrete list of reflectance measurements at each point into a continuous representation that is practical for rendering. Analytic spatially-varying bidirectional reflectance distribution functions (SVBRDFs) are the most common choice of models for this purpose, and the straightforward approach would be to fit the parameters of an SVBRDF model given the measurements; however, this does not generalize well to measurements of many real-world materials. Most BRDF models make strong assumptions about the function they represent, such as Helmholtz reciprocity, energy conservation etc.; furthermore, they assume that the shape of the reflectance lobes can be approximated accurately with the analytic expression of the BRDF (often based on Gaussians).

In practice, most real-world materials are far from perfectly defined and isolated, which changes the reflectance response drastically. They contain many imperfections (dust, scratches, fuzziness etc.) which humans are perceptive of and which contribute to a more realistic appearance. For many spatially varying materials, the measurements at single positions also contain traces of light transport from nearby points (such as subsurface scattering, interreflections, shadowing etc.). Therefore we follow the convention of Koudelka et al. [KBMK01] and refer to the texel responses as A(pparent)BRDFs or reflectance functions. To reconstruct these ABRDFs accurately, we need more expressive models that make fewer assumptions about the data.

One possible approach to tackle this issue is to learn not only the set of parameters, but also the reflectance model (see [RJGW19]). Using a neural architecture similar to an auto-encoder, reflectance functions are projected to a parameter vector in latent space. The decoder (analog to the BRDF model) is learned during training, along with the projection into latent space. Effectively, this means that no assumptions on the reflectance model are made. One of the main shortcomings of this approach however, is that it does not generalize well across materials. A different instance of the network is trained for every new material BTF, meaning that the parameter space is not shared between materials.

Ideally, we desire an infrastructure that is common to all materials, a way of encoding them all to the same space. This would make the neural reflectance models truly analog to analytic parametric BRDFs. In this paper, we present our new unified architecture that is trained on a wide range of BTF texels and projects them all to a common latent space, and investigate the flexibility, stability and robustness of such encoding. Furthermore, we demonstrate the practical advantage of such a unified architecture for efficient encoding of ABRDFs of a new, previously unseen, material.

2. Related Work

Bidirectional Texture Functions (BTFs) were first introduced by Dana et al. [DvGNK99]. Intuitively, they are organized as a stack of 2-dimensional textures, where each texture corresponds to the material's appearance under a certain combination of viewing and lighting directions. Relinquishing spatial information, one can look at individual texel responses as reflectance measurements of a single point on the material. In that sense, individual texel responses are a list of reflectance values at different angles on a particular position on the material defining its appearance function. Although considered individually, in the case of BTFs, these local responses still contain non-local lighting effects such as subsurface scattering or interreflection.

Fitting parametric models Researchers have often employed fitting of a parametric or analytic model to explain real-world reflectance data. Here, early work in the context of BTFs used polynomials [MGW01] and Lafortune lobes [MLH02] to model the directional dependence of each texel. Later approaches model directional variation as mixtures of parameteric models [WDR11, SPS13]. 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. [LBAD*06, MG09, AWL13]. Related to our process of learning a decoder, Genetic Programming (GP) has previously been employed to learn new analytic BRDF models that better describe specific materials [BLPW14]. Many recent works have proposed employing deep learning to efficiently fit a parametric BRDF model instead of employing traditional non-linear optimization [AAL16, LDPT17, LSC18, LXR*18, DAD*18, MHRK19, BL19, KXH*19]. Many of these methods are derived from U-Net or auto-encoder architectures [KCW*18, GLD*19].

Standard neural architectures usually have fixed numbers of inputs (in the non-convolutional dimensions), so recent developments have witnessed the use of size- and order-independent aggregation mechanisms, used for instance in the Generative Query Networks (GQNs) of [EJRB*18]. Deschaintres et al. [DAD*19] use a max-pooling operator to allow their SVBRDF estimation process to accommodate an arbitrary number of input photos of a material. Kim et al. [KGT*17] on the other hand use moment-pooling to obtain the same independence.

Unfortunately, analytic BRDF models are generally not sufficiently expressive to capture the rich variety of local reflectance behavior observed in real-world materials, which leads to significant residuals compared to the original data. Accordingly, 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.

Latent spaces of appearance Researchers have also investigated finding a common parameter space for real-world measurements of appearance. Soler et al. [SSN18] have proposed a non-linear manifold using a Gaussian-process latent-variable model that is suitable for interpolation over the space of measured materials (MERL database). Sun et al. [SJR18] have proposed a data-driven diffuse-specular separation which enables efficient material editing operations on the separated diffuse and specular components of measured BRDFs and a novel low-dimensional PCA model for measured BRDFs with similar dimensionality as analytic models. Lagunas et al. [LMS*19] instead learn a perceptual feature space for materials (based on data gathered from crowdsourced experiments) that correlates with perceived appearance similarity. In the context of their BTF compression scheme, Havran et al. [HFM10] extract common intrinsic data between materials, but in general there has been little work on finding a shared projection basis for BTFs.

Neural encoding of appearance Several recent works have applied neural networks to encode material representations or light transport in scenes ([RDL*15]). A comprehensive survey about deep appearance modelling can be found in [Don19]. 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] employ a neural network to render previews of materials with static scene geometry. Also related to this is neural texture rendering [TZN19], which features texture maps along with a neural renderer that allows to store much more information than diffuse textures, such as specular highlights and parallax. Kuznetsov et al. [KHX*19] use Generative Adversarial Networks to avoid explicit modeling and simulation of the surface microstructure, achieving a more flexible representation of specularity.

Closest to our work is that of Rainer et al. [RJGW19] who first proposed neural encoding of BTFs. However, they employ a material-specific autoencoder architecture that, while very good at encoding and interpolation of a specific material, does not generalize to other BTFs. This means that the entire network has to be expensively re-trained for encoding a different material BTF. The lack of a common parameter space for encoding ABRDFs also means that their method cannot support applications such as appearance synthesis or interpolation in latent space. Instead, we present a unified BTF encoding architecture that makes such latent space operations possible, while also making it very efficient to encode a previously unseen material without requiring any re-training.

3. Problem Analysis

BTFs are spatially-varying reflectance maps, meaning they also contain information about the spatial layout of ABRDFs. To keep input complexity low, we choose to ignore the spatial disposition and process each texel individually, without making use of the neighboring information. This means we encode each Apparent BRDF separately. The difficulty lies in the fact that ABRDFs describe a larger space of possible appearances than BRDFs. Since during the acquisition of the BTF, the point captured in the ABRDF is surrounded by the rest of the material, under directional lighting, the measurements contain a lot of non-local lighting effects, such as subsurface scattering and interreflections. Having these effects in the measurements allows for a more realistic rendering in the end, but it also makes individual treatment of texels more complex. This is one of the reasons why standard BRDFs, which by design model light transport in a single isolated point only, are not an optimal choice to approximate ABRDFs.

Another difficulty comes from the sample spacing of the measurements. ABRDFs are in practice a list of reflectance values with the corresponding light and view directions, for one position on the material. Depending on the acquisition protocol, the number of entries in that list, as well as the light/view directions that were sampled, is variable. Since we want to design an approach to encode any set of BTF measurements, no assumptions on angular resolution and sampling pattern can be made. The only prerequisite we impose on the input data, is that the hemispheres of lighting and viewing be sampled fairly uniformly and at a sufficient resolution to correctly sample most reflectance lobes. Fortunately, BTFs are usually sampled regularly in the angular domain, as there is little utility in adaptive sampling patterns for materials with spatial variations: a sampling strategy that is optimal for some set of points on the surface will be suboptimal for other points.

Since we want to learn the space of reflectance functions, beyond a mere mapping of ABRDFs to parameters, we model the entire process using neural networks. We refer the reader to [GBC16] for explanations of the deep learning concepts used in the following sections. Conceptually, our approach is close to an autoencoder ([HS06]): The input measurements are encoded to a vector in parameter space, which, when run through the decoding model, should approximate as well as possible the input values. In that sense, the decoding network is analog to a BRDF model, with the difference that the decoding function is learned rather than analytically fixed.

Neural networks are known to yield excellent performance for a range of challenging problems when the input data is arranged on a regular grid (e.g. a 2D image), and the network architecture relies on convolutional layers that effectively constitute a type of regularization strategy. When processing ABRDFs, such an approach is unfortunately not possible, since their angular positions are irregular. The number of angular samples may even change from one ABRDF to another, which means that another standard neural network element—the fully-connected layer—is also not admissible. To handle the unstructured angular nature of the data, we introduce a new architecture that is invariant to both the number of angular measurements as well as their exact positions and ordering.

4. Method

In this section, we delve into the specificity of our neural encoding and decoding method. An input is an ABRDF, which we format as a list of n 7-dimensional entries: incoming light direction (2 dimensions), outgoing light direction (2 dimensions), and respective RGB reflectance measurement (3 dimensions). Our encoding structure

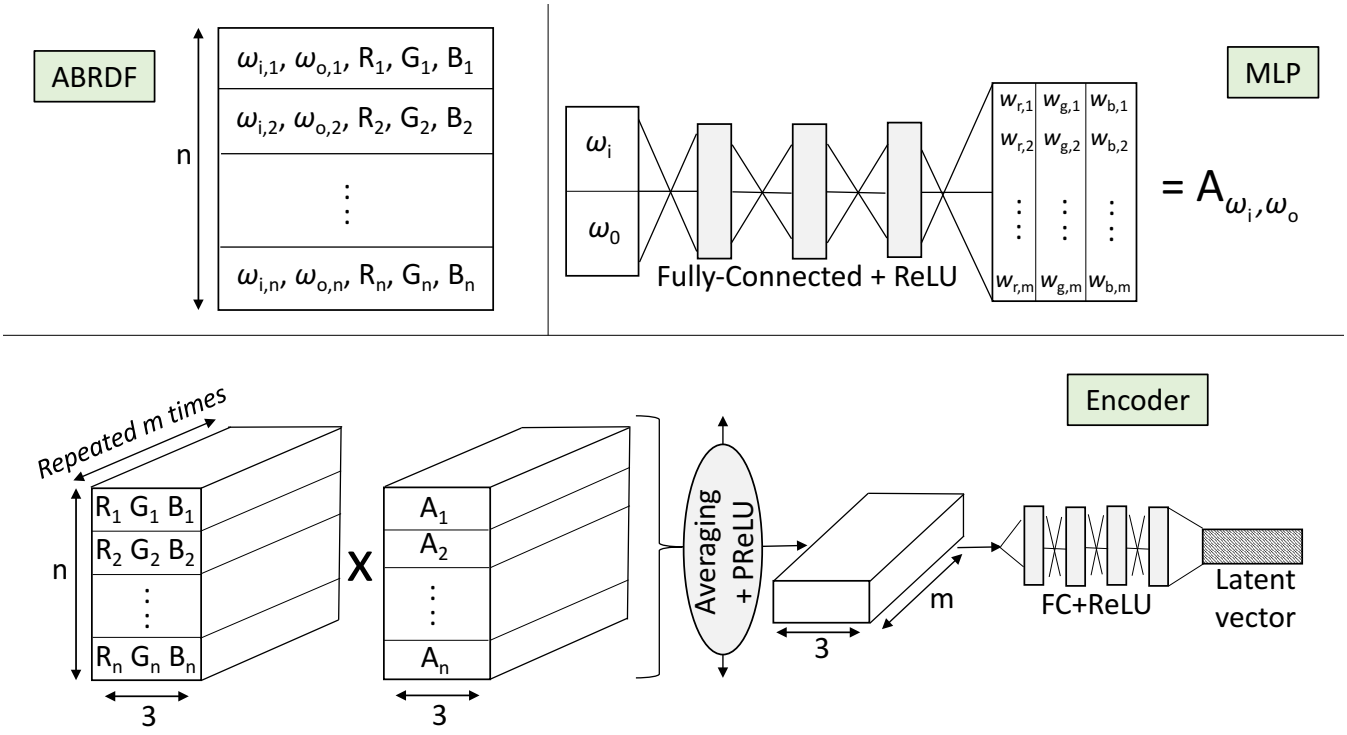

Figure 2: *Our architecture is conceptually an autoencoder for BTF texels, that works for any angular sampling resolution and pattern. It encodes input ABRDFs of arbitrary ordering and length n to a low-dimensional latent vector. Using an MLP that predicts weights from angles, we build a weights matrix that the expanded RGB measurements are multiplied with. Averaging across the vertical dimension allows us to recover a 3-by-m feature matrix for any input, satisfying the BTF sampling invariance criterion. Intuitively this is equivalent to discrete angular integration of the product of reflectance signal with angular filters. The remainder of our encoding architecture consists of standard fully-connected networks with ReLU activations.*

(Figure 2) projects the input to a latent vector of small, fixed dimensionality. The decoding structure (Figure 3) is able to reconstruct the input ABRDF given the corresponding latent vector. Similarly to an auto-encoder, the full encoding-decoding pipeline is optimized to best approximate an identity transformation (at the angles sampled in the input).

Neural Architecture Since we cannot make any assumptions about the input structure (in angular space), a flexible encoding pipeline is required. To satisfy this, we split the encoding network into two parts. In a first processing phase, a Multi-Layer Perceptron (MLP) outputs basis vectors of fixed dimensionality at each sampled angular position, which the reflectance measurements are projected on. Integration along the angular dimension reduces this to a fixed-size feature vector. In essence, this is a discrete approximation of an integration (in angular space) of the product of the reflectance lobes with learned filters.

The aim of this integration in the angular domain is to detect inherent properties of the reflectance functions through their responses to the filters. For instance, in the case where the filter learned by the MLP were to be constant, the recovered response would be the mean reflectance, which is a good approximation of diffuse albedo.

Another possibly more intuitive way of looking at our encoding approach is as an approximation of a linear layer. The most straightforward architecture would be a fully connected layer between the input list of RGB measurements and the latent vector. However, this is not possible because the ordering and number of angles in the input ABRDF can change between datasets. So instead, we use an MLP (parametrized on the light/view angles) to predict the weights that this fully connected layer would apply. Essentially, the MLP learns a continuous representation of the intended fully connected layer, in the angular domain.

Angular MLP The MLP takes the angles (in stereographic parametrization, similarly to [RJGW19]) of one light-view combination as input and returns a vector of weights. When processing a set of angular reflectance measurements, the angular MLP is run at every sampled light-view position, and we concatenate all *m*-dimensional output vectors into a weight matrix *A*. On the other side, the list of RGB reflectance values is expanded *m* times. Multiplying the resulting weight matrix elementwise with the list of reflectance values is then equivalent to a basic dot product of the angular filters with the reflectance signals.

Encoder Network The flexibility of our encoding architecture lies in the order- and resolution-invariant *integration* along the vertical dimension in Figure 2. While MaxPooling also satisfies the invari-

ance requirements, we determined empirically that an averaging operation produces better results. This allows us to squash the dimension of n elements to one, which means that independently of the ordering and the number of angular samples, the result of this operation is always a 3-by-m matrix. The output of this processing step is a feature vector of fixed dimensionality, that we use as input to the more traditional encoder.

Consistently with the aim of the angular MLP to simulate a linear layer, we first apply a non-linear activation (parametric ReLU (Rectified Linear Unit) & addition of bias) on the unrolled $3m$-dimensional feature vector. The remaining part of the the encoder is composed of standard fully connected layers with ReLU activations. In practice, the fully connected part of the encoder only contains one hidden layer before projection to the latent space.

Comparison of our encoder with Rainer et al. Rainer et al.'s [RJGW19] encoder performs 1-dimensional convolutions and downsampling (max pooling) over the ordered list of reflectance values, followed by fully connected layers with ReLUs. Because of this 1-dimensional treatment of data parameterized on 4 dimensions (incoming and outgoing directions), the learned mapping to the latent space has limited complexity. The reason for this convolutional downsampling is that the input ABRDFs are extremely large (almost 80,000 values), so directly using fully connected layers is impractical. Furthermore, a fully connected layer would require fixing the angular sampling of the ABRDFs, which works for Rainer et al. who train a new network per material. In our case, the angular sampling can change between datasets, so we want to remain flexible. To achieve this, our encoder learns a continuous representation of these fully connected weights based on the respective light/view angles with a small MLP (50,000 parameters), which allows us to create an approximate version of this fully connected layer at any given input size.

Decoder Network The decoder (Figure 3) is also a fully connected network with non-linear activations, following the same decoder design as Rainer et al [RJGW19]. It takes as input the latent coordinates of the ABRDF, along with the light and view directions in stereographic coordinates, which makes it practical for rendering. In practice, we use 4 hidden layers with ReLU activations.

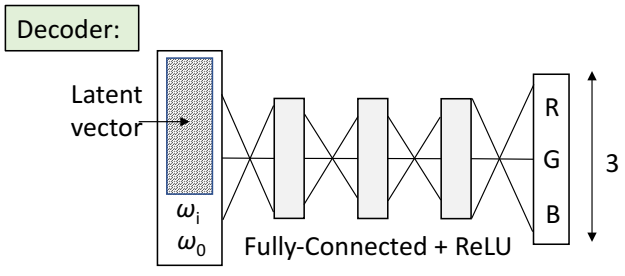

Figure 3: *Our decoder architecture is identical to [RJGW19].*

Training Specifications The entire architecture is trained end to end. To cheaply augment the data and to avoid overexposing the network to certain hues, we permute the RGB channels of input ABRDFs at every iteration. Furthermore, to make the network more robust to variations in the angular resolutions, the decoder receives a random subset of between 20% and 100% of the samples during training. The loss is still computed on the full set of angular samples, though. This ensures that even with a lower angular resolution, the projection still converges to the same position in latent space, and that the decoder interpolates smoothly between sampled angles.

We train on BTF texels from the Bonn BTF database [WGK14], which contains 7 material classes, each featuring 12 material BTF. We train and test on the texels of 11 out of 12 BTFs of each class. The 12^{th} material of each class is kept for validation and used for evaluation in the next section.

To keep training stable, the reflectance values in the ABRDFs are normalized in preprocessing, i.e., the mean gets subtracted and the resulting values are divided by their standard deviation. Additionally, to reduce the high dynamic range of measurements, a logarithmic transformation is applied to the values before the normalisation.

Once training is completed, compressing the appearance of a BTF texel simply becomes a matter of evaluating the network given the corresponding list of measurements as input. For rendering, only the projected latent maps and the decoder layers are required.

Implementation Details In our implementation, the angular MLP consists of 4 hidden layers, each with 128 neurons plus ReLU activations, and $m = 800$. The MLP hence outputs 3 vectors of 800 weights for each angular configuration of light/view, that are multiplied elementwise with each RGB reflectance measurement (expanded 800 times). The encoder only consists of a PreLU activation, one hidden layer with 128 neurons and a ReLU activation. The decoder consists of 4 hidden linear layers of 106 neurons with ReLU activations (same architecture as [RJGW19]). Whilst those parameters remain fixed, we explore several possibilities of latent space dimensionality in the following section.

We train with standard stochastic gradient descent, learning rate of 0.2, batch size of 10, 100 ABRDFs per dataset per epoch. At every epoch, we load a new random set of 100 ABRDFs from each material BTF. We train for 1000 epochs, which takes about 40 hours on average on an NVIDIA GeForce RTX 2070. We found empirically that using an L1 loss gives our encoding a more accurate average hue and preserves more contrast than the L2 loss used by Rainer et al.

5. Results

To assess the performance of our network, we visualize reconstruction results on the 7 datasets used for validation (unseen at training/testing time). We compare to the architecture from [RJGW19], which we consequently refer to as *custom network*, as Rainer et al. train a new instance of the network for every material. In contrast, we refer to our architecture as *general network* as it is trained on many different BTF datasets and evaluated on unseen materials.

5.1. Accuracy

When dealing with BTFs, it is difficult to compare with ground truth, because as soon as rendering is performed, the original textures are

(commonly linearly) interpolated, which introduces bias. The only available method is comparing reconstructions with the original textures at the angles that were sampled in the ABRDF. Figure 4 displays the texture reconstructions of each of the validation materials at one particular combination of light/view directions.

Comparisons in texture space In Figure 4, we compare the ground truth with the custom network of [RJGW19] trained on all Bonn training materials, except the validation materials displayed in the figure, to the custom network overfitted to the individual material, and to our network trained on all training materials. This is a skewed comparison in the sense that the networks of columns 2 and 5 are evaluated on unseen materials, while the network of column 3 was trained solely on the evaluated material.

Furthermore, we use the same latent space dimensionality and decoder size for all networks. This means they all dispose of the same compression and decoding budget, allowing us to assess how well each architecture is able to learn a more general embedding. As the custom network in column 3 is overfitted to the specific material, it represents the upper performance bound that a general architecture could reach.

On average, the custom network overfitted to the specific material performs slightly better than our network. However, this is to be expected as the custom network uses all its encoding budget to cater to the specific appearance of the material, while our network adopts an average solution that works well for all classes of materials. In that sense, our network performs almost as well on an unseen material as the custom network on an overfitted material, given the same parameter budget.

Overall, the main drawbacks of the encoding are firstly a loss of spatial detail (the reconstructions are slightly blurrier than the original). This is most likely due to slight misalignment or parallax in the original data, which means that individual positions on the material still move around in texture space, making the information harder to encode when we process ABRDFs individually. The other issue seems to be a damping of specular highlights for some materials (e.g., Fabric12). The most likely explanation for this is that specular highlights only show in a small subset of captured angles, making this part of the signal less crucial to the reconstruction loss. Diffuse albedo, anisotropy, intershadowing, etc., play a much bigger role in the loss than localized specular highlights.

Influence of the number of latent dimensions In order to tackle these loss-of-detail issues, we investigate the influence of latent space dimensionality, i.e., how much storage budget is given to the network to encode each ABRDF. More latent dimensions means more specific reflectance details can be encoded on each direction (anisotropy, specularity etc.) and the network is given more parameters to separate similar-looking texels.

Rainer et al. set a standard for a reasonable reconstruction performance. When overfitting the projection to a specific dataset, 8 latent dimensions are a good compromise between maximizing reconstruction accuracy and minimizing storage. We attempt to find the best compromise between a small network and similar performance.

Table 1 shows the average reconstruction error on the unseen datasets for our network at varying latent sizes. We compare our performance with that of Rainer et al.'s network. It is to be noted that we train our network on L1 loss on the logtransformed and whitened texels, while Rainer et al. train with an L2 loss on the texels preprocessed in the same way. From Table 1, we choose a latent size of 32 dimensions as the best compromise between accuracy and compression. The improvement gained by going to higher latent dimensions is marginal, and the network seems to have more difficulties generalizing for materials such as wood12, possibly overfitting to the training data. At 32 latent dimensions, our network achieves a reconstruction error lower or equal to the custom network on 4 out of 7 datasets. The latent maps are 4 times the size of Rainer et al.'s, but the decoder network is shared between all materials, so does not require extra storage per material.

	wood12	wallpaper12	stone12	leather12	felt12	carpet12	fabric12
[RJGW19](8)	**2.0**	1.5	**3.7**	**26**	2.5	4.1	**1.5**
Ours (8)	5.1	2.7	6.0	110	3.7	5.0	8.2
Ours (16)	4.5	1.9	4.9	101	2.7	3.3	6.7
Ours (32)	4.0	1.5	**3.7**	90	2.0	2.4	4.5
Ours (64)	4.1	1.5	**3.7**	98	**1.8**	**2.2**	4.0
Ours (128)	4.5	**1.4**	**3.7**	96	**1.8**	**2.2**	4.3

Table 1: *Mean Square Reconstruction error ($\times 10^4$) for networks with varying latent size. Datasets unseen by our network during training.*

Regarding the size of the decoder, Rainer et al. showed that 4 hidden layers represent the sweet spot in depth. We experimented with wider layers (more neurons), which also decreases the reconstruction error, but in a far less drastic measure than increasing the latent size. Additionally, it makes the network heavier and slower to train, as well as much slower to run for rendering applications. For this reason, we consider a latent size of 32 with the original layer width of 106 neurons to be the most efficient compromise for maximized performance at reasonable compression: this amounts to storing roughly 10 RGB textures for one encoded BTF material.

Comparisons on renderings When rendering with the original dataset, the ground truth renderings are inevitably corrupted by linear interpolation between nearest sampled directions (usually 9 light/view combinations). Many very localized specular details can get lost or blurred out in the process. Hence, some additional reconstruction accuracy can be gained with networks that interpolate well in between the original sampled views, even if at the originally sampled positions (as in Figure 4) the reconstruction is not perfect. The stability of the neural interpolation was demonstrated in [RJGW19]: with superior interpolation capabilities, even if the reconstruction performance at originally sampled directions is not perfect, we still obtain a near-equal quality performance on renderings.

In Figure 5, we compare renderings with the custom overfitted network and with two instances of our architecture, to renderings with the original BTF. Both full BTF and neural rendering use the same rendering code as [RJGW19]. For reference, renderings of the BTF-mapped cylinders in Mitsuba, at 800×800 pixels, at 32 samples per pixel, parallelized on 10 processes, pathtraced with parallax mapping of the height field associated to the material, take 1.2 minutes on our machine for our general network with 32 latent dimensions, versus 1.1 minutes with the custom network of [RJGW19] at 8 latent dimensions.

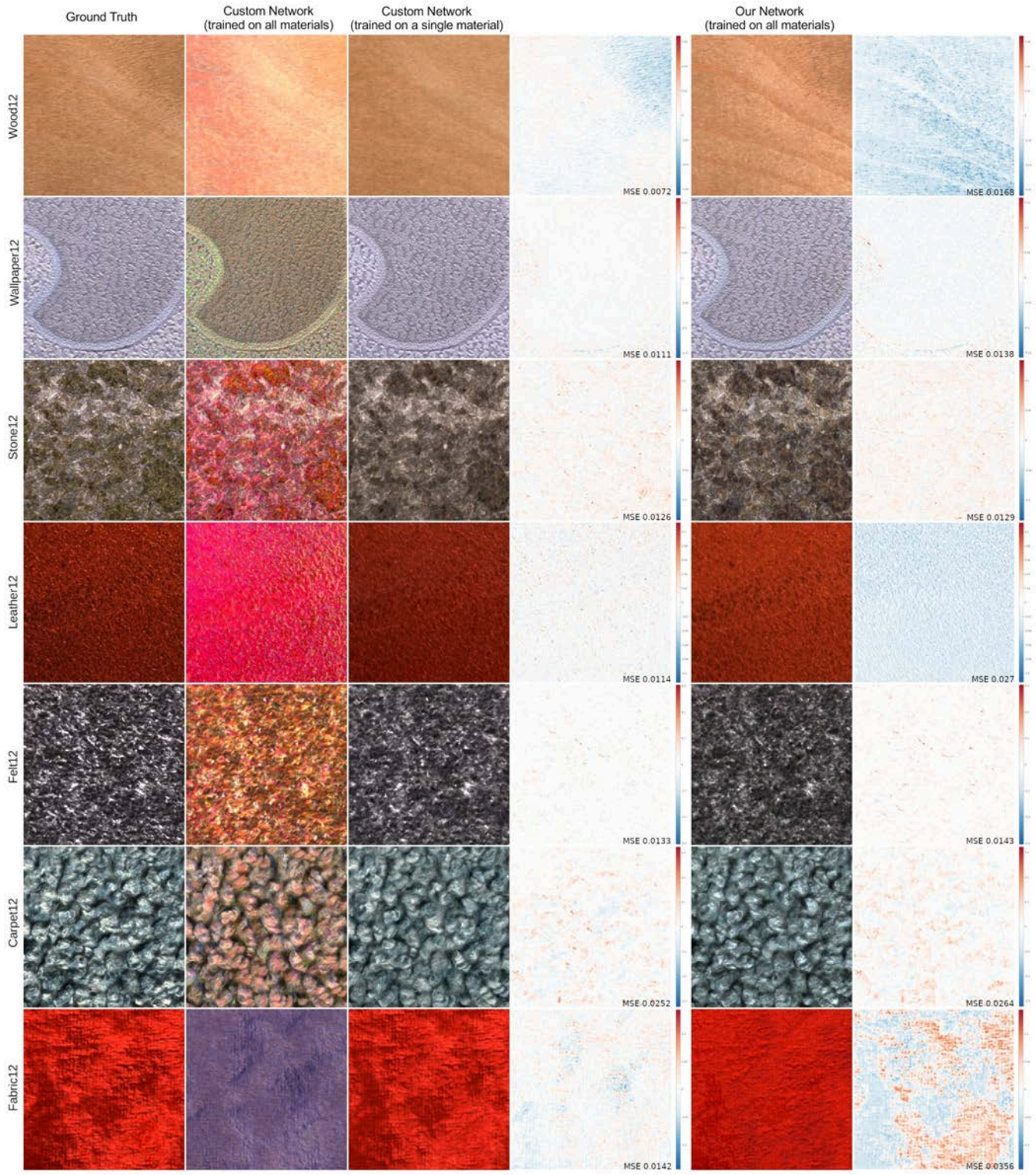

Figure 4: *Comparison of the reconstruction using different architectures (at equal latent space and decoder size) with the original texture from BTF datasets kept for validation. Except for the custom network [RJGW19] overfitted specifically to the respective dataset, the other networks have not seen the data at training time. View/light azimuth and elevation angles of the shown textures: 0°, 45°, 90°, 30°.*

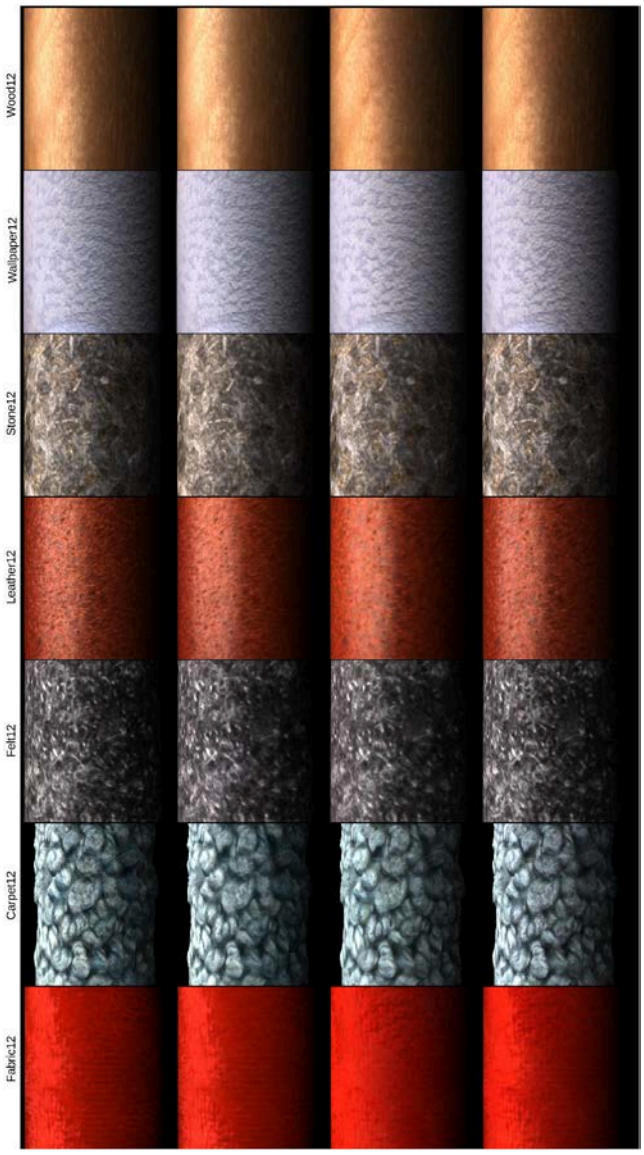

Figure 5: *Renderings,* from left to right: *full BTF, [RJGW19] network (overfitted to the material), our network (unseen material) with latent size 8 and 32.*

In the third column we use our architecture with 8 latent dimensions and decoder layers of 106 neurons (same budget as [RJGW19]). In the fourth column, we show our model of choice, with 32 latent dimensions this time. The increase in encoding budget greatly improves the reconstruction, even though all the materials displayed are unseen by our network at training times (reserved for validation). This means our codec generalizes well outside of the training set.

It is apparent that our network performs very well at encoding spatial detail (most noticeable on the leather12 dataset), better than the custom overfitted network. Non-local effects like subsurface scattering and intershadowing are particularly well replicated by our architecture. However, for materials with sharp specular lobes (see wood12 and fabric12), some of the specular highlights are damped. In this area, the custom network remains slightly more faithful, albeit applying more spatial blur. This is most likely due to the custom decoder learning a specific reflectance shape that is characteristic to the overfitted material's texels. Our network however, has to learn an average reflectance shape across many materials with very varied appearance. Specular highlights proportionally only play a small role as the appearance of most materials in the database is dominated by other properties (diffuse albedo, intershadowing, etc.). For visualization of temporal coherence, we provide animations of a BTF-textured plane under a moving point light source in the supplemental material, comparing the original BTF rendering to the custom network and our general network.

Evaluation on a different data source For additional comparisons, we process the first ten datasets of the UTIA MAM database [FKH*18] with our network (trained and tested solely on the Bonn datasets). The UTIA BTFs, too, are uniformly sampled, but are less dense, containing 6,561 angular measurements compared to 22,801 for Bonn. However the materials are generally more specular and some of them contain transparent layers. To our knowledge, there is no heightfield parallax correction. Figure 6 show point-light renderings of the first ten materials of the UTIA MAM database, rendered using the original BTF, the custom network of Rainer et al. [RJGW19] overfitted to the respective material, and our network's predicted latent maps. The most notable difference is that our network tends to dim the specular highlights that are truncated in the original dataset, because it has not seen any examples of those in the training data, whereas the overfitting network [RJGW19] can learn those. Overall, we observe the expected slight degradation compared to Rainer et al., but beyond that, our network generalizes plausibly to a new, unseen, data source.

5.2. Stability of the latent space

Robustness to angular resolution We show that our novel architecture can accommodate any structure of input sampling. To enforce this flexibility, at training time, we randomly feed between 20% and 100% of the angular inputs to the angular MLP and encoder. For reference, the 6,561 angular measurements of the UTIA datasets amount to approximately 28% of the angular resolution of the Bonn datasets. The loss is still computed at all originally measured angular combinations, to make sure the model learns correct interpolation. Using the validation datasets, we show the convergence of the reconstruction loss as a function of the size of the angular subset in Figure 7. From less than 10% of angular samples on (2,280 randomly chosen out of the original 22,801), the reconstruction is stable and has converged.

Figure 8 shows the same comparison on reconstructed textures. Each texture was rendered using a latent map that was projected from a random subset (of the respective proportion) of angular samples. Visually and quantitatively, the reconstructed appearance converges at less than 10% of the original samples. Since the network only needs a tenth of measurements of the Bonn datasets for similar reconstruction quality, this would allow capture times of these datasets to be reduced tenfold. Even datasets with a low-resolution angular sampling will benefit from this encoding that creates an appearance

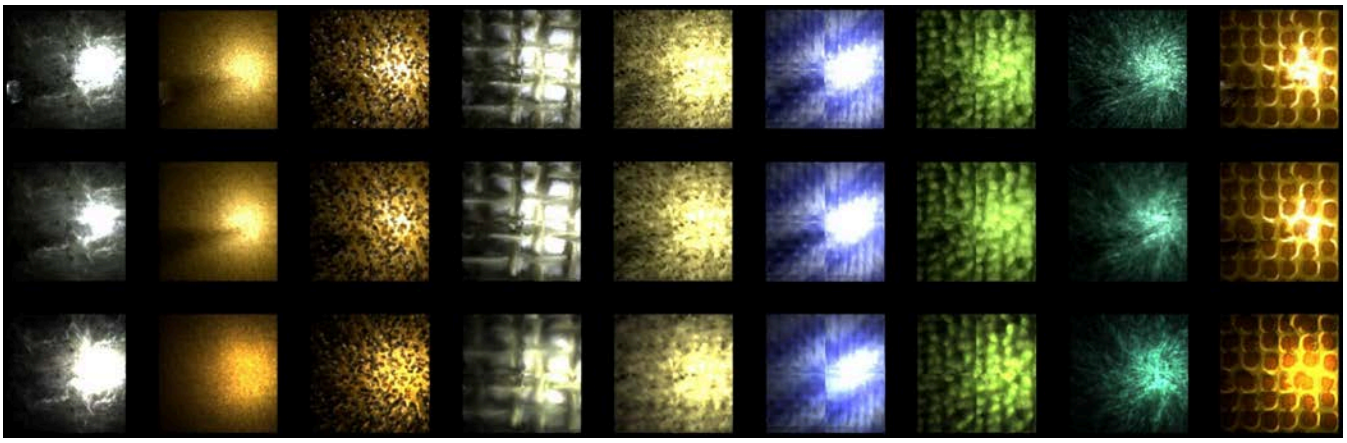

Figure 6: *Point-lit renderings of BTF datasets from [FKH*18].* From left to right: *materials 1 to 10.* Top to bottom: *Original BTF, [RJGW19] (overfitted), our reconstruction (unseen material).*

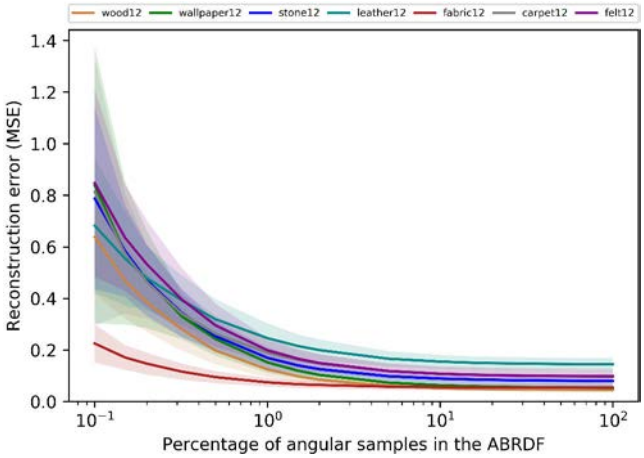

Figure 7: *Mean and standard deviation of the reconstruction loss as a function of percentage of angular samples, computed over a random selection of pre-processed ABRDFs for each validation material.*

model that interpolates smoothly and behaves well using the knowledge it gained from the densely sampled training materials, which will produce better renderings than interpolating a sparsely sampled set of angles.

Filtering in latent space An important technique to facilitate rendering at different scales is spatial downfiltering of textures, commonly applied as mipmapping. Texture pyramids are typically precomputed before rendering, to allow texture lookups at different resolutions depending on the footprint of the material in the rendering. We investigate the equivalence between downsampling in latent space before renderings, versus downsampling in reconstructed texture space in Figure 9. The difference is barely noticeable, filtering in latent space proves to be very stable. Furthermore, this saves computation time as it allows precomputation of latent mipmaps and avoids having to run the decoder multiple times.

Texture synthesis using the most compressed representation Finally, a stable latent space allows for efficient BTF synthesis. Texture synthesis on BTFs is challenging because BTFs are basically N-dimensional RGB textures, N being the number of angular measurements. Using our encoding, we can directly synthesize from the latent maps instead (32-dimensional). We use straightforward image quilting ([EF01]) on the latent maps in Figure 10 to enlarge the BTF by a factor of 5.

Visualization of the latent space We use a t-distributed Stochastic Neighbor Embedding (t-SNE) to visualize the behavior of the latent projection (see Figure 1). We observe that within a BTF, the projections of texels are very well clustered. There is however not an obvious consistency with the semantic classes defined in the Bonn database. Nevertheless, this is to be expected as our clustering is relative to appearance, while the semantic classes refer rather to fabric/material types and fabrication procedures. For instance, some of the carpet ABRDFs are very close in diffuse albedo and reflectance shape to those from felt materials, even though they are in different semantic classes.

6. Conclusion

We presented a novel architecture model capable of handling unstructured angular reflectance measurements. Based on autoencoders, our network projects ABRDFs into a low-dimensional latent space, analogous to analytic BRDF model parameters, while the decoder network is analogous to the analytic BRDF model expression. The network is trained on a variety of ABRDF samples from the Bonn BTF datasets, and evaluated on previously unseen materials. Compared to Rainer et al. who train a new network instance for every new material [RJGW19], encoding a new material with our method requires a simple evaluation of the encoder. Having a single autoencoder instance also means that the latent space is shared between materials, i.e., texels are projected into the same domain. This allows for exploration of the parameter space for applications such as latent mipmapping, texture synthesis etc.

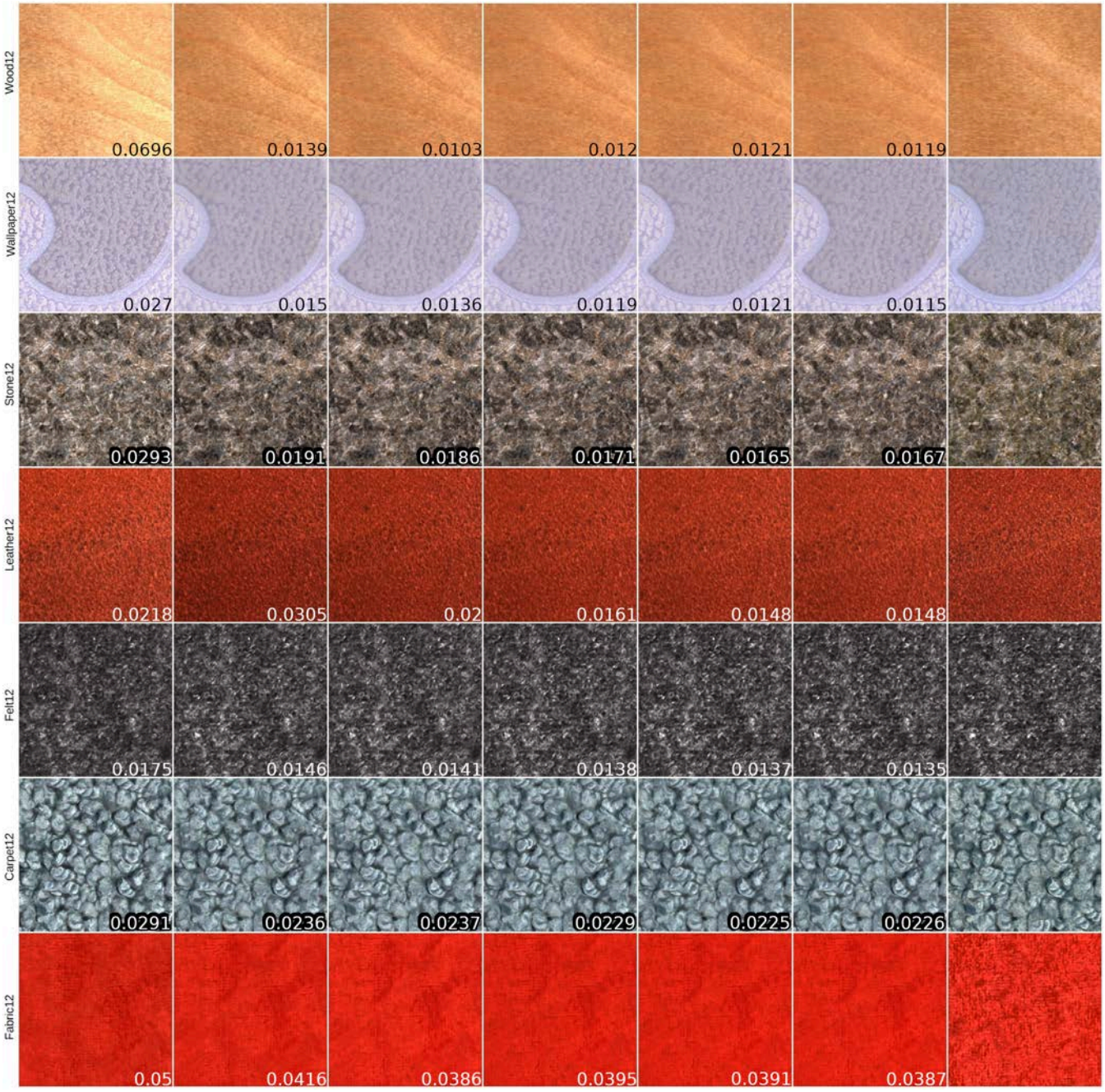

Figure 8: From left to right: *Reconstruction using our network (latent size 32) at 1, 5, 10, 20, 50 and 100% of angular samples, with average reconstruction error.* Rightmost column: *Ground truth.* ***Angles*** *(view azimuth, elevation, light azimuth, elevation): 0°, 45°, 0°, 90°.*

Future Work One of the main obstacles hindering more generalization seems to be the lack of data. The Bonn database is the biggest available BTF database, but it is still relatively sparse compared to standard deep learning problems. Only 77 materials are shown to the network, and the validation datasets we evaluate on do not all have close matches in the training set. One way of tackling this issue would be to augment the training data with synthetic ABRDFs. This is a tricky endeavor, however, as most synthetic SVBRDF datasets are generated using common analytic BRDF models. This could bias the network into learning current analytic models, which conflicts with our goal of staying flexible to learn all the components of real-world reflectance functions.

Acknowledgments

We would like to acknowledge the EPSRC grant EP/K023578/1 and the EPSRC Early Career Fellowship EP/N006259/1. We would also

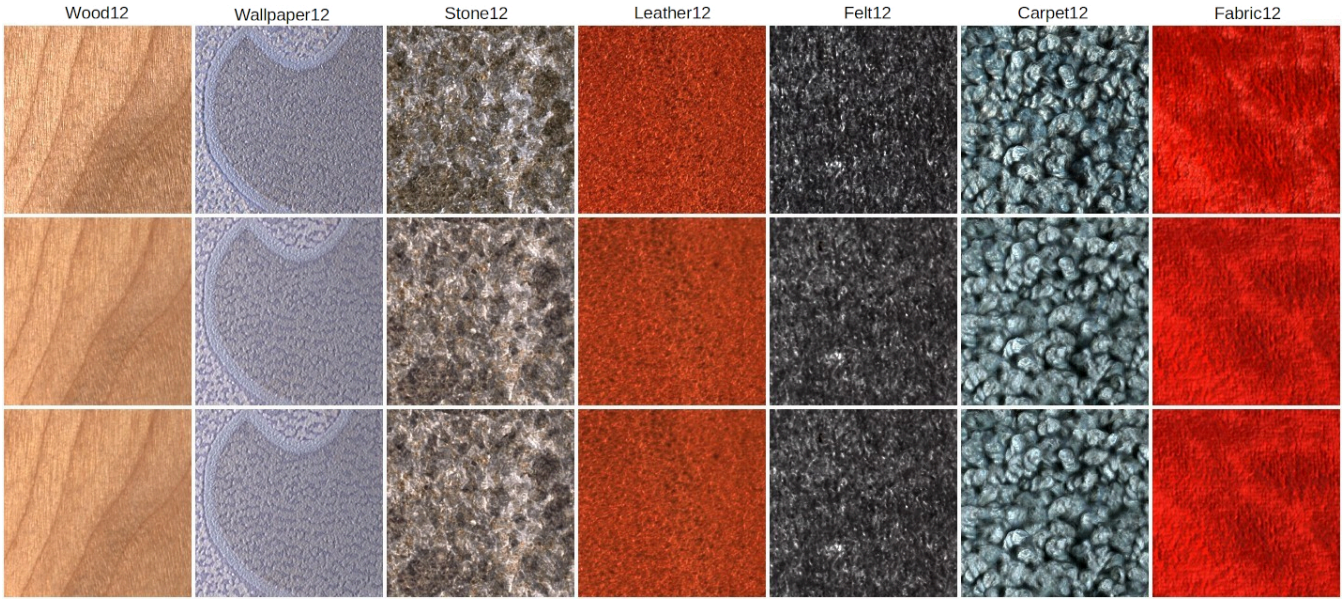

Figure 9: Rows (top to bottom): *Ground truth, reconstruction filtered in image space, and reconstruction filtered in latent space.* ***Angles*** *(view azimuth, elevation, light azimuth, elevation): 0°, 90°, 270°, 30°.*

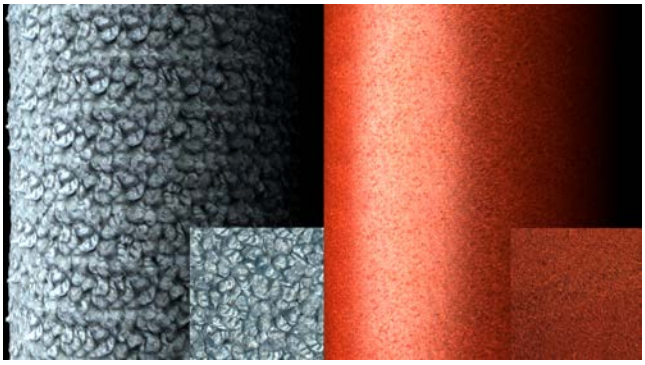

Figure 10: *Renderings with textures synthesized in latent space.* Bottom corners: *Ground truth BTF texture.*

like to thank the University of Bonn's Computer Graphics group for making their BTF database publicly available.

References

[AAL16] AITTALA M., AILA T., LEHTINEN J.: Reflectance modeling by neural texture synthesis. *ACM Trans. on Graphics (Proc. SIGGRAPH) 35*, 4 (July 2016), 65:1–65:13. URL: http://doi.acm.org/10.1145/2897824.2925917, doi:10.1145/2897824.2925917. 2

[AWL13] AITTALA M., WEYRICH T., LEHTINEN J.: Practical SVBRDF capture in the frequency domain. *ACM Trans. on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 110:1–110:12. 2

[BL19] BOSS M., LENSCH H. P. A.: Single image brdf parameter estimation with a conditional adversarial network, 2019. arXiv:1910.05148. 2

[BLPW14] BRADY A., LAWRENCE J., PEERS P., WEIMER W.: genBRDF: Discovering new analytic BRDFs with genetic programming. *ACM Trans. Graph. 33*, 4 (July 2014), 114:1–114:11. URL: http://doi.acm.org/10.1145/2601097.2601193, doi:10.1145/2601097.2601193. 2

[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. doi:10.1145/3197517.3201378. 2

[DAD*19] DESCHAINTRE V., AITTALA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Flexible svbrdf capture with a multi-image deep network. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 38*, 4 (July 2019). URL: http://www-sop.inria.fr/reves/Basilic/2019/DADDB19. 2

[Don19] DONG Y.: Deep appearance modeling: A survey. *Visual Informatics 3*, 2 (2019), 59 – 68. URL: http://www.sciencedirect.com/science/article/pii/S2468502X1930035X, doi:https://doi.org/10.1016/j.visinf.2019.07.003. 3

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (Jan. 1999), 1–34. URL: http://doi.acm.org/10.1145/300776.300778, doi:10.1145/300776.300778. 2

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 341–346. URL: http://doi.acm.org/10.1145/383259.383296, doi:10.1145/383259.383296. 9

[EJRB*18] ESLAMI S., JIMENEZ REZENDE D., BESSE F., VIOLA F., MORCOS A., GARNELO M., RUDERMAN A., RUSU A., DANIHELKA I., GREGOR K., REICHERT D., BUESING L., WEBER T., VINYALS O., ROSENBAUM D., RABINOWITZ N., KING H., HILLIER C., BOTVINICK M., HASSABIS D.: Neural scene representation and rendering. *Science 360* (06 2018), 1204–1210. doi:10.1126/science.aar6170. 2

[FKH*18] FILIP J., KOLAFOVÁ M., HAVLÍČEK M., VÁVRA R., HAINDL M., H. R.: Evaluating Physical and Rendered Material Appearance. *The Visual Computer (Computer Graphics International 2018)* (2018). 8, 9

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep Learning*. MIT Press, 2016. www.deeplearningbook.org. 3

[GLD*19] GAO D., LI X., DONG Y., PEERS P., XU K., TONG X.: Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Trans. Graph. 38*, 4 (July 2019). URL: https://doi.org/10.1145/3306346.3323042, doi:10.1145/3306346.3323042. 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. doi:10.1111/j.1467-8659.2009.01585.x. 3

[HS06] HINTON G., SALAKHUTDINOV R.: Reducing the dimensionality of data with neural networks. *Science (N.Y.) 313* (08 2006), 504–7. doi:10.1126/science.1127647. 3

[KBMK01] KOUDELKA M. L., BELHUMEUR P. N., MAGDA S., KRIEGMAN D. J.: Image-based modeling and rendering of surfaces with arbitrary BRDFs. In *Proc. IEEE Conf. Comp. Vision & Pat. Rec. (CVPR)* (Dec. 2001), pp. I–568–I–575. doi:10.1109/CVPR.2001.990524. 2

[KCW*18] KANG K., CHEN Z., WANG J., ZHOU K., WU H.: Efficient reflectance capture using an autoencoder. *ACM Trans. Graph. 37*, 4 (July 2018). URL: https://doi.org/10.1145/3197517.3201279, doi:10.1145/3197517.3201279. 2

[KGT*17] KIM K., GU J., TYREE S., MOLCHANOV P., NIESSNER M., KAUTZ J.: A lightweight approach for on-the-fly reflectance estimation. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 20–28. 2

[KHX*19] KUZNETSOV A., HAŠAN M., XU Z., YAN L.-Q., WALTER B., KHADEMI KALANTARI N., MARSCHNER S., RAMAMOORTHI R.: Learning generative models for rendering specular microgeometry. *ACM Transactions on Graphics 38* (11 2019), 1–14. doi:10.1145/3355089.3356525. 3

[KXH*19] KANG K., XIE C., HE C., YI M., GU M., CHEN Z., ZHOU K., WU H.: Learning efficient illumination multiplexing for joint capture of reflectance and shape. *ACM Trans. Graph. 38*, 6 (Nov. 2019), 165:1–165:12. URL: http://doi.acm.org/10.1145/3355089.3356492, doi:10.1145/3355089.3356492. 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. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 735–745. URL: http://doi.acm.org/10.1145/1179352.1141949, doi:10.1145/1179352.1141949. 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. doi:10.1145/3072959.3073641. 2

[LMS*19] LAGUNAS M., MALPICA S., SERRANO A., GARCES E., GUTIERREZ D., MASIA B.: A similarity measure for material appearance. *ACM Trans. Graph. 38*, 4 (July 2019), 135:1–135:12. URL: http://doi.acm.org/10.1145/3306346.3323036, doi:10.1145/3306346.3323036. 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). 2

[LXR*18] LI Z., XU Z., RAMAMOORTHI R., SUNKAVALLI K., CHANDRAKER M.: Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018 Technical Papers* (2018), ACM, p. 269. 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. doi:10.1145/1053427.1053458. 2

[MG09] MENZEL N., GUTHE M.: g-BRDFs: an intuitive and editable BTF representation. *Computer Graphics Forum* (2009). doi:10.1111/j.1467-8659.2009.01432.x. 2

[MGW01] MALZBENDER T., GELB D., WOLTERS H.: Polynomial texture maps. *Proc. SIGGRAPH* (2001), 519–528. doi:10.1145/383259.383320. 2

[MHRK19] MERZBACH S., HERMANN M., RUMP M., KLEIN R.: Learned fitting of spatially varying BRDFs. *Computer Graphics Forum 38*, 4 (2019), 193–205. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13782, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13782, doi:10.1111/cgf.13782. 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

[MRF18] MAXIMOV M., RITSCHEL T., FRITZ M.: Deep appearance maps, 2018. arXiv:1804.00863. 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. doi:10.1145/2766899. 3

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural BTF compression and interpolation. *Computer Graphics Forum (Proc. Eurographics) 38*, 2 (2019), 1–10. 2, 3, 4, 5, 6, 7, 8, 9

[SJR18] SUN T., JENSEN H. W., RAMAMOORTHI R.: Connecting measured BRDFs to analytic BRDFs by data-driven diffuse-specular separation. *ACM Trans. Graph. 37*, 6 (Dec. 2018), 273:1–273:15. URL: http://doi.acm.org/10.1145/3272127.3275026, doi:10.1145/3272127.3275026. 3

[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. doi:10.1016/j.cag.2013.06.006. 2

[SSN18] SOLER C., SUBR K., NOWROUZEZAHRAI D.: A versatile parameterization for measured material manifolds. *Computer Graphics Forum 37*, 2 (2018), 135–144. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13348, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13348, doi:10.1111/cgf.13348. 3

[TZN19] THIES J., ZOLLHÖFER M., NIESSNER M.: Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics 2019 (TOG)* (2019). 3

[WDR11] WU H., DORSEY J., RUSHMEIER H.: A sparse parametric mixture model for BTF compression, editing and rendering. *Computer Graphics Forum 30* (2011), 465–473. URL: http://dx.doi.org/10.1111/j.1467-8659.2011.01890.x, doi:10.1111/j.1467-8659.2011.01890.x. 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. 1, 5

[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. doi:10.1145/3197517.3201307. 3