Acquisition, Encoding and Rendering of Material Appearance Using Compact Neural Bidirectional Texture Functions

Gilles Rainer

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

March 14, 2021

I, Gilles Rainer, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis addresses the problem of photo-realistic rendering of real-world materials. Currently the most faithful approach to render an existing material is scanning the Bidirectional Reflectance Function (BTF), which relies on exhaustive acquisition of reflectance data from the material sample. This incurs heavy costs in terms of both capture times and memory requirements, meaning the main drawback is the lack of practicability.

The scope of this thesis is two-fold: implementation of a full BTF pipeline (data acquisition, processing and rendering) and design of a compact neural material representation.

We first present our custom BTF scanner, which uses a freely positionable camera and light source to acquire light- and view-dependent textures. During the processing phase, the textures are extracted from the images and rectified onto a unique grid using an estimated proxy surface. At rendering time, the rectification is reverted and the estimated height field additionally allows the preservation of material silhouettes.

The main part of the thesis is the development of a neural BTF model that is both compact in memory and practical for rendering. Concretely, the material is modeled by a small fully-connected neural network, parametrized on light and view directions as well as a vector of latent parameters that describe the appearance of the point. We first show that one network can efficiently learn to reproduce the appearance of one given material.

The second focus of our work is to find an efficient method to translate BTFs into our representation. Rather than training a new network instance for each new

material, the latent space and network are shared, and we use an encoder network to quickly predict latent parameter networks for new, unseen materials.

All contributions are geared towards making photo-realistic rendering with BTFs more common and practicable in computer graphics applications like games and virtual environments.

Impact Statement

In this thesis, we present our custom material appearance scanning device and the associated data processing pipeline, which we disseminated at a conference. We also propose two novel approaches to model the scanned appearance in a compact manner and to render it efficiently on arbitrary surfaces. Both approaches have been presented at Eurographics, one of the premier venues in Computer Graphics research, and published in the journal Computer Graphics Forum. The proposed method uses elements from Machine Learning, in particular the emerging deep neural networks, and pioneers their deployment in this domain of Computer Graphics. We envision that this will spark a lot of further research in neural appearance models.

Outside of academia, our model allows efficient photo-realistic rendering of complex materials such as fabrics, which can have many applications in the games and movie industry, as well as in fashion, or in any other domain that relies on predictive rendering and realistic visual previews of materials, such as architecture or automotive design, for instance.

Acknowledgements

First and foremost, I sincerely thank my supervisor Tim Weyrich for his mentoring and support throughout my thesis. This project would not have existed without his initiative, and would not have come to fruition without his expert guidance. I also thank Change of Paradigm Ltd. for funding parts of this project, and Philip Delamore for his keen interest and advice from a fashion design point of view. Many thanks to my second supervisor, Tobias Ritschel, and to my co-authors: Abhijeet Ghosh, Wenzel Jakob, Kevin Rathbone. Finally, I would also like to thank my examiners, Niloy Mitra and Reinhard Klein.

My deep thanks also go to all my friends at UCL, for making the office a friendly and familiar environment. Adam Sherwood, Alejandro Sztrajman, Joep Moritz, Martin Ruenz, Stratos Skordos, Denis Tome, Thomas Tanay, Jerone Andrews, and many others, made the days fly by quicker than I thought they would.

This thesis is dedicated to everyone I consider family, for supporting me unconditionally in tough times, but also for pushing me to see things through. My deepest thanks to my parents, as well as to Cecile, and Thomas and Micha.

This PhD allowed me to meet many great people during the years, inside and outside of academia, so these acknowledgments also go out to everyone who was not named here explicitly, but with whom I shared good times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans experience the world around them through their eyes in a first instance, yet human vision is a very subjective phenomenon. A picture speaks a thousand words and provides intuitive information about materials, objects, and their spatial relationship, which motivates the search for techniques to produce realistic images of scenes.

Technological advances have provided two complementary tools to create such visuals: Photography allows us to take measurements of the appearance of existing real-world scenes. Computer Graphics, on the other hand, allow us to synthesize images of virtual scenes. Physically-based rendering can be seen as a virtual simulation of photography: the transport and diffusion of light is computed using complex mathematical models and algorithms, and measured by a virtual camera sensor to create the final image. In that sense, rendering is the virtual counterpart to photography. With enough real-world data, it is then possible to tune the models used to render and achieve accurate renderings of different settings and new scenes.

While general rendering quality in Computer Graphics often reaches photo-realistic appearance, there are still many materials that pose a big challenge and can only be reproduced faithfully at the cost of a very high memory footprint and computational resources. One example of materials with extremely complex appearance are fabrics, as they combine most of the more challenging effects into one class of materials. This project was born out of the needs of an industrial partner

focused on developing a capability for photo-realistic previews of clothing. The collaboration highlighted the need for a new state-of-the-art representation that addresses memory and practicability, e.g. for online visualizations and sharing of rendering assets.

This thesis tackles how to both efficiently and faithfully represent and simulate the appearance of real-world materials that exhibit complex reflectance behaviors, with a particular focus on fabrics.

1.1 Common Material Representations

There are two main classes of techniques that approach the problem in completely opposed ways. First-principles modeling uses the physical laws as a starting point to derive the rendering from, while data-driven methods start with the observed result and re-use it as an approximate appearance prior for rendering.

First-principles approaches (see Section 2.3) aim for a physically correct simulation down to the finest parts of the problem. In the case of fabrics, this would mean attempting to simulate light transport as accurately as possible, down to the fiber level. While such a simulation exhibits high faithfulness to the real-world situation, it requires large resources, both in the memory required to store the complexity of the model and in terms of computational efforts. Even a small piece of clothing is already composed of millions of fibers, which all interact with the light – rays will perform many bounces inside the arrangement of fibers before shooting out and towards the viewer. Inspired by hair rendering, a significant number of early approaches ([KK89, MM06, ZW07, ZYWK08]) focus on finding explicit reflectance functions for fibers, yarns, or compound yarn arrangements, with parameters closely tied to physical or optical properties. Luan et al. [LZB17] alleviate some of the storage requirements by generating fiber geometry on-the-fly and the renderings demonstrate stunning realism. Zhao et al. ([ZJMB12, ZJMB14]) also rely on accurate physical simulation of small-scale structures, but instead of basing their method on the fiber model, they use volumetric density and orientation fields, which can be measured and validated from real-world fabrics using micro CT scans.

On the other side, *data-driven techniques* (see Section 2.2) are less concerned with accurate simulation of complex light transport phenomena, and instead directly use the data from the measurements, usually photographs of the original material. Rather than measuring small-scale material properties and feeding those to an expensive light transport simulation, the resulting appearance of the material is measured in a variety of conditions and directly retrieved in renderings. In a way, image-based material rendering can be viewed as a virtual collage of patches from photographs onto the virtual object geometry. This is far simpler to model and gradually becoming more convenient to acquire (as low-cost cameras are omnipresent in cellphones for instance), but the accuracy directly depends on the quality and number of measurements.

Pushing the concept of image-driven material modeling to the extreme limit would mean taking an exhaustive set of measurements of a material's appearance. This corresponds to capturing a dense *Bidirectional Texture Function* (BTF), as introduced by Dana et al. [DvGNK99]. A camera and a directional light source are placed at various positions on the hemispheres and capture calibrated photographs of a planar material sample. The denser the sampling of light and view hemispheres, the higher the number of textures in the BTF, and the more accurate the reproduction of the original appearance at rendering time. In this thesis, we choose to use BTFs as they provide great flexibility in the range of materials they can faithfully represent, and this flexibility is necessary to render materials with as varied appearances as fabrics for instance.

1.2 Rendering Fabrics

Fabrics are an example of materials that have always fascinated artists through their diverse appearance. Their wide visual variety (see Figure 1.1) results from the complex fabrication process, where many alternative steps are available at each stage of the process. Each choice will result in a slightly modified appearance of the final fabric.

Essentially, a piece of textile can be described at three natural scales: fibers,

Figure 1.1: Fabrics encompass a range of materials with very diverse appearance.

yarns and the final yarn-composition. Fibers are the elementary building blocks of yarns. Hundreds of fibers (micro-scale diameter) are grouped to produce plies, which in turn are twisted together to produce yarns. Depending on the ply arrangement, yarns will have different structures and appearances. For woven cloths, different weaving patterns of yarns (Figure 1.2) will produce varied visual results. Other types of textiles such as felt are produced by heating and pressing fibers together, which produces another distinct type of appearance.

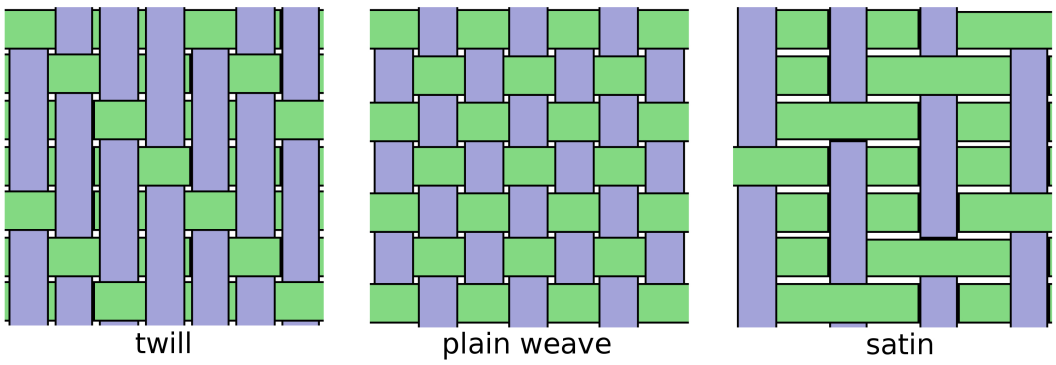


Figure 1.2: Different weave patterns creating different textiles (illustration taken from [IM06]).

From a Computer Graphics point of view, there is no obvious straightforward approach to modeling fabrics. Fibers can be intertwined in virtually any arrangement, which makes it complicated to simplify the spatial pattern with an analytical expression. The fiber type and weave strongly impact the final appearance, which can be very diverse, making it difficult to design a straightforward appearance model

that generalizes to all textiles.

Figure 1.3: Close-up of fly-away fibers on corduroy.

Additionally, the human eye is conditioned to perceive a degree of softness when looking at fabrics, relying on different visual cues. One of these is created by fly-away fibers. These are short fiber ends that stick out of the fabric surface in a fairly random fashion, making the silhouette look softer as the surface edge is blended out. The randomness and complexity of existing fly-away fiber distributions is very challenging to simulate and match realistically, most fabric renderings will look either too hard or solid, and too clean. Legendre et al. [LHWD17] tackled a similar topic, the simulation of asperity scattering in vellus hairs on silhouettes of human faces, which produces the same visual softness effect as fly-away fibers on fabrics. Although the results are visually pleasing and the realism is increased, the synthesized hairs look quite regular and give a *rendered impression*, which is something we would like to avoid at all costs. Throughout this thesis, we will often use fabrics as materials to evaluate our methods on, as they exhibit many of these complex appearance behaviors that are difficult to simulate and that our eyes are very accustomed to evaluating. This makes fabrics great materials for qualitatively

assessing the photo-realism of renderings, that is whether they look as plausible to the human eye as a photograph of the real-world material.

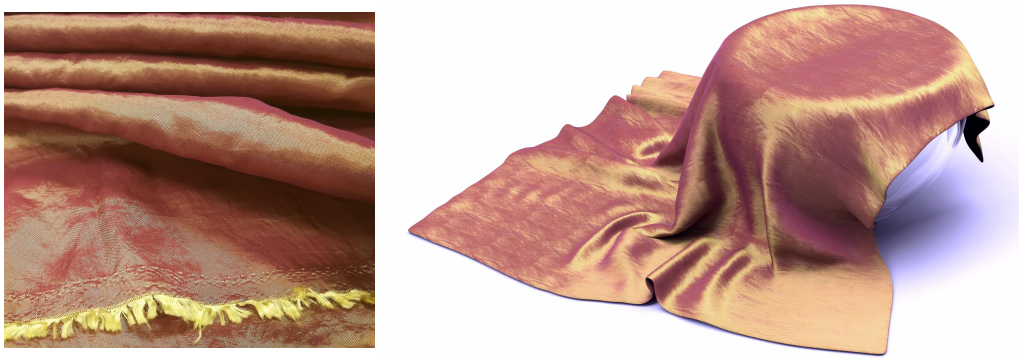


Figure 1.4: Photograph (left) and BTF rendering (right) of the same material (shantung).

Our problem statement is quite straightforward: we have interesting materials (such as Figure 1.4 left) and we would like to be able to reproduce their appearance plausibly in any given virtual scene. In practice, we need to be able to scan the appearance of a small sample of the material, and accurately render it on a virtual object.

1.3 Thesis Overview and Contributions

BTFs have proven to be a very flexible and accurate approach to capture and render real-world materials; however, they come with drawbacks, mainly in terms of practicality of acquisition and storage requirements, which we will address in this thesis.

- First of all, BTFs are difficult to acquire. The high number of precise measurements require a system capable of carrying out the capture automatically. In addition to the hardware challenge, BTFs also require a great deal of software processing to convert the raw data into calibrated and processed data that can be used to render. In Chapter 3, we introduce our custom BTF data processing pipeline. This entails both a description of the specificities of our hardware setup, as well as the particular processing steps we carry out to extract the final BTF texture. The system is showcased in **Publication 1** [RRW19].

- The large number of measurements not only makes BTFs tedious to acquire, but also raises issues in terms of storage. BTF data is very expensive to transmit or to load into memory for rendering, making it impracticable. This motivates the need for a compression strategy, as well as an efficient model for rendering from the compressed representation. Chapter 4, which corresponds to **Publication 2** [RJGW19], introduces a new learned representation for data from BTFs, based on Neural Networks. Our neural BTFs are lightweight in terms of memory footprint and can be used to render directly from the compressed representation. The original BTFs are encoded using only a handful of textures, and the decoder network can be queried for any light-view direction, handling angular interpolation implicitly.

- Although able to reproduce complex appearance behaviors, the encoding previously described is *material-specific*. We address this shortcoming in Chapter 5, which is based on **Publication 3** [RGJW20]. We introduce a single network, trained on many BTF datasets jointly, capable of providing a lightweight descriptor for any new, unseen BTF via simple inference of the encoder network. BTF texels are encoded in a unified parameter space and we can render using these latent descriptors as input to a single learned function (the decoder network). Not only is this approach more practical as it does not require to train a new network for each datasets, but the unified encoding space for BTFs also opens doors for future work in the domain.

Publications

Publication 1

High-Resolution BTF Capture for Delicate Materials.

Gilles Rainer, Kevin Rathbone, Tim Weyrich.

In Conference on Visual Media Production (CVMP) Posters, London, UK, 17–18 December 2019.

Publication 2

Neural BTF Compression and Interpolation.

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, Tim Weyrich.

Computer Graphics Forum (Proc. Eurographics), 38(2), 10 pages, 2019.

Publication 3

Unified Neural Encoding of BTFs.

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, Tim Weyrich.

Computer Graphics Forum (Proc. Eurographics), 39(2), 13 pages, 2020.

Chapter 2

Background and Foundations

In this Chapter, we first briefly review the fundamental models and algorithms of computer graphics that are relevant to this thesis, followed by previous work on material appearance capture. We refer to Weyrich et al. [WLL+08] and Guarnera et al. [GGG+16] for two extensive surveys on appearance acquisition and material modeling. We also assume the reader is familiar with basic deep-learning models and architectures, which are explained in detail by Goodfellow et al. [GBC16].

2.1 Traditional Physically-based Rendering

Photo-realistic rendering is concerned with algorithmically generating images that are indistinguishable from real photographs. To this end, the approach is conceptually close to simulating real photography: a virtual camera is placed in a virtual three-dimensional scene, containing virtual objects and light sources. To produce the final synthetic photograph, light rays are traced from the emitters into the camera, filling in pixels one by one. The virtual camera is assumed to be a perfect pinhole camera with instantaneous exposure. When scene geometry is explicitly modeled, objects are represented by triangle meshes that describe infinitely thin surfaces. Light transport is restricted to geometric optics, which means light propagates along straight lines and is either absorbed or reflected/refracted when intersecting surfaces. For simplicity of notation, we will omit wavelengths in all formulas. Additionally, wavelengths are often not modeled as a spectrum, but instead as a discretized RGB (red, green, blue) vector. Simulation of phenomena such as wave interference, diffraction, phosphorescence, fluorescence is not discussed in this thesis, even though their effect can technically

be present in image-based measurements and renderings.

2.1.1 Radiometric Quantities and Units

Formally, Computer Graphics is concerned with *radiometry*, the measure of electromagnetic radiation. In this subsection, we define the main radiometric quantities and their units.

The elementary unit of measure in radiometry is *radiant energy* Q, measured in *Joules* $[J]$. Light emitters deliver a certain amount of electromagnetic radiation per time unit, which is referred to as *radiant flux* $\phi = \frac{\mathrm{d}Q}{\mathrm{d}t}$. This is expressed in Joules per Second, or *Watts* $[W]$, as it refers to *power*.

Now that we have introduced time into our radiometry concepts, we also need to consider space. The quantity we are interested in is *radiant flux density*, that is the radiant flux passing through a unit area at a given point $\boldsymbol{x}$ on a surface, expressed in $[\frac{W}{m^2}]$. If we only consider incoming light, at a given point, from all directions on the hemisphere, the radiant flux density is referred to as *irradiance*

$$E_x = \frac{\mathrm{d}\phi}{\mathrm{d}A} = \int_{\Omega^+} L(\boldsymbol{x}, \boldsymbol{\omega_i}) \, (\boldsymbol{n} \cdot \boldsymbol{\omega_i}) \, \mathrm{d}\omega_i \tag{2.1}$$

where $\mathrm{d}A$ is the infinitesimal area patch oriented orthogonally to the surface normal $\boldsymbol{n}$ at the point $\boldsymbol{x}$. In the integral notation, $\boldsymbol{\omega_i}$ denotes the direction of incoming light on the positive hemisphere Ω^+. It is important to note that the integrated term is weighted by a dot-product factor of the incoming direction with the normal of the differential surface patch, traditionally referred to as the *cosine law*.

When considering a single incoming direction, we obtain the main quantity of interest in computer graphics, corresponding to the power transported by a hypothetical single light ray. The *radiance* L is the measure of radiant flux per unit solid angle per unit surface area. It is defined as the amount of radiance flux along a direction defined by a differential solid angle $\mathrm{d}\omega$ passing through an infinitesimal cross-section $\mathrm{d}A \, \cos(\theta)$, where $\cos(\theta)$ is the effect of the cosine law, expressing the

foreshortening of the area patch with respect to the ray direction:

$$L = \frac{\mathrm{d}\phi}{\mathrm{d}\omega \; \mathrm{d}A \; \cos(\theta)} \tag{2.2}$$

Ultimately, the rendering process corresponds to propagating radiance distributions from all the light emitters in the scene, all the way to the virtual camera pixels. This is illustrated by the rendering equation [Kaj86]

$$L_o(\boldsymbol{x},\boldsymbol{\omega_o}) = L_e(\boldsymbol{x},\boldsymbol{\omega_o}) \; + \; \int_{\Omega^+} L(\boldsymbol{x},\boldsymbol{\omega_i}) \; f_r(\boldsymbol{x},\boldsymbol{\omega_i},\boldsymbol{\omega_o}) \; (\boldsymbol{n} \cdot \boldsymbol{\omega_i}) \; \mathrm{d}\omega_i \tag{2.3}$$

The exitant radiance L_o in the direction $\boldsymbol{\omega_o}$ is the sum of the emitted radiance L_e and the sum of all reflections of light coming from other sources. These reflections are the result of incoming radiance L_i from all directions $\boldsymbol{\omega_i}$, weighted by the cosine term and multiplied by the reflectance distribution function f_r of units $[sr^{-1}]$, which we explain further in section 2.3.2. In this thesis, we tackle the problem of finding a flexible and convenient model for f_r that leads to efficient and photo-realistic renderings of real-world materials.

2.1.2 Rendering Algorithm

To obtain the final rendered image, a renderer must compute the radiance coming into the virtual camera at every pixel. This problem is usually tackled in an inverse manner: rays are cast from the camera, into the scene, to find the intersections with scene geometry (which correspond to the points visible in the final image). At each of these intersections, the rendering equation must be solved in order to compute how much light is reflected towards the camera.

The first difficulty comes from the recursive nature of the problem: at each of these points, the incoming radiance must be evaluated, which is done by recursively shooting new rays into the scene. Another difficulty comes from the continuous nature of the problem: solving the integral analytically is practically not possible, so Monte Carlo approximations are used, reducing the continuous integral to a discrete sum. This in turn raises a new issue – how to best distribute the samples used in this sum.

In practice, this corresponds to shooting new rays from each point of intersection with a surface.

Many different strategies have been devised to efficiently direct the rays towards the stronger light contributions to the integral, so that the rendering converges more quickly. One common approach consists in approximating the distribution of f_r and using it to guide importance sampling [MUGL08]. In the remainder of this thesis, we will leave importance sampling for the proposed techniques to future work – all renderings used for evaluation will be created using uniform sampling in all directions, with path tracing, because of its unbiased nature.

2.1.3 Distinction of Shape versus Material

In the traditional rendering paradigm, the geometry of scene objects is represented explicitly, at three levels:

1. The first is the overall shape of the object, its macro-scale geometry. In practice, it is most often represented with a triangle mesh.

2. The second one is the surface structure of the object, the meso-scale details in the surface geometry, commonly modeled by bump or displacement maps.

3. The last component is the reflectance of the object. Concretely, this corresponds to physical properties of the matter (composition, absorption coefficient, refractive index, density etc...) as well as microscopic surface details, such as roughness.

In essence, these three components each dictate how light rays will interact with the geometry of the object, at different scales. The boundaries between these scales can however be blurry and are usually chosen such as to balance accuracy of the simulation with model complexity. In practice, this means the definitions of the different scales depend on the scene, and on the distance at which the virtual viewer is from the objects. Classically, the three levels listed here are grouped into two categories, and the distinction is made between *shape* (1.) and *material* (2., 3.). In this thesis, we do not concern ourselves with shape and focus on material appearance modeling instead.

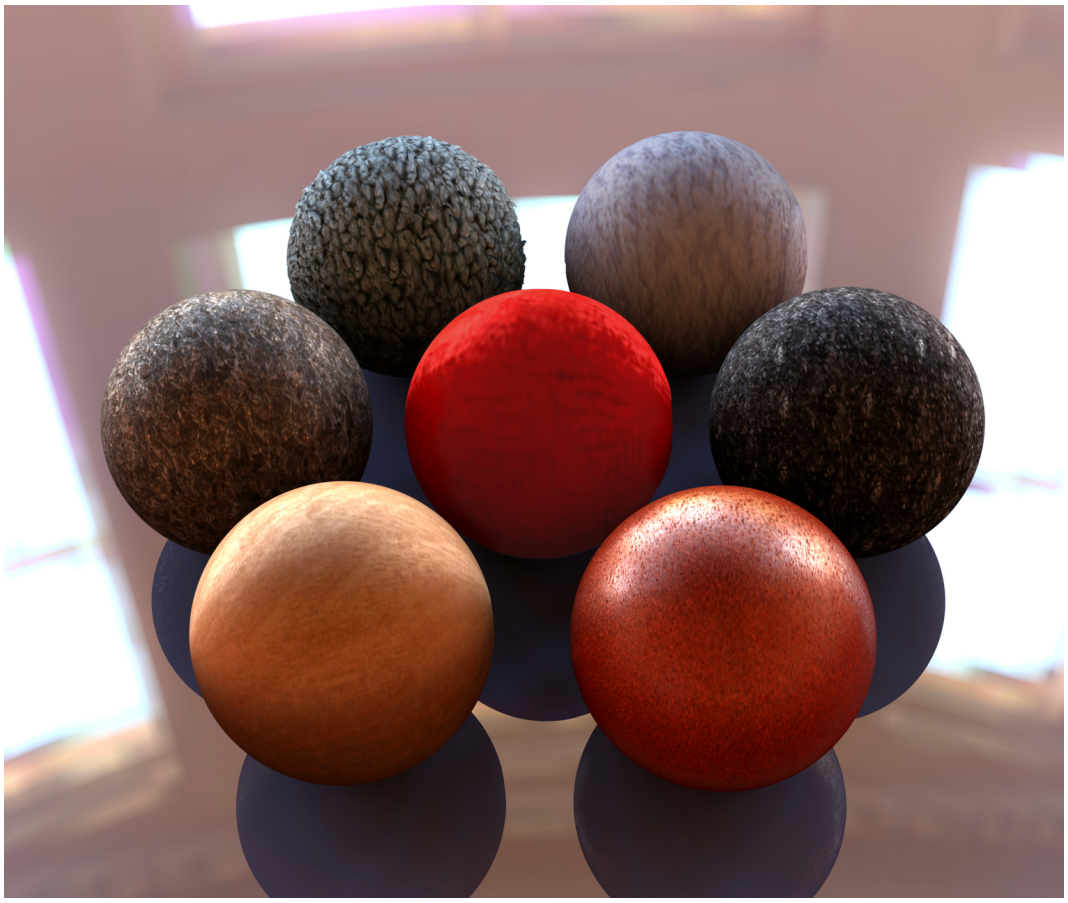


Figure 2.1: Rendering of 7 spheres (same object shape/macro-geometry), out of different materials (micro- and meso-structure).

A vast amount of research in Computer Graphics tackles techniques to convert real-world objects into this representation. The goal is to be able to *digitize* existing objects accurately enough to be able to render them photo-realistically in any virtual scene. Various approaches and devices have been conceived to efficiently capture the shape of objects, ranging from photogrammetry to structured light scanners to time-of-flight sensors, to name a few. For most types of objects, there is an efficient approach to acquire a virtual representation of its shape.

On the other side of the scale however, capturing general material appearance is still a difficult problem, in particular, due to the diversity of materials: from translucent, to fuzzy, to shiny, to layered materials, the way light interacts with the object changes tremendously, to the point that it would require different approaches on the modeling side. This makes finding a general material appearance capture and

modeling technique very challenging.

In this thesis, we focus on improving an existing technique to flexibly and practically acquire and reproduce material appearance, that can insert itself gracefully in the common rendering paradigm. Although our contributions operate in the classical rendering pipeline, several alternatives exist that do not rely on the same explicit modeling of objects, geometry and materials. Since these methods apply concepts that are relevant to ours and served as inspiration, we will briefly mention them while reviewing the main approaches to material appearance modeling.

2.2 Image-based Appearance Models

The main focus of this thesis is the design of a new material appearance descriptor that is compatible with the classical graphics pipeline. To position our work in context, we summarize the main existing material appearance models and review how to use them to match the appearance of real-world materials.

The most straightforward approach to drive appearance and rendering is to directly use real-world measurements, without translation into an explicit model. At the scene level, this is referred to as *image-based rendering* (IBR).

2.2.1 Image-based Rendering

The plenoptic Function. Explicit modeling of the geometry is not a necessary condition to render it. In fact, even the notion of objects is not required, the only quantity that has to be known or computed is the spatial distribution of radiance. Essentially, this means having access to what a virtual eye would see at every point in space. The *plenoptic function* represents the intensity and chromaticity of light observed at every position $\boldsymbol{x}$ and direction $\boldsymbol{\omega}$ in the scene. Dropping time and wavelength dependencies for simplicity of notation, the plenoptic function becomes five-dimensional:

$$P(\boldsymbol{x},\boldsymbol{\omega})$$

Image-based rendering. The high number of dimensions makes capturing and handling such a representation a delicate task that requires simplifications to be made practicable. The most drastically reduced form of the plenoptic function is

the two-dimensional panoramic image (cylindrical [Che95] or spherical [SS97]) – displaying all directions of light at a fixed point in space. Captured photographs of a scene can be considered observed samples of a limited set of directions at a fixed viewpoint. Image-based rendering is the task of creating a continuous approximation of the plenoptic function from such a discrete set of observations, in order to render novel (unseen) images.

Early approaches use two images and impose constraints on the new viewpoints and directions that can be rendered. Image interpolation based on optical flow [CW93] allows the creation of intermediate views, using two input images that are not too far apart. View morphing techniques [SD96] can extend the range of novel viewpoints to camera motion in a plane [Sch96].

Lumigraphs and Lightfields. For outside-looking-in scenes with fixed illumination, lumigraphs [GGSC96] and light fields [LH96] are efficient techniques to render novel views from a discrete set of observations. Conceptually, these approaches rely on a four-dimensional parametrization of the plenoptic function: imagine two parallel planes surrounding the scene (for example "behind" and "in front", in a sandwich fashion), then the view is characterized by the ray intersection with the first (u,v) and the second plane (s,t). These representations assume a constant scene appearance along the ray, only allowing perspectives from outside into the scene. It is not possible to get closer into the scene; the viewer has to stay outside the convex hull.

For both approaches, a database of photographs of the scene has to be captured and pre-processed. Lightfields usually rely on a uniform grid of measured views while lumigraphs can manage arbitrary input viewpoints. Additionally, lumigraphs can use approximate geometry to reduce blurring and ghosting artifacts. The introduction of even more explicit geometry enables the accurate reproduction of more global effects such as inter-reflections and self-shadowing. Surface light fields [MRP98, CBCG02] are usually stored on high-resolution triangle meshes and hence allow for even more realistic novel renderings. However, unlike methods that use varying illumination in the capture phase [SWRK11], the scene lighting is still fixed and cannot be modified.

2.2.2 Diffuse Texture Mapping

Although there are many efficient applications for IBR, the traditional graphics pipeline relies on the arguably artificial separation of shape and material (see Section 2.1.3). The most simple use of data-driven appearance models with an explicit scene geometry would be *diffuse texture mapping*.

Intuitively, this consists in projecting a given image of a material onto the object's surface via *(u,v)-mapping*. During rendering, for every point on the material surface, the corresponding pixel in the texture image is looked up, and its color is used to shade this point. Modulated by light intensities and cosine terms, this method, although very basic, can already provide decent visual results, as long as the material is fairly diffuse. Most importantly, it provides a very cheap and accessible way of approximately capturing a real-world material, as it only requires one photograph.

2.2.3 Reflectance Transformation Imaging

Diffuse texture mapping uses a single image, but why stop at one? Malzbender et al. [MGW01] use a *Reflectance Transformation Imaging* (RTI) Dome to capture 81 images of an object (or material) with a static camera and different incoming light directions (distributed on the hemisphere). When used to scan a material, this set of images can directly be used for rendering, as a light-dependent texture. Adding two degrees of complexity to the data-driven model already creates a significant increase in realism.

2.2.4 Bidirectional Texture Functions

If we also add view-dependency to our dataset of textures, we obtain the *Bidirectional Texture Function*, as introduced by Dana et al. [DvGNK99]. In practice, many photographs of a material are taken under directional illumination, carefully sampling both light- and view-direction hemispheres. At rendering time, the captured data is interpolated to obtain the relevant light- and view-direction combinations in every point on the object mesh. For a detailed early survey on BTFs, we refer to Filip and Haindl [FH09a].

Figure 2.2: Diffuse texture mapping (left) versus full BTF rendering (right).

The BTF is thus a six-dimensional function

$$L(u,v,\omega_i,\omega_o)$$

that takes the position on the object surface, the incoming light direction and the view direction as inputs and returns the associated (observed) reflectance value. The advantage of BTFs [MMS+04] is that they implicitly contain non-local material-light interactions such as the inter-reflections and the shadowing and masking phenomena present in the original material. They even acquire some of the subsurface scattering that goes on inside the material, although it is not technically reproduced accurately at rendering time. All of these additional effects contribute to a much more realistic and softer appearance (see Figure 2.2).

BTF modeling assumes collimated incoming light on a flat texture, a hypothesis which is not always valid: BTFs basically model every surface point as if it laid on an infinitely thin, perfectly flat, opaque surface, which is not true for most materials. Nevertheless, for materials with visually predominant shadowing and masking, such as wool or cotton, BTF rendering can produce much more realistic results than an analytic surface reflectance model would at the same number of dimensions (6), because it also contains measurements of global lighting effects. For highly specular (shiny) materials that can be accurately approximated with a continuous function however, BTFs can be inefficient as a very dense angular sampling would be needed to accurately reproduce high-frequency angular details. Furthermore, Filip et al. [FVHK17] suggest that there exists a *critical distance* beyond which any opaque material can be efficiently represented by a corresponding spatially-uniform model. BTF modeling hence only makes sense for objects viewed from a certain close distance onward, when high spatial resolution is needed and larger-scale effects of light scattering in the material cannot be accurately modeled with a single texture.

Another particularity of data-driven models such as BTFs, that rely on brute-force uniform sampling of all degrees of freedom of the captured function, is that the accuracy of the result depends on the number of samples acquired. If the reflectance of the material has very high-frequency angular variation, the realism of BTF renderings

will be deteriorated if the number of samples is insufficient. Since there are four angular dimensions to sample, ideally exhaustively, datasets rapidly become very large, which causes memory problems both for storage of datasets, as well as for loading of datasets at rendering time. BTF compression is a widely studied topic, of which we review the main approaches in Section 4.1, as well as discuss the specificities and differences of tensor-based compression techniques in Section 4.4.

The original BTF database from Dana et al.'s work [DvGNK99] contains 61 real-world materials, each observed under 205 different viewing and lighting conditions, plus additional measurements, resulting in more than 14,000 images. However, 205 measurements in a 4-dimensional angular space is comparatively very coarse. Currently, the main publicly available BTF databases are provided by the University of Bonn [WGK14] and UTIA [FKH+18]. The UTIA MAM2014 datasets contains 81 light and view directions, resulting in $81 \times 81 = 6561$ angular samples per BTF, whereas the UBO2014 datasets contain 151 light and view directions, amounting to a total $151 \times 151 = 22801$ angular measurements. This high number of measurements makes the acquisition design crucial for captures to remain practicable.

2.2.5 Existing Designs for BTF Capture Systems

For a very complete overview of capture devices and calibration and processing pipelines, we refer to Chapter 4 of Christopher Schwartz's dissertation [Sch15]. To guide the design of an acquisition setup, it is useful to start from the final appearance model, determine the requirements the capture must fulfill, and reverse engineer the design. The BTF is parameterized on light direction, view direction, and spatial position within the material plane: the capture setup must hence allow for an exhaustive sampling of all these dimensions. Some setups are conceived to allow for the full scanning of objects rather than materials [SWRK11]; this entails a 3D reconstruction of the object geometry in addition to the BTF, as well as a mapping from the BTF (u,v)-space to the object surface. In this section however we focus on BTF scanners for flat material samples. Schwartz et al. [SSW+14] give a detailed overview of existing BTF acquisition systems and outline the fundamental constraints each system should respect:

- **Light-Field Capture** – The outgoing light field has to be sampled as densely as possible with an optical sensor. Since we are scanning a spatially extended, opaque sample, this means photographs from any direction in the positive hemisphere.

- **Controlled Illumination** – The sample must be lit with controlled far-field illumination. Arbitrary bases of illumination are allowed as long as any incident light field can be approximated as a linear combination of the basis illuminations that were captured.

- **High-Dynamic Range-Imaging** – The system must allow for HDR captures in order to avoid quantification errors or clamping of measured radiance values.

- **Geometric Calibration** – All captured images must be rectified into a unique planar coordinate grid, so either the sample holder contains targets for camera pose estimation or a separate 3D scan of the material is performed.

- **Radiometric Calibration** – The ratio of light intensity to pixel value must be known for every photograph, to convert all measurements into a unique scale. This calibration can be absolute (which is very hard to obtain), or, more commonly, up to a constant scale.

- **Practical Limitations** – Acquiring a BTF requires hundreds of thousands of measurements, a compromise has to be found to keep the angular capture as dense as possible while keeping capture times reasonable.

Gonioreflectometers. To address all of these requirements, various designs have been proposed. The most straightforward solution is to directly measure reflectance values for a given view- and light direction – by definition, a gonioreflectometer is a device for measuring reflectance at different angles. The Stanford Spherical Gantry (Figure 2.3 is a notorious example of gonioreflectometer setup.

Originally employed for reflectance measurements of spatially-uniform materials, gonioreflectometers can accommodate for BTF captures [DvGNK99] when combined with a camera instead of a photo-resistor. Some setups keep the sample stationary

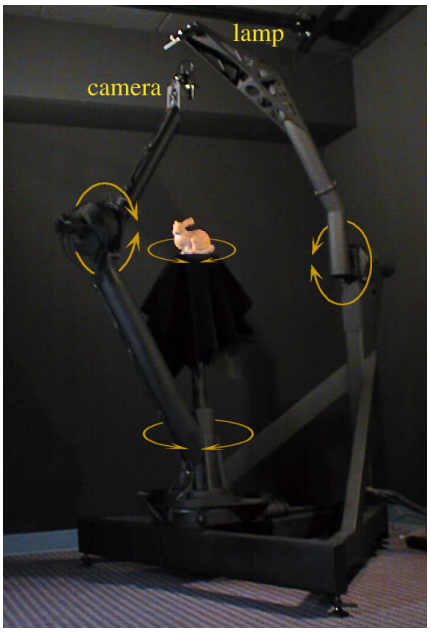


Figure 2.3: The Stanford Spherical Gantry [Sta02].

while light and sensor move to different positions on the hemisphere [HLZ10]. Other designs are able to cover both angular hemispheres by keeping either the light or the sensor at a fixed position, and rotating the material sample into relevant angular configurations [KMBK03, SSK03, TANS05].

Array Setups. Rather than using a single light source and sensor that need to be moved to the desired angular positions, several designs have been proposed that rely on arrays of cameras or light sources. This allows for quicker captures, albeit restricting angular configurations to the static array. Most existing setups employ a dense light array in combination with a couple of fixed cameras [DHT+00, NZLVG05, WMP+06] that can be combined with a turntable sample [SSWK13]. The portable "Lightdrum" system [HHN+17] uses a rotating arc of cameras in combination with a hemispherical array of LEDs, for efficient on-site capture. The BTF Dome I [MBK05] uses a dense array of both cameras and light sources (flashes on the cameras), allowing for comparatively ultra-quick captures.

Mirror-based Setups. Also first attempted for acquisition of spatially-uniform materials [MSY07], some systems rely on the use of a curved mirror to capture a portion of the view hemisphere in one measurement. This idea has been extended by

the use of a translation stage to accommodate for spatial varying reflectance [DW04]. This allows for a serial scanning of different points on the material surface, while the view hemisphere is sampled via the mirror. Other systems rely on planar mirrors in a kaleidoscopic arrangement [HP03, HP03, CHH+17], or in an elliptical arrangement [GTLL06, MTK+10]. While this class of devices efficiently applies parallelization in the angular domain, the approaches can have considerable drawbacks in terms of precision and resolution.

2.2.6 Discussion

The high number of measurements makes BTFs tedious and intimidating to handle, causing bottlenecks at all stages of the appearance capture pipeline: acquisition, processing, storage, and rendering. Nevertheless, when faced with the task of capturing the appearance of an unknown material with complex reflectance and spatial variation, the BTF remains the approach most likely to produce plausible renderings. Indeed, the soundest approach when faced with an unknown material remains measuring an as dense as possible set of samples of the 6-dimensional reflectance function. In this thesis, we implement a custom version of the entire BTF scanning and processing pipeline (see Chapter 3) and search for ways to make the capture more practicable.

2.3 Analytic Appearance Models

Data-driven appearance models produce very faithful results. However, they do not provide any mathematical representation or intuitive parameters that influence the final appearance. In an effort to formalize the processes that create renderings and allow editing of materials, researchers designed different analytic reflectance models.

2.3.1 The BSSRDF

In its most complex form, material appearance is analytically described by the *Bidirectional Scattering-Surface Reflectance Distribution Function* (BSSRDF [NRH+77]).

$$S(u_i, v_i, \boldsymbol{\omega_i}, u_o, v_o, \boldsymbol{\omega_o}) = \frac{dL_o(\boldsymbol{x_o}, \boldsymbol{\omega_o})}{dE_i(\boldsymbol{x_i}, \boldsymbol{\omega_i})}$$

The BSSRDF depends on eight variables: position (u_i, v_i) and direction $\boldsymbol{\omega_i}$ of light entry, and position (u_o, v_o) and direction $\boldsymbol{\omega_o}$ of light exit from the object surface. This allows to account for light leaving the surface at a different point than where it entered, which is particularly important for translucent and subsurface scattering materials. Because of the high dimensionality however, the BSSRDF is difficult to tune and use in practice. A common simplification of the full BSSRDF is given by the *dipole approximation* introduced by Jensen et al. [JMLH01]. This representation models the diffusion of light inside a material similarly to the spreading of heat (only applies to homogeneous materials and assumes half-plane geometry of infinite extent).

2.3.2 The (SV)BRDF

If we restrict light transport inside the material to a single point on the surface, we can eliminate four spatial dimensions. The *Bidirectional Reflectance Distribution Function* [NRH+77] disposes of the most complex light behaviors, reducing the dimensionality of light-material interactions to four parameters: the direction of incoming light and the outgoing direction, each controlled by two spherical angles. Because of its simplicity, this is the most commonly used appearance representation.

Under the assumptions of the rendering equation (2.3 on 25), that is, light hitting an opaque material at a certain position and leaving the material at the same position, the BRDF describes how incident energy is redirected in all directions across a hemisphere centered around the surface normal in the considered point. Mathematically, it is expressed as the ratio between incoming differential irradiance E_i in the direction $\boldsymbol{\omega_i}$ and the reflected differential radiance L_o in the direction $\boldsymbol{\omega_o}$:

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{dL_o(\boldsymbol{\omega_o})}{dE_i(\boldsymbol{\omega_i})} \tag{2.4}$$

In order to be physically plausible, a BRDF must respect two conditions: energy conservation and Helmholtz reciprocity. When integrating the BRDF over outgoing angles ω_o, not matter what the choice of incoming angle ω_i is, the result cannot exceed 1. If it did, it would mean that the material reflects more light power than

it receives, which is physically impossible. It would imply that the material emits light, which should be counted in the emitted radiance term L_e of the rendering equation, while the BRDF only accounts for reflected light. Helmholtz reciprocity on the other hand is respected when the BRDF is symmetric with respect to incoming and outgoing directions, i.e. $f_r(\boldsymbol{\omega_i},\boldsymbol{\omega_o}) = f_r(\boldsymbol{\omega_o},\boldsymbol{\omega_i})$.

The BRDF is only concerned about opaque surface reflections, but there exists a more general version, the Bidirectional Scattering Distribution Function (BSDF). It simulates both the reflection in the positive hemisphere around the surface normal, as well as the transmission into the directions in the complementary hemisphere (inside the material). To this end, the BSDF is generally implemented as a combination of a BRDF and a *Bidirectional Transmittance Distribution Function* (BTDF), which describes how light is refracted through a transparent surface [WMLT07].

A BRDF (or BSDF) describes material appearance in one point. However, most real-world materials exhibit spatial variation in their appearance. To cope with this, the *Spatially Varying Bidirectional Reflectance Function* (SVBRDF) adds two additional dimensions ((u,v) coordinates of the point $\boldsymbol{x}$ on the object surface) to the model:

$$f_r(u,v,\boldsymbol{\omega_i},\boldsymbol{\omega_o}) = \frac{dL_o(\boldsymbol{x},\boldsymbol{\omega_o})}{dE_i(\boldsymbol{x},\boldsymbol{\omega_i})}$$

In practice, SVBRDFs are a combination of one fixed mathematical expression with *parameter maps*. The chosen model remains the same across the material, only the parameters that tune the resulting appearance are varied across the surface. Reproducing the appearance of a real-world material hence requires two steps: choosing the optimal BRDF expression and determining the corresponding parameter maps.

2.3.3 Matching Real-World Appearance

Image-based appearance representations do not require any extra steps to reproduce real-world appearance, as the data is used directly for rendering. Analytic representations, however, have to be *matched* to the observed data first: the parameters of the model need to be tuned to best reproduce the measurements.

Explicit BRDF models can in theory be chosen arbitrarily, as long as they satisfy the constraints for physical validity. Early models were derived from particular material properties or knowledge about its internal geometric structure. For instance, a mirror can simply be represented by a perfectly specular BRDF, while very diffuse materials can be approximated well with a Lambertian BRDF. Kajiya and Kay [KK89] introduced a customized reflectance model based on reflections on a thin cylinder, for materials such as hair or fabrics that contain fibers. If the model is chosen well and is capable of reproducing a real-world material, matching the appearance can be achieved by optimization of the model parameters. For instance, Zinke et al. [ZRL+09] define a *Bidirectional Fiber Scattering Distribution Function* (BCSDF), for a thin, smooth dielectric cylinders with circular or elliptical cross sections. Aliaga et al. [ACG+17] explore the link between parameters of such models to measured physical properties of the existing material, allowing them to efficiently match the appearance of certain fabrics by measuring a single fiber.

When it is not possible to directly link physical quantities to parameters of the BRDF, the solution is to match their visual appearance. Classical reflectance capture methods work by acquiring calibrated measurements of the real-world appearance and finding the parameter combination that best matches. Solving this inverse problem is often referred to as *inverse rendering*.

BRDF Sampling. Performing an exhaustive sampling of both light and view direction hemispheres for a given material is very cumbersome, so most designs and solutions are geared towards making a dense angular sampling more practical. Marschner et al. [MWLT00] propose a straightforward setup, using two cameras and one light source. They propose to use cylindrical or spherical material objects to allow the capture of many BRDF samples in a single image. Matusik et al. [MPBM03] apply a similar approach, but additionally constrain the space of scanned materials to isotropic reflectance – this reduces the dimensionality of the appearance from 4D to 3D. Ngan et al. [NDM05] measure anisotropic materials by imaging strips of the materials, cut along different tangential directions and pasted on a cylinder. In order to further reduce the number of measurements, some approaches [VF16, DJ18] use adaptive

rather than exhaustive samples, only capturing a subspace of the four-dimensional angular domain, which is sufficient to determine the parameters of their model.

Spatially Varying Materials. Most real-world materials are not uniform, however, and exhibit spatial variation in their appearance. This adds two degrees of freedom to the capture problem, making the problem much more complex. Here, exhaustively sampled material appearance corresponds to BTFs. Early work used polynomials [MGW01] and Lafortune lobes [MLH02] to model the directional dependence of each texel. Later approaches model directional variation as mixtures of parametric models [WDR11, SPS13], spherical radial functions [TFLS11], or using a decomposition in terms of measured BRDF responses from the MERL database [WWHL07]. Methods fitting a single model often aspired to reconstruct physically meaningful and potentially user-editable quantities characterizing the geometry (e.g. surface normals), surface albedo, etc. [LBAD$^+$06, WMP$^+$06, MG09].

However, analytic SVBRDF 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. This reflects in the fact that residual of the fit is often kept and compressed separately [MCT$^+$05, WDR11]. For the most part, this residual contains non-local appearance effects such as inter-reflection, shadowing and masking between nearby points, and subsurface scattering.

Multiplexed Illumination. Other methods aimed at modifying the capture approach to reduce the number of necessary measurements. One way to do this is multiplexing the illumination so that it covers many different light directions simultaneously. This is especially useful for highly specular, spatially-varying materials, where it is easy to miss the sharp reflection lobe when using directional illumination from discrete angles. Multiplexed or extended illumination increases the probability of illuminating the material in the specular direction, and hence capturing the specular lobe of a high number of points on the material in the measurements. As a downside, this puts more complexity into the subsequent processing and parameter fitting, as it is difficult to disentangle the light directions in the observations. The advantage of the multiplicity

of light directions, however, is that thin highlights have a higher chance of being captured, and the measurements contain less noise in cases where light intensity and exposure settings cannot be further improved.

Gardner et al. [GTHD03] capture flat samples using a linear light source(neon tube) that is translated horizontally over the surface of the object, in sync with camera acquisitions. Tunwattanapong et al. [TFG+13] achieve controlled spherical harmonic illumination conditions by sweeping the incoming light hemisphere with a rotating semi-circular arc of LED light sources. Ghosh et al. [GAHO07, GCP+09, GCP+10] and Kampouris et al. [KZG18] exploit polarized and spherical gradient/harmonic illumination patterns to efficiently separate reflectance components and estimate isotropic and anisotropic SVBRDFs. Also using basis function illumination, Aittala et al.[AWL13] present a capture method for isotropic SVBRDFs that uses an LCD screen to display windowed sinusoids (Fourier basis functions), allowing the model fit to be performed in Fourier space.

In the vein of low-cost acquisition strategies, Riviere et al. [RPG16] propose a mobile reflectometry solution for isotropic planar materials using a mobile device's LCD screen. Lambertian reflectance is generally a way to model the compound effect of many (inter-)reflections on uniformly oriented reflectors at a microscopic scale, which also has the tendency to undo any linear polarization of incoming light. Specular reflections on the other hand preserve the polarization. Riviere et al. implement a system that exploits the polarization of LCD screens via two photographs of the material with a differently oriented linear polarized in front of the camera, to capture parallel and cross polarization, which allows them to separate diffuse from specular reflectance. In the same low-cost and commodity hardware acquisition spirit, Aittala et al. [AWL15] capture stationary materials using a pair of mobile phone images with and without flash, and reconstruct full anisotropic SVBRDFs.

A Priori Knowledge. Inverse rendering is inherently an ill-posed problem, as many different combinations of parameters can yield the same visual outcome. It gets even more complex when area light sources are introduced and the number of measurements is reduced. To constrain the optimization space, the fitting process can

be guided and be made more stable via the use of priors. Point-wise priors correspond to constraints on the parameter values themselves, while smoothness priors exploit the fact that many existing material exhibit slowly varying spatial variation. Aittala et al. [AWL13] leveraged the fact that parameters of close-by points in real-world space should also be similar in parameter space. Designing such priors by hand is very challenging however, and often involves a tedious tuning phase to balance the weighting of priors and inverse rendering loss.

2.3.4 Neural SVBRDF Estimation

Many recent works have incorporated deep learning into the appearance matching pipeline, which already heavily relies on optimization procedures. Convolutional Neural Networks have emerged as a tool to replace the traditional non-linear optimization [LXR+18, DAD+18, MHRK19, GLD+19]. One particularity of these approaches is that the networks are able to implicitly learn effective priors from the data, improving the SVBRDF estimation. The implicit knowledge can be powerful enough to allow for SVBRDF estimation from a single image [LSC18, BL19], even in unknown, uncontrolled lighting conditions [LDPT17].

Deep learning has also been applied in other parts of the appearance matching process. Kang et al. [KCW+18] use an auto-encoder to learn illumination patterns and decode texels of multiplexed lighting captures of a material. They extend this work in [KXH+19] to jointly estimate shape and reflectance using learned illumination patterns, and decoder networks that estimate SVBRDF parameters.

Aittala et al. [AAL16] use a network as a better judge of similarity: rather than a point-wise comparison of their inverse renderings to the measurements, they make use of the statistics of a neural network-based texture descriptor. This allows them to accurately compare each patch of their estimated material to different patches of the original material, thereby obtaining more information. Compared to [AWL15], which uses a photograph with and without flash, the new method only requires the flash image.

Deep learning has hence proven to be an effective way of combating the under-determined aspect of the problem. The priors are implicitly and automatically learned

from statistics in the observations. On the other hand, the learning procedure requires a large database of examples. Fortunately, explicit models lend themselves well to synthetic data generation, which is a strategy often used for data augmentation, at the risk of biasing the learning. A comprehensive survey about appearance capture in combination with deep learning can be found in [Don19].

2.3.5 Discussion

Analytic models are very practical because they lend themselves to many optimization strategies during rendering, from importance sampling to GPU accelerations. However, this comes at the cost of reducing the space of functions that can be accurately reproduced. The simpler the model, the more restricted the variation of reflectance functions that can be fitted. This limit is most noticeable when the materials contains compound effects. Fabric appearance, for instance, can be modeled well at a microscopic scale with analytic models of cylinder reflectance, when each fiber is modeled individually. When a textile is modeled as a surface however, the expressiveness of the chosen analytic function is not sufficient anymore to approximate the compound reflectance effect accurately. In this situation, data-driven approaches often produce better results. In this thesis, we want to preserve the practicality of more explicit models, without the reduced expressiveness that occurs when choosing a specific analytic model.

2.4 Neural Appearance Models

Orthogonally to the two sides of appearance modeling previously described, another approach has emerged recently. Based on learned, more implicit representations, recent advances in deep learning have given rise to *neural rendering*. Rather than using deep learning as an advanced, more informed form of optimization, several works have started modeling scenes and light transport quantities themselves with networks. For a comprehensive survey on neural rendering, we refer to [TFT+20]. In this section, we restrict the review to methods that operate at different levels of explicitness of scene modeling, with camera and/or lighting controls, but we leave out techniques that synthesize plausible photographs purely in the image domain,

without any explicit 3-dimensional control parameters or supervision.

2.4.1 Neural Image-Based Rendering

To learn to encode and reproduce scene content, autoencoding networks (see [HS06, GBC16]) have been successfully applied to produce increasingly abstract visual representations, proving to be capable of implicitly learning scene structure. Kulkarni et al. [KWKT15] present an early approach, where a rendering network is trained to compress scenes of a specific class (faces, chairs) to a latent code and decompress the code into a rendering. Specific attention is given to the disentanglement of meaningful parameters in that latent code, such as pose, shape and light. Lombardi et al. [LSSS18] follow in the same vein of research, introducing a deep appearance model for face rendering, while similar approaches have also been developed for scene relighting [RDL+15, XSHR18]. Meka et al. [MHP+19] achieve faithful reproduction of even more complex appearance phenomena such as high frequency details and specular highlights, relighting facial scenes with arbitrary environment maps, given only two color gradient images as input.

Simultaneously, *Convolutional Neural Networks* (CNNs) have also emerged as a powerful tool to learn and approximate complex functions and pipelines. Zsolnai-Fehér et al. [ZFWW18] render previews of materials, mapped onto a static scene geometry, harnessing the power of neural networks to quickly produce a faithful, though approximate solution, which suits their application of material design particularly well. Also related to this is *Deferred Neural Rendering* [TZN19], which features texture maps along with a neural renderer that allows to store and decode much more information than diffuse textures, preserving specular highlights and parallax. Conceptually closer to early image-based rendering, Hedman et al. [HPP+18] reconstruct a geometry proxy from a set of images of a scene and warp them into the novel view. Their method generates a set of ranked contributions (mosaics) from the input images, and a CNN is used to predict the respective blending weights. In the same vein of research, Mildenhall et al. [MSOC+19] combine deep blending of light fields with the use of multiple, local multi-plane images, combining neural approaches with more traditional novel-view synthesis techniques.

Most recently, Mildenhall et al. [MST+20] introduced *NeRF*, a neural light-field representation that allows high-fidelity novel-view synthesis. Their approach relies on a scene-specific network parametrized on spatial coordinates and directions, that outputs radiance and density values for a given input ray. Renderings are produced with a differentiable volume rendering technique. Chen et al. [CWZ+18] and Bemana et al. [BMSR19] also use networks to encode light fields, but with different goals in mind. Chen et al. work with surface light fields, whereas Bemana et al. use light field videos (containing spatial and temporal dimensions). The common particularity of all these approaches is that a new network instance is trained for every new light field. This means that the network only takes the plenoptic parameters as input, instead of any additional images or latent code describing the scene. The light field information is learned and contained in the network itself, as an abstract, implicit representation.

The idea of light-field encoding is taken a step further in *Neural Volumes* [LSS+19]. Lombardi et al. introduce a network to encode light fields in a latent representation. Taking additional controls like view direction or head pose as input, the decoder creates an RGB-alpha volume that can be rendered via ray marching. Since the network is pre-trained, it can directly encode new inputs, allowing for latent encoding of dynamic scenes. This also allows the authors to optimize the decoder's performance, to the point of reaching real-time rendering performance.

In essence, the presented approaches are similar to image-based rendering, with a network taking 2D observations of a scene as inputs and using them to predict the appearance of the scene in new viewing or lighting conditions. In that sense, the network learns to encode view-dependent effects and geometry in an implicit representation. Other works, however, have approached the topic differently and attempt to learn the entire traditional rendering pipeline. Captured (or synthesized) views are then only used to compute the loss, but not used as inputs to the network. Instead, the network learns to process explicit scene representations.

2.4.2 Deep-Learning the Complete Light Transport

Scene Representation Networks [SZW19] are a perfect example of fully-neural rendering pipelines. The first network learns to map points in a scene to a high-

dimensional feature vector, effectively learning an implicit representation of a scene. A second network then learns to extract an RGB pixel value from a high-dimensional feature. The final rendering is obtained by ray-marching through the scene, at every pixel, to find points of intersection with the geometry and retrieve the features that are passed to the rendering network. To generalize across scenes, a hyper-network is trained to predict the weights of the scene representation network for every new scene.

One common characteristic of fully-neural rendering pipelines is that the processing of 3D and 2D are often separated into distinct networks. Chen et al. [CCZ+20] take the idea of neural textures of Thies et al. [TZN19] a step further by adding a network that extracts global appearance features from the input mesh, and a fully neural renderer that learns to simulate light transport. It takes neural textures, normal maps, view direction maps, etc., in combination with spherical harmonic light descriptors as input, and outputs the final rendered image.

In the same fully neural paradigm, Sanzenbacher et al. [SMG20] also learn the entire rendering pipeline, but their approach exploits the 3D domain as much as the image domain by predicting local appearance features in 3D in addition to the 2D treatment of normal, albedo, etc. maps. They show that, for a fully neural rendering system to produce convincing realistic results, it needs to reason in the 3D domain as much as in the image domain.

3D reasoning is difficult for convolutional networks as the number of elements grows in a cubic manner, which is a significant obstacle for applying deep learning techniques to explicit scene representations. *RenderNet* [NPLBY18] uses a voxel-based representation with 3D convolutions, which limits the final resolution of objects to 128^3 voxels. Most recent approaches, however, have been relying on architectures similar to *PointNet* [QSMG17] for the processing of 3D shapes, usually in the form of point clouds.

2.4.3 Approximating Components of the Rendering Pipeline

Some hybrid approaches operate inside the traditional rendering pipeline, replacing only components with learned networks, or optimizing elements of the process such

as neural importance sampling [MMR+19]. Ren et al. [RWG+13] pioneered this direction by regressing indirect illumination in a scene via small neural networks, that take as input a point in space, as well as SVBRDF parameters and light/view directions, and return radiance. Essentially, they perform the same task as an appearance model, restricted to indirect illumination. This allows them to learn a quick, approximate solution which, when added to a direct illumination rendering, achieves global illumination at far lesser computational cost. To handle more complex scenes, the space is partitioned, and a separate network instance is used in each part.

Hermosilla et al. [HMRR19] extend this to learning of global light transport effects. Their method works on point clouds, extracting light transport priors from the geometry with a first point-cloud network. These features allow them to reconstruct ambient occlusion, global illumination and subsurface scattering with another image-space network. Although their method is only applied to single objects and a maximum of two diffuse, homogeneous materials, the results are fast and accurate.

2.4.4 Neural Material Models

Both of these approaches show that the geometry can be modeled explicitly and that individual parts of the light transport can be learned efficiently. Preserving almost the entirety of the traditional rendering pipeline described at the beginning of this chapter, Brady et al. [BLPW14] have employed genetic programming to develop a framework for learning of new analytic reflectance functions that better describe specific materials, effectively learning a new material model. Following in this direction of preserving flexibility and expressiveness by learning appearance functions rather than choosing analytic representations, Maximov et al. [MRLF19] introduced the concept of *Deep Appearance Maps* (DAM), which use a small fully-connected network as a learned material reflectance function itself. However, DAMs are parameterized on normal and view direction, rather than light and view direction.

Similarly, Kuznetsov et al. [KHX+19] use a *Generative Adversarial Network* (GAN) to avoid explicit modeling and simulation of the surface micro-structure, instead learning a range of microscopic normal distributions from data. This allows them to obtain a more flexible representation of specularity that is additionally closer

to real-world materials.

Finally, Henzler et al. [HMR19] tackle the domain of texture synthesis with a network that learns a continuous 3-dimensional representation from 2D or 3D texture exemplars. The network is trained over multiple textures so it does not need to be retrained to encode a new material. Using a continuous 3D representation brings many advantages over traditional uv-mapping, and a representation can be computed for a new material simply via inference of the network.

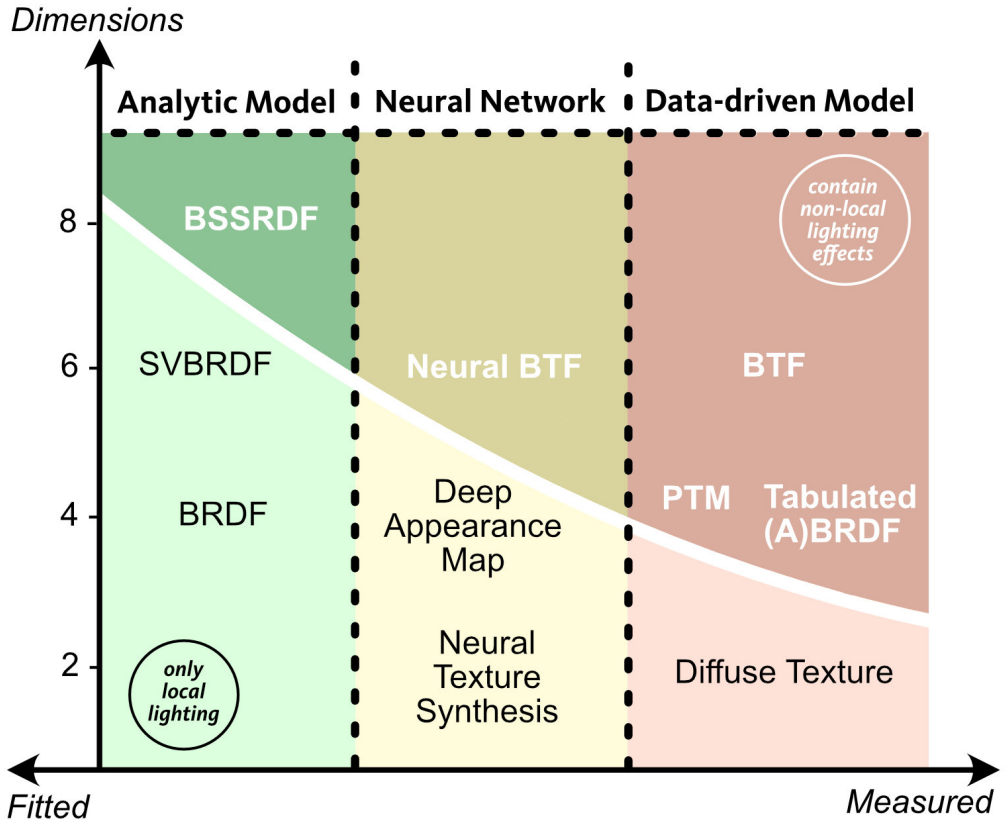


Figure 2.4: Hierarchy of material appearance models previously mentioned.

2.4.5 Discussion

The approaches presented in this thesis rely on an explicit scene model (separation of geometry and material), but we draw a lot of inspiration from neural image-based rendering. The neural reflectance models reviewed in this last subsection are closest to the approach we adopt in this thesis – scene macro-geometry is still represented with explicit meshes, but complex light behaviors that we do not have good explicit

approximations for are learned by networks.

Neural material models rely on a choice of explicit architecture with parameters that need to be fitted to real-world measurements. The model parameters are however less intuitive and less explicit than in traditional analytic models. For this reason, neural material models can be viewed as an intermediate between analytic and data-driven models. We lay out all the previously described representations in a general taxonomy shown in Figure 2.4.

2.5 Conclusion

Based on the findings of this literature review, we aim to design a material model that is flexible enough to incorporate non-local effects from real-world materials. BTFs emerge as the most flexible data-driven technique for complex materials; however, they are very tedious to acquire and to use for rendering. Handling the raw BTF is too cumbersome, it needs to be converted into an efficient representation that stays as faithful as possible to the raw data. Analytic SVBRDF models have proven to lack the expressiveness to accurately reproduce the original BTF. Neural networks, on the other hand, have shown themselves capable of accurately approximating complex light effects, as well as learning priors, better than their hand-made equivalents. We draw on all these conclusions from the related work to propose our own, custom BTF acquisition pipeline (Chapter 3) and conversion into a new, practical, *neural* BTF model (Chapters 4 and 5).

Chapter 3

Capture Hardware and Processing Pipeline*

Because of their data-driven nature, bidirectional texture functions are tightly coupled with their acquisition process: calibration and processing play a critical role. This Chapter describes the capture hardware (shown in Figure 3.1)) and the data processing pipeline. Section 3.1.1, which describes the hardware designed specifically for this project, was developed before my involvement.

The device was specified by my supervisor Tim Weyrich as part of an industrial collaboration with Change of Paradigm Ltd. I received access to the hardware with the capability of reaching a given camera position, light position, activating the required camera settings and capturing a photograph. Beyond that, I had to develop the full calibration and data processing infrastructure as part of my thesis project. For completeness, this section will also give a detailed description of the hardware design and design considerations.

3.1 Device and Data Acquisition

Our device was designed drawing inspiration from existing setups described in Section 2.2.5. Array-based designs are the most efficient in terms of capture times and require little to no mechanical movement. However, since they involve many different sensors and light sources, achieving a precise radiometric and geometric

* Hardware specification, design and build were performed by Tim Weyrich and Kevin Rathbone prior to my involvement. The software processing pipeline was implemented by myself exclusively.

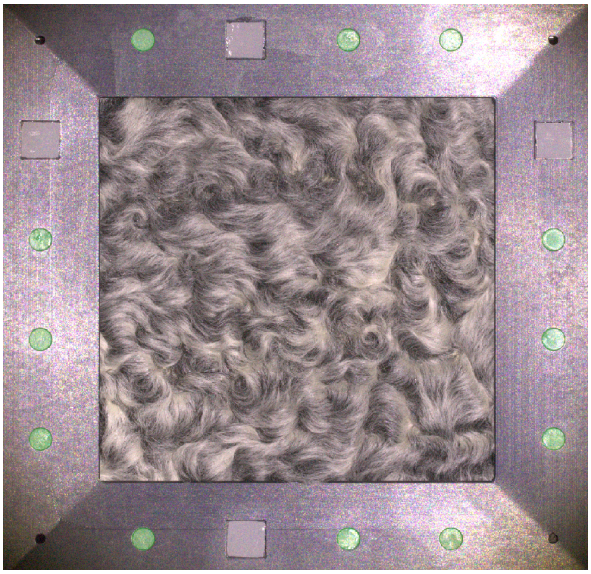
Figure 3.1: Our custom BTF capture system under room lighting.

calibration of the whole system becomes a complex task. Another drawback comes from the fact that the positions in the array are fixed. These setups cannot adapt their angular sampling to accommodate for different materials and strategies, the capture protocol is fixed. In our build, we wanted to keep possibilities open for future experiments with different angular sampling strategies and resolutions. We opted for a system with a single emitter and sensor, that are mechanically moved to (arbitrary) positions. Some of the previously existing builds simplify construction and reduce mechanical load by mapping one or two of the rotations to a turntable holding the sample. However, this restricts the range of materials that can be scanned, requiring them to be either rigid or stretched in frame, so that the material does not move under gravitational influence. To keep our captures open to delicate soft structures such as fine fabrics, we opted for a stationary sample and keep the light source and camera freely positionable.

3.1.1 Specificities of our Design

The gonioreflectometer (Figure 3.2a) consists of two computer-controlled arms that respectively carry a camera and a light source, and travel on a circular rail around a flat platform. Via inclination of the arms, the camera and light effectively reach

(a) Acquisition device.



(b) Sample holder.

any position on the positive hemisphere above the platform, down to 5 degrees elevation above the sample holder's plane. Camera and light are pointed at the sample holder, which rests at the center of the platform. The system is designed to scan flat 10×10 cm material samples that the sample frame holds down. We use a Canon EOS 5DS R 50-Megapixel camera along with a Canon EF 100mm $f/2.8$L lens, at 902 mm from the sample center. This amounts to a raw frontal resolution of 37 µm per pixel on the physical sample under frontoparallel observation.

The Smart Vision SXA30-WHI LED light source delivers an even illumination over the sample and a balanced light spectrum. It travels on a slightly narrower hemisphere (757 mm radius) to avoid collisions with the camera arm, and should be kept at least at 5 degrees arc distance to avoid occluding the view. The entire system is located on vibration-isolating mounts in a dark room, and building materials are chosen such as to minimize stray light reflections.

In order to keep angular capture configurations flexible, we accepted the overhead of mapping all four degrees of freedom in the capture to the articulation of camera and light source arms. We keep this overhead minimal by using very rigid but lightweight robotic arms on harmonic drives, enabling fast, jerk-limited motion. This eliminates

waiting times for vibrations to subside and allows new positions to be reached while the previous image is still downloading.

One usual limitation of this type of design remains: back-scattering reflectance can not directly be sampled. A minimum of 5 degrees arc distance between light source and camera must be preserved to avoid occluding the view. This limitation prevails in most BTF scanners. One option to circumvent this problem would be the use of a semi-reflective mirror, which is difficult to incorporate in such a system with multiple moving parts and would require an additional light source. We chose to address this issue by measuring additional light directions around the back-scattering configuration, as closely to the occluding region as possible.

Illumination of the Sample Holder. At rendering time, when queried for a given incoming and outgoing light angle, the BTF model assumes that the local neighborhood of the shaded point is illuminated under the same conditions (uniform, directional lighting). This way, the material model can faithfully reproduce non-local light transport effects within the meso-geometry, as they are contained in the data, without explicitly computing any non-local interactions. However, for these effects to be correctly contained in the captured BTF data, the same conditions have to be met during the acquisition: within a local neighborhood (spatial extent varies from material to material), the illumination has to satisfy the directionality and uniformity condition, at sufficient accuracy. Also, BTF data is traditionally binned according to discrete illumination directions, which facilitates processing and compression, since it allows to write the BTF as a consistent 2D-matrix of binned discrete angular and spatial positions.

In practice however, with a compact light source significantly smaller than the sample itself, parallel rays can only be approximated by moving the light source far away, which has to be balanced against geometric constraints and the resulting loss of light intensity. In our setup, the maximum angular deviation between the rays hitting the corners of the extracted squares and the center will be 2.14 degrees. For binned acquisition at angular resolutions as we use them in our project, we

deem this acceptable as it is well below the angularly discrete lighting resolution. Higher-accuracy applications would have to explicitly take the variation in angle of incidence into consideration and forgo the binned representation, instead opting for a more flexible encoding such as the unified model presented in Chapter 5 for example.

For the uniformity condition, we chose a light source with integrated projection optics designed to minimize light loss and to produce smoothly varying illumination that could easily be calibrated for. Experimental observation of the resulting flat field showed that the uniformity was high enough that we forwent flat-field calibration in our work: in Figure 3.3, we observe a deviation of ±2.5% across the entire sample holder (which holds 4 material samples).

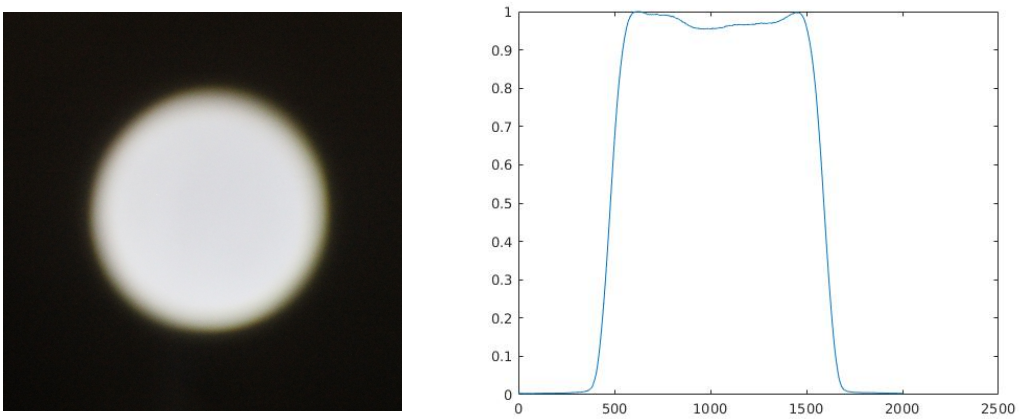


Figure 3.3: *Left:* Photograph of the irradiance field on a diffuse surface. *Right:* Plot of the pixel intensities (in linear RGB) across a line through the middle of the image.

3.1.2 Raw Capture Protocol

At every angular configuration of light and camera, several photographs are recorded to obtain accurate measurements and calibrations (Figure 3.4). We acquire one image under programmable, diffuse ceiling lighting, one measurement with the LED, and optionally one measurement in the dark, for situations where stray light might be expected. The last two are exposure stacks (consisting of up to 10 photographs) that will be merged into a high-dynamic-range image. In the remainder, we refer to these respectively as *primary light image*, *modeling light image* and *black image*.

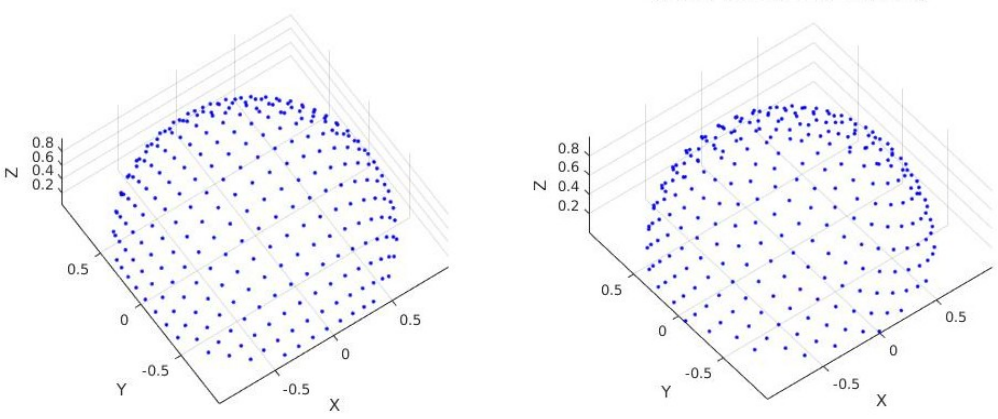
Figure 3.4: Raw images captured at one position. Left to right: *modeling light* image, followed by *primary light* exposure stack.

Capturing that many additional shots means capture sessions are much longer than one would expect from the size of the final BTF set. Each raw dataset contains more than ten thousand images and takes up over a terabyte of memory. Depending on the reflectivity of the materials and the potentially long shutter times in the exposure stack, the captures take up to 48 hours. For calibration purposes, we capture four more datasets in addition to the BTF:

- a fairly dense sampling of the view hemisphere under modeling light, to 3D-reconstruct the material surface – around 70 images;
- an HDR measurement of a target of known albedo and with a color grid of known RGB values, for radiometric calibration;
- a handheld set of photographs of a camera intrinsics calibration target in different orientations, in our case we use OpenCV's assymmetrical circles – around 20 images.

Angular Sampling Pattern. The choice of sampling pattern for the view and light hemispheres is up to the user. No angular sampling is perfect, as any optimal strategy for a certain set of points on the surface will be sub-optimal for other points on the material. Some sampling patterns (Figure 3.5a) are generated by regularly sampling the unit disk and lifting the positions to the unit hemisphere. In order to avoid too much grid-like regularity in our sampling, we opt for an approach based on the Fibonacci sequence [MBR+13] as shown in Figure 3.5b.

In our first captures, we aim for a resolution of approximately 40 light positions by 40 view positions. To achieve this, we place 40 points on the sphere according to the sampling strategy, and then remove the positions that are beneath the 5 degree elevation threshold that would result in occlusion by the sample holder frame. This leaves us with 37 positions on the hemisphere (Figure 3.6, left). For the light



(a) Grid sampling of the disk lifted onto the hemisphere.

(b) Sampling of the hemisphere driven by the Fibonacci series.

positions, we add 4 illumination directions around every back-scattering configuration (which would be occluded), making the light positions denser around the camera (Figure 3.6, right). Filtering out the occluded positions again, we obtain 1508 angular configurations, that is, view-light combinations, in the end.

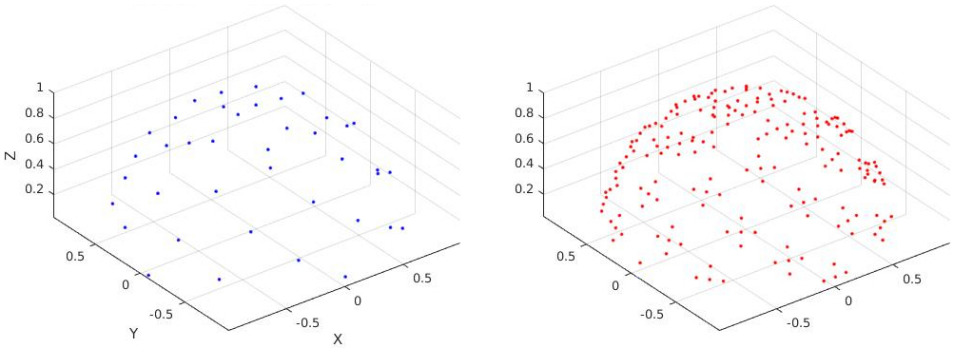


Figure 3.6: Angular positions of the camera (*left*) and the light source (*right*) in our captures.

3.1.3 Comparison with Publicly Available Datasets

Currently, the most prominent sources for publicly available datasets come from the University of Bonn and the Institute of Information Theory and Automation (UTIA). Both contain fairly recently captured (2014) BTFs of high angular and spatial resolution, processed with high accuracy. The UBO2014 datasets [WGK14] from the University of Bonn were acquired with the Dome I[†] setup. The database

[†]The camera DOME. Computer Graphics Group, University of Bonn. https://cg.cs.uni-bonn.de/en/projects/btfdbb/dome/

contains 7 semantic categories of 12 materials each. Each material sample has been scanned using the bi-directional sampling of 151 viewing directions and 151 lighting directions. The cameras are distributed uniformly on a hemispherical grid, and the camera flashes are used for lighting. Each capture hence contains a total of 22,801 images for material samples with a size of 5×5 cm^2. The available BTFs contain the central 400×400 texels (corresponding to an area of approximately 3.9×3.9 cm^2), which represents a footprint of 97.5×97.5 µm^2 on the physical sample.

UTIA, on the other hand, offers 6 BTF datasets [HM12] from 2011, as well as a more recent database of 16 materials, the MAM2014 database [FKH+18]. The materials were presented at the Eurographics Workshop on Material Appearance Modeling and were chosen to describe a wide range of appearances. The BTFs were captured with a 4-axis gonioreflectometer of high precision, at 81×81 uniformly distributed illumination and view directions that follow the sampling of an earlier version of the Dome I. The datasets have varying spatial dimensions, usually between 100 and 200 texels in height and width. In frontal views, a pixel records a spatial area of 46×46 µm^2 on the physical sample.

In contrast, our initial BTF datasets use a lower angular sampling, but a higher spatial sampling (10×10 cm^2 material samples at a raw frontal resolution of 37 µm per pixel). This allows us to capture more spatial detail at the expense of lower angular frequency variation, which means our current BTFs capture effects such as subsurface scattering and diffuse reflectance better and are less suited for highly specular materials. However, highly specular materials will always be sampled sub-optimally, regardless of how densely the angular hemisphere is sampled. Additionally, our setup can acquire arbitrary angles, so there is no obstacle to raising the angular density in future captures.

3.2 Data Processing

Processing the raw data is an integral part of the BTF capture pipeline. The raw values are to be geometrically and radiometrically calibrated to obtain data that can be used to render from. In practice, this means converting the measurements into

physical units, and mapping them to a unique, shared coordinate grid on the material.

3.2.1 Radiometric Calibration

At each angular configuration, the exposure bracket is merged into a high-dynamic-range image. However, the quantity we are interested in is the reflectance, that is the ratio between incoming and outgoing radiance. In order to obtain this, the pixel values must be normalized by the light intensity, and the RGB components must be white balanced.

White Balance. Using the same exposure bracket as for the BTF capture, we acquire an HDR measurement of the X-Rite ColorChecker Classic Mini® under primary light illumination. The sRGB colors of each of the squares on the target are known. By fitting a transformation (in linear RGB) from the observed values to the reference values, we can solve and find the values of a color calibration matrix with 12 entries. Many transformation models are possible, but we found an affine model to be the best compromise between flexibility and rigidity (to prevent overfitting the color patches). See [FL00, Ami02] for a discussion of different color calibration models.

To white balance our HDR images, we use the following formula:

$$\begin{bmatrix} r' \\ g' \\ b' \end{bmatrix} = \mathbf{T} \times \begin{bmatrix} r \\ g \\ b \end{bmatrix} + \mathbf{t} \tag{3.1}$$

where $r'g'b'$ are the calibrated color channels, rgb are the input color channels, $\mathbf{T}$ is a 3-by-3 matrix and $\mathbf{t}$ is a 3-dimensional vector. The result of the color calibration is shown in Figure 3.7.

Intensity Normalization. The previous step allows us to calibrate the color response of the camera channels in relation to each other, but the measurements are still in relative units: across the HDR images, radiance is measured up to an unknown global scale. Luckily, we are interested in reflectance, which is the ratio between radiance and irradiance, so this scale is canceled out. To anchor our reflectance scale, we acquire an HDR measurement of a Spectralon™ in the sample holder frame, under primary light illumination, which will allow us to find the global multiplying constant

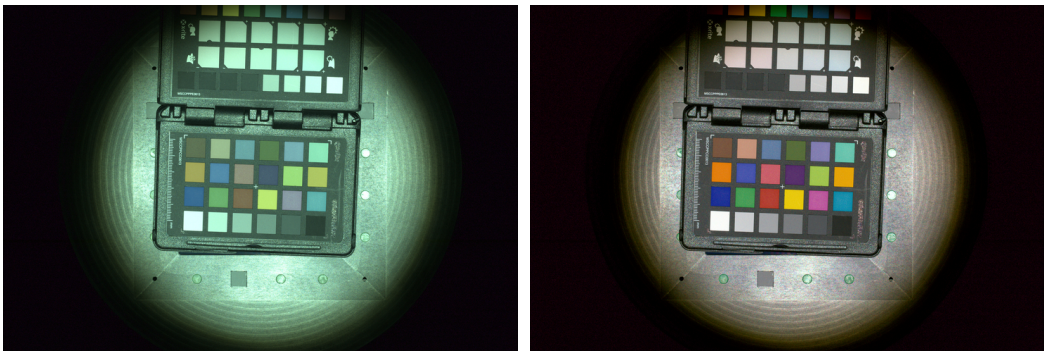


Figure 3.7: Color checker under primary illumination, before (left) and after (right) color calibration.

to convert all observed values.

The Spectralon has a known reflectance, as it diffusely reflects 99% of incoming light. Note that in the measurement of the Spectralon under primary light, we need to additionally take into account the cosine foreshortening term of the primary light (which is at 87 degrees elevation in this configuration to avoid obstructing the camera). Since the incoming energy is uniformly reflected over the positive outgoing hemisphere, this effectively means that the reflectance of the Spectralon into a single direction is $\frac{0.99}{\pi}$. The value observed in the image should then be matched to the expected radiance of $\frac{0.99}{\pi} \cdot \cos(\frac{3\pi}{180})$ to anchor all observed reflectance values.

Final Calibration. Using the color checker and the Spectralon, this procedure allows us to determine all the coefficients to convert the HDR measurements of the BTF into reflectance values that can be used for rendering. The entire process relies on additional assumptions, such as, that the light source is distant enough to be considered directional and evenly lighting the sample, as well as that the light source intensity does not change over time. Experiments on light fall-off conducted by Tim Weyrich for the gonioreflectometer specification showed that the illumination, which relies on additional optics, was very evenly distributed in the region of interest (sample holder). The camera and lens models, in combination with the capture settings – small aperture (f/22), long exposure times – also produces very low vignetting. We only use pixel values of a central region in the captured image, so color aberrations in the corners are also not corrected. We consider all of these simplifying assumptions reasonable in our current setup – if higher precision is required for future captures,

one could add more complexity to the calibration.

3.2.2 Geometric Rectification

The aim of the rectification is to project all the textures into a common reference coordinate system on the material, which we choose to be an orthonormal 3D grid anchored in the sample holder. In order to do this, we must find the transformation from world to image coordinates, that the camera applies, and reverse it. Commonly, this is separated into intrinsic calibration and extrinsic calibration. Extrinsics model a rigid transformation of the world coordinates to convert it into the camera's perspective. This contains the camera's pose, that is its origin and orientation in space. Intrinsics represent the camera's lens system (focal lengths, aspect ratio, radial distortion) and model the mapping from 3D coordinates in camera space to 2D image coordinates. The camera transformation from world to image coordinates is classically modeled as a matrix multiplication with the intrinsic and extrinsic matrices:

$$\mathbf{P}_{\mathbf{image}} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \cdot \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}} \cdot \mathbf{P}_{\mathbf{world}} \tag{3.2}$$

Intrinsic Camera Calibration. The camera intrinsics are traditionally represented by an upper triangular matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where, f_x, f_y are the x and y focal lengths and c_x, c_y are the x and y coordinates of the optical center in the image plane. However, in contrast to the perfect pinhole camera model, real-world cameras suffer from lens distortions. This has to be modeled by a non-linear warping of the image coordinates, which is modeled by five additional parameters $(k_1, k_2, k_3, p_1, p_2)$.

To find these parameters, we use our setup to capture multiple views of a common calibration pattern with an asymmetric arrangement of circles (Figure 3.8).

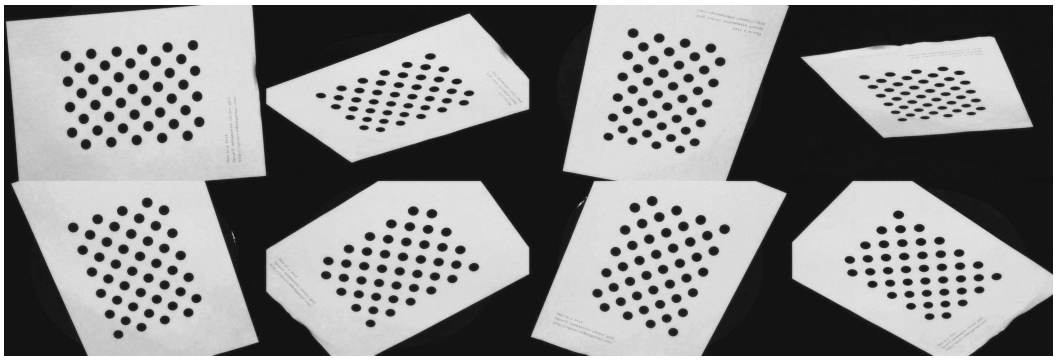


Figure 3.8: Different views of the calibration target, under modeling light, used to estimate camera intrinsics.

We run these through the standard camera calibration of the OpenCV library, which is based on an algorithm by Zhang et al. [Zha00]. The estimated distortion parameters are used to undistort raw camera images prior to all subsequent steps, and in the remainder, we always use the built-in OpenCV functions to project 3D points back into the images.

Perspective Unwarping. To undo the perspective, we need to know the camera position in world coordinates. We could directly use the camera pose sent to the scanning system during the acquisition. However, even for a very precise placement of the camera, any slight misalignment will result in a non-negligible deviation in the pixel grid of the camera. The most reliable way to proceed is to detect the sample holder frame in the images and use the image coordinates for the perspective unwarping.

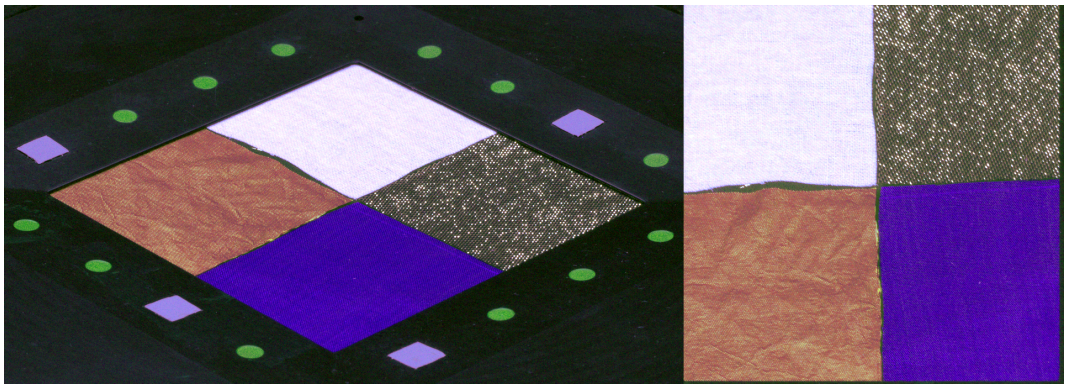


Figure 3.9: *Left*: Raw, unprocessed image. *Right*: Extracted texture within the target frame.

Our sample holder (Figure 3.9) contains target ellipses milled into the aluminum

frame and filled with green paint. We detect their position in the image using a blob detector on the green channel and eliminate any outliers based on the goodness of an ellipse fit and priors on the major axis scale of these ellipses, as the camera is always at the same distance from the frame. Although this process is not entirely infallible (it could for instance become challenging to rectify images of a textile with fluorescent green blobs in the texture), it has worked reliably in all the captures we have done so far. Potentially the angles from the capture script could be used to make the procedure more robust by estimating an approximate location of the ellipses and only searching in those windows, but we leave such small optimizations to future work.

We then use the centers of the fitted ellipses to estimate the edges of the frame by regressing a line through the three ellipses on each side. This in turn provides us with the four corners of the sample holder (Figure 3.10). Since the position of the corners is known in real-world coordinates, we can use it to obtain a more precise estimate of the camera pose. Additionally, unwarping the perspective transformation becomes a trivial application of a planar homography (Figure 3.9).

Figure 3.10: Detection of the sample holder frame (highlighted in yellow) in a photograph under diffuse *modeling* light (which causes the purple tint).

Unfortunately, this planar unwarping reaches its limits when the material sample has a meso-structure that creates variations in height (see Figure 3.11b). For any material that violates the planar surface assumption, the homographic unwarping will induce severe parallax effects in the registered textures. This means that when a point on the material surface does not lie on the same height as the sample holder plane, its projected position in the extracted texture pixel grid will shift around for

(a) Extracted (*cropped*) texture for a frontal view.

(b) Extracted and *unwarped* texture for a grazing view.

Figure 3.11: BTF texture rectification using a planar homography.

different camera poses. Instead, we would like every point on the material surface to always map to the same pixel position in the extracted textures.

To allow the system to capture such materials and to limit parallax effects, we require a more complex approach. The solution is to reconstruct the surface of the material, and unwarp each point individually, taking into account its height deviation from the sample holder plane. The ellipses are hence used to compute the camera pose and to sample texture values by projecting the estimated geometry back into the image.

3.2.3 Surface Reconstruction

Depending on the given material, different types of geometric representation can be appropriate. For opaque, thin materials, like wallpaper, a triangle mesh would be appropriate. For materials like wool or felt, however, it is difficult to find an adequate representation because there is no clearly defined surface. In the interest of improving our rectification without adding too much complexity to our model, we opt for the most simple geometric descriptor – we extend the (u,v) texture with a height field. We discard the idea of reconstructing an accurate surface, instead choosing to only use a geometric proxy that reduces parallax. A comparison between planar unwarping and the use of a proxy surface is shown in Figure 3.12.

For reasons of practicability, we decided to use only data captured with the gonioreflectometer for the geometry reconstruction. The setups at the University of

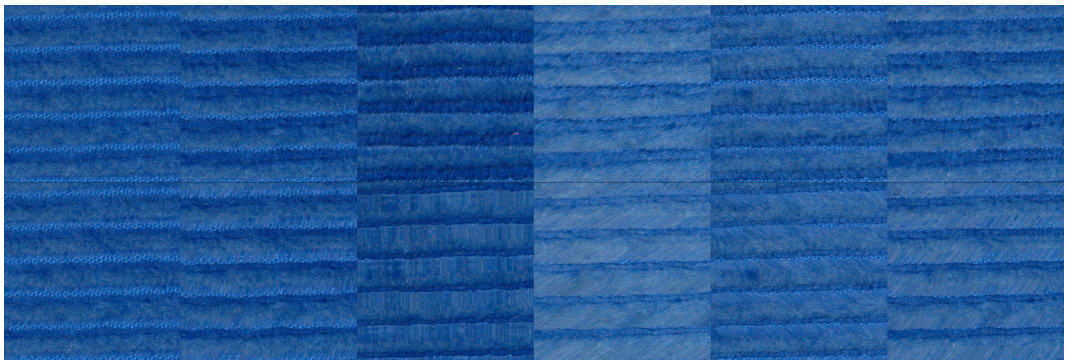


Figure 3.12: *Top*: Examples of extracted BTF textures using an average height plane. *Bottom*: Extracted textures using the estimated proxy surface.

Bonn use a reconstruction method based on structured light and super-resolution, introduced by Weinmann et al. [WSRK11]. In our setup, we only have access to a freely positionable camera and light source, and diffuse ceiling light (modeling light).

We use the additional dataset containing multiple views under modeling light is used to perform multi-view stereo. The difficulty of the task comes from the memory requirements – each raw image takes up to 100 Megabytes, which makes it memory-expensive to process the whole dataset. In order to simplify the procedure, we run the reconstruction using the rectified and extracted textures (1000×1000 texels). Our approach is based on the graph-cut formulation of the problem by Roy et al. [Roy99].

Following the maxflow-mincut formulation, we create a regular voxel grid around the textile sample in real-world coordinates. Each voxel is reprojected into each of the rectified images, and we apply normalized cross correlation on the 3×3 pixel neighborhoods in all of the rectified images. This provides us with a unique reprojection cost for every voxel in our graph. Along the z-axis (orthogonal to the sample holder plane): all the top voxels are connected to the source, all the bottom voxels connected to the sink. The voxels in between are connected to their upper neighbor (on the z-axis) and the cost of that edge is set to the voxel's reprojection cost. In the plane: each voxel is connected to his 4-neighborhood, the weight of these edges is set to an empirically determined smoothness constant.

Our implementation uses a solver [Kol10] implemented by Kolmogorov, based on the Boykov-Kolmogorov algorithm [BK04] to obtain the minimum cut through this

graph. By design, the algorithm is guaranteed to return a unique surface that defines a unique height for every (x,y) position on the grid, i.e. a height map. Depending on the chosen neighbor edge weight, the height map will be more or less smooth (see Figure 3.13).

Figure 3.13: *Left to right:* Close-up view of the material from a grazing angle under diffuse ceiling light, one extracted BTF texture and different reconstructed height maps for decreasing smoothness coefficients (0.002, 0.0002, 0.0001, 0.00002). All scalar height maps are gamma-corrected for display. **Dataset:** `whitemesh` from our database.

In order to make the method more robust, we use a multi-scale approach: we start off with a coarse resolution on the z-axis and a high smoothness constant. This will produce a very flat surface at the average height of the sample. We iterate this process with progressively finer voxel grids and lower smoothness costs, using the estimated height field from the previous step. This method consistently produces good estimates for the material surface, even for fuzzy materials like the corduroy in Figure 3.14.

3.2.4 Contents of a Processed Dataset, Notations, Variables

The final format of our uncompressed BTF data is a collection of rectified OpenEXR images of the textile sample, and a height map describing the geometry of the fabric. The height values are not expressed in metric units, but instead normalized with regards to the physical distance on the material, between neighboring texels. For the BTF textures, the respective light and view directions are encoded in the filename.

The raw merged HDR images take up around 400 Gigabytes for our datasets, which is reduced to 11 Gigabytes when the material textures are extracted at a resolution of 800×800 pixels. The processing can be parallelized on multiple cores – the HDR merging takes the longest, over 10 hours, while the texture extraction only takes two hours, using 8 parallel processes on our machine.

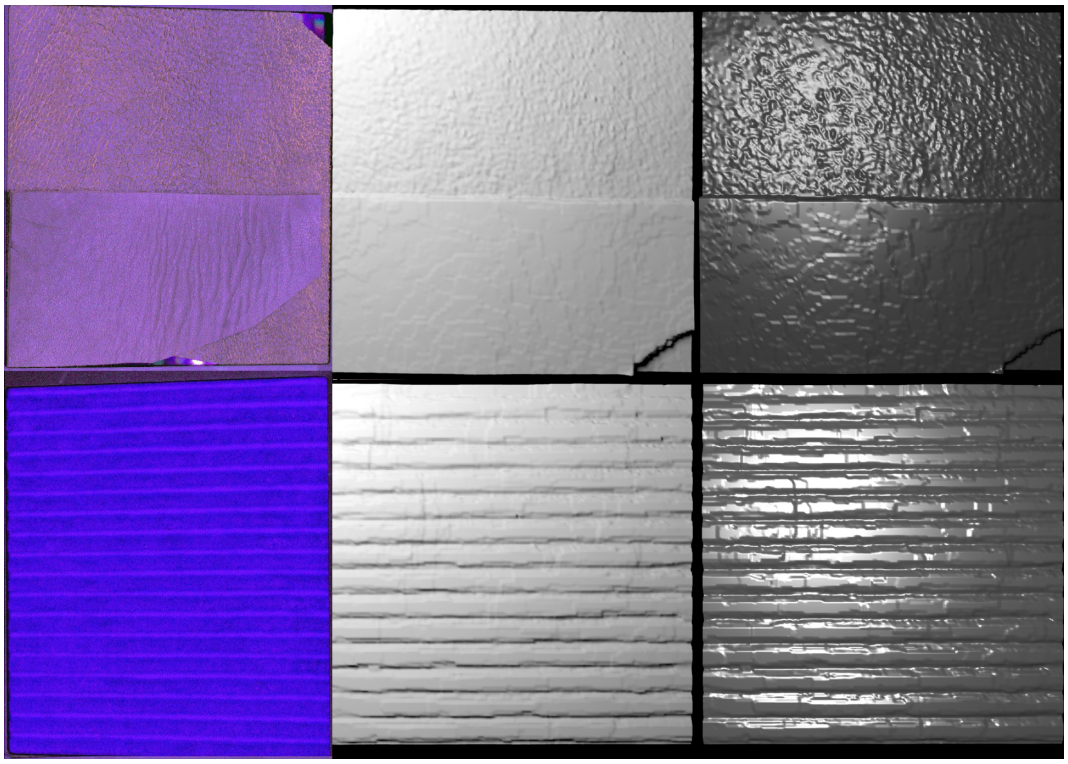


Figure 3.14: Examples of reconstructed surfaces for challenging materials (specular leather and coarse corduroy) under ceiling light (*left*) and renderings of the corresponding 3D surface reconstruction of a central area (diffuse in the *middle* and glossy on the *right*, for better visualization).

3.3 Rendering with a BTF

To render using the processed BTF dataset, we wrote a plugin for the open-source renderer Mitsuba [Jak10], that loads the entire BTF into memory. The final renderings are computed using unbiased pathtracing in Mitsuba, on scene geometries created in Blender [Com18], with uniform angular sampling of the reflectance function (no importance sampling).

The BTF provides reflectance values at a discrete set of angular configurations and a discrete grid of spatial positions on the material. At rendering time however, continuous spatial and angular positions are queried. These need to be interpolated from the discrete BTF. Spatial interpolation of the BTF is implemented as straightforward bilinear interpolation in between texels. Angular interpolation however is a more complex process. This is a common topic in BTF rendering, and we will demonstrate a more natural way to interpolate in the angular domain in Section 4.3.2.

Here we present the traditional approaches to angular BTF interpolation.

3.3.1 Angular Interpolation

In the angular domain, the ground truth reflectance values are only available at a discrete set of sampled positions. In between, values are unknown and have to be computed via interpolation, which inevitably introduces bias. There is no perfect solution to this issue, and depending on the selected method, the resulting values can be drastically different and change the appearance (see Figure 3.15). Additionally, the samples are not positioned on an orthonormal grid, which makes the interpolation much more tricky than in the spatial domain where samples live on a square pixel grid.

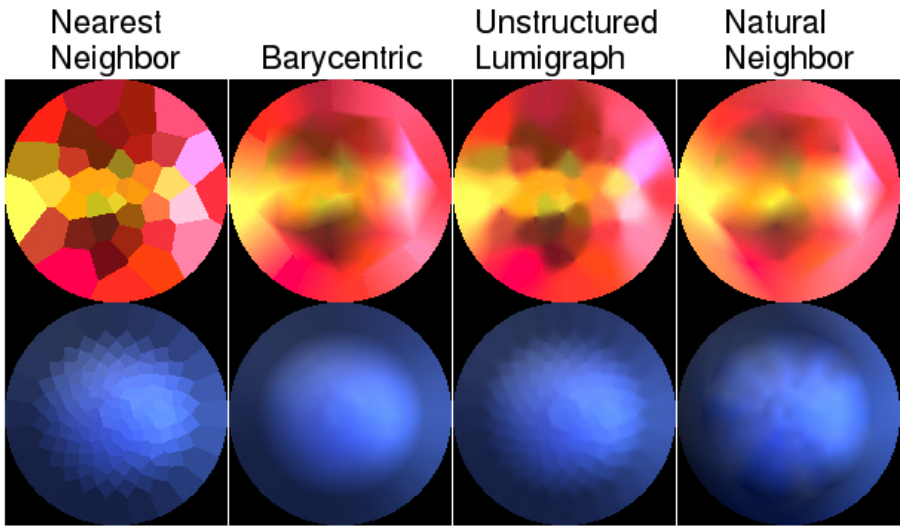


Figure 3.15: Angular plots of the reflectance of a single texel at orthogonal viewing, as a function of light direction. **Left to right**: different interpolation strategies. **Datasets**: `shantung` (top), `fabric04` from the Bonn UBO2014 database (bottom).

Figure 3.15 displays angular reflectance plots for two different materials. The values correspond to reflectance at a fixed viewing direction (orthogonal), for varying light direction – the light hemisphere is mapped to the x and y axis via stereographic projection. The top row corresponds to the `shantung` material from our captures, sampled at 40 different light directions. The bottom row corresponds to the `fabric04` material from the Bonn datasets, containing 151 different light directions. Since the shantung sample is more sparsely sampled and exhibits more specularity and

anisotropy, the influence of the interpolation strategy is very apparent.

In the first row, each angular value is taken from the nearest neighbor. This shows the ground-truth values we have at our disposal. Unstructured Lumigraph interpolation [BBM+01] remains visually closest to nearest neighbor interpolation, with subtle blending in between regions. Although this was originally used to compute penalties for *Unstructured Lumigraph Rendering* (ULR), this technique can also be used as a general scattered-data interpolation strategy. We compute the starting weights of all samples as inversely proportional to the distance to each sample position, and choose to use $k = 4$ final interpolation weights. ULR interpolation then determines final positive weights for the k closest samples; all other weights are set to zero. The key property of ULR interpolation is that for smoothly varying starting weights (distances here), the final blending weights also vary smoothly. This is noticeable in the lack of harsh discontinuities of the blending in the angular plot.

Barycentric interpolation relies on a Delaunay triangulation of the sample positions, followed by barycentric interpolation within each triangle. This produces characteristic artifacts between triangles, and outside the convex hull of samples, it is not clear how to extrapolate. In our case, we project onto the convex hull and interpolate along the line between the two closest samples of the hull. Hatka and Haindl use a variant of barycentric interpolation in their Blender BTF plugin [HH11]. Instead of computing the interpolation weights via barycentric coordinates on the triangle, however, they use the relative occupied portion of the tetrahedron created by the direction vectors going from the origin to the points on the unit sphere.

Natural neighbor interpolation [LG05] indirectly relates to barycentric Delaunay interpolation, as it uses the Voronoi tessellation of the point set. The artifacts are of similar nature to the barycentric interpolation, although the result seems smoother and better behaved. The interpolating weights are calculated by finding how much each neighboring area would be reduced if the considered point was inserted into the Voronoi tessellation.

The problem of the choice of angular interpolation method shows one of the inherent difficulties of dealing with BTFs. The notion of *ground truth* only exists

on the discrete set of angles and positions that was captured. Any comparison on renderings will be biased by the choice of interpolation strategy, and *no rendering can be considered ground truth, even if solely relying on the original BTF data*. Quantitative evaluations can only be performed on the very sparse discrete set of known observations, any other evaluations will be qualitative. Widely used perceptual error metrics such as structural similarity [WBSS04] cannot be applied without bias. The use of more perceptually relevant quality measures has been explored [FCGH08, JWD+14], but most works involving BTFs still use standard, perception-agnostic losses.

3.3.2 Parallax Mapping

In the processing pipeline, the BTF textures were extracted according to an estimated proxy surface in order to minimize parallax. This means that the material points that the texels refer to do not lie on a plane, but on a 3D surface. Accordingly, during rendering, the mapping of the textures onto the object mesh needs to use the same surface to correctly project back. However, this is different to bump or displacement mapping, which are commonly used techniques to incorporate meso-structure of materials. In our case, we only want to undo the parallax compensation applied when unwarping the textures, which consists in perturbing the (u,v)-coordinates of the textures, without modifying the underlying mesh, the surface normals, or the shading (see Figure 3.16: modifying the geometry adds shadows, which we want to avoid).

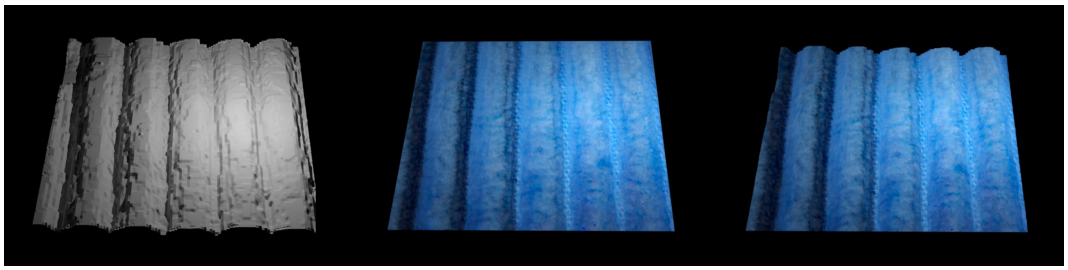


Figure 3.16: *Left*: Rendering of a diffuse plane using the Mitsuba height field plugin, which displaces the surface according to the height field and computes corresponding shading normals, introducing shadows. *Middle*: BTF rendering without parallax mapping. *Right*: Our parallax mapping only displaces the (u,v)-coordinates but the shading is already contained in the BTF data. The shading frames at each surface point remain in accordance with the macro-scale geometry, instead of the height field.

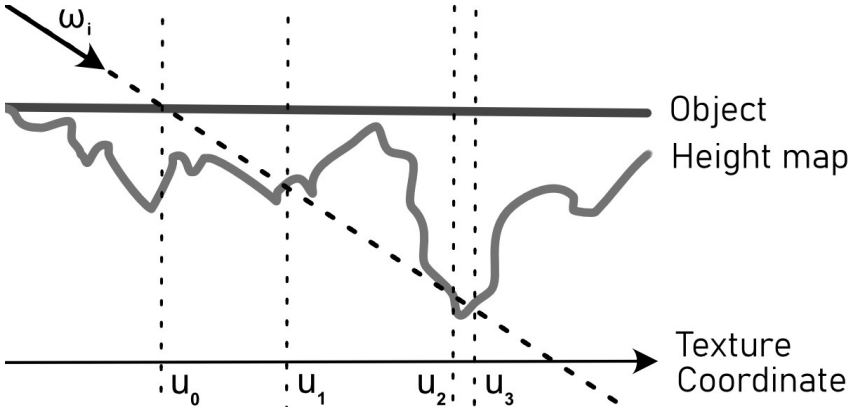


Figure 3.17: Parallax mapping, in 2D for simplicity.

We implement a variant of relief mapping (similar to [OP05]), where only the (u,v)-mapping is warped, without introducing changes to the shading. The material's height map is virtually situated underneath the object surface. Without loss of generality, we represent the problem in two dimensions in Figure 3.17. Every time the BTF shader is called, we compute the intersections of the incoming ray with the height map in the given setting. Internally, this is done by joining the neighboring points of the height field into triangles and computing ray-triangle intersections. The first intersection is used to update the (u,v)-coordinate pair used for shading. In the diagram, this means that the original coordinate u_0 will be replaced by u_1, which is the actual point on the material that would be seen from the direction ω_i given the corresponding height field.

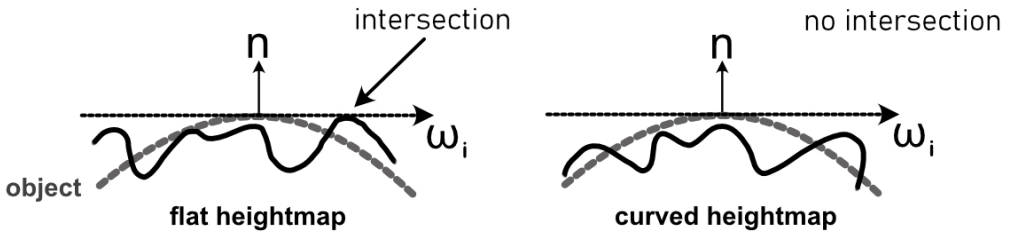


Figure 3.18: Problem with standard parallax mapping at silhouettes. *Left*: The viewing ray intersects the object geometry and finds an incorrect intersection with the planar height field. *Right*: The height field should ideally follow the topology of the object, in which case there is no intersection with the viewing ray. This is necessary to create silhouettes.

This procedure allows us to reverse the projection that was applied during BTF

data processing (unwarping) and recreate the correct perspective. However, in this setting, it is assumed that the object surface is planar at a macroscopic scale. At object silhouettes (Figure 3.18), this planarity assumption breaks.

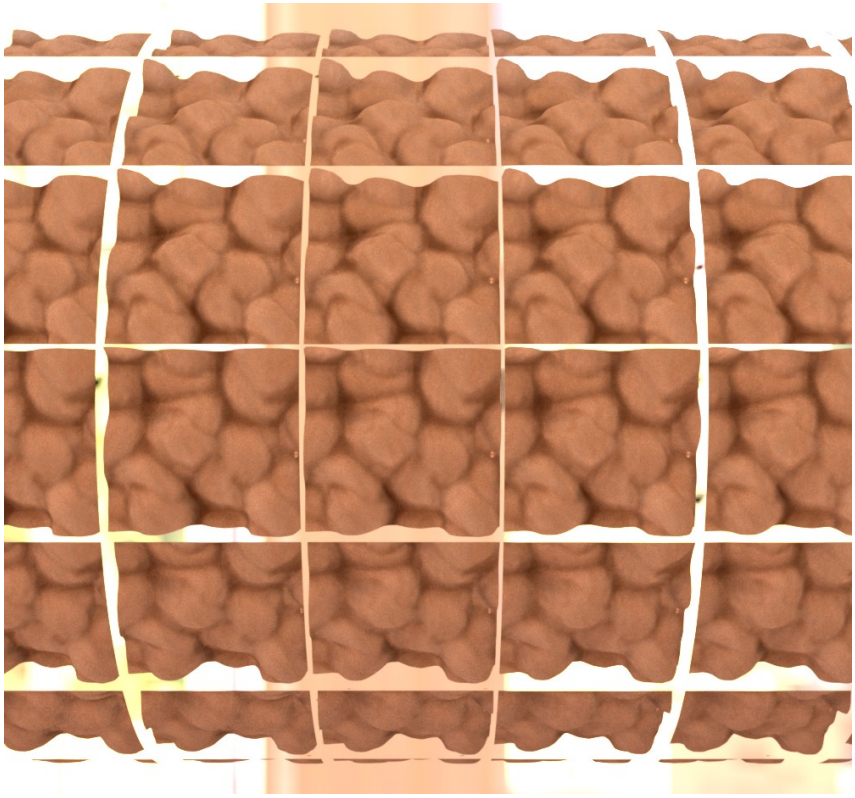


Figure 3.19: Another issue with parallax mapping when the texture is tiled, using `carpet05` from the Bonn UBO2014 database.

The silhouette is determined by whether a ray intersects the object geometry or misses it, so with straight-forward relief mapping, the shape of the silhouette is not affected by the underlying height map: since the ray hits the object mesh, it cannot be parallel to the tangent plane, meaning that it will eventually intersect the globally planar height field, potentially very far from the initial intersection, thus not reflecting the curved topology of the object. Alternatively, if the texture is small and tiled, and we only search for an intersection within the current tile, we obtain holes between neighboring tiles, as shown in Figure 3.19.

To achieve completely correct silhouettes, the height field would need to be warped to follow the topology of the object surface (Figure 3.18, right). However, this is complicated and expensive to perform at runtime, so we implement a very

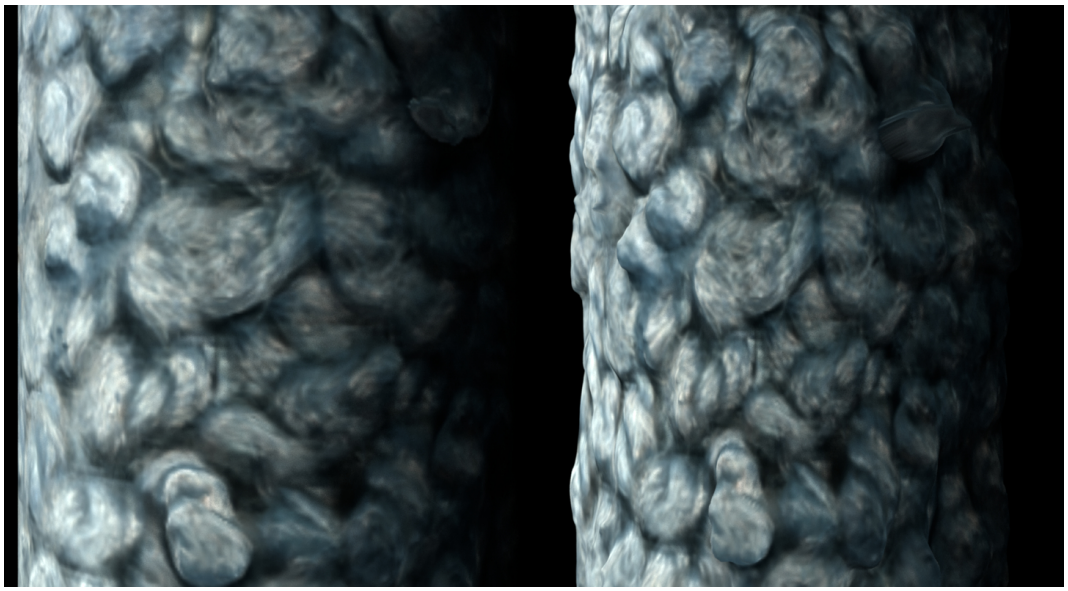


Figure 3.20: Rendering of `carpet12` from the Bonn UBO2014 database, mapped on a cylinder, without and with parallax mapping.

simplified alternative: the height field remains locally planar, but at locally convex points on the surface, we restrict the search for a height field intersection. At the initial ray-mesh intersection, the mean curvature of the object geometry is computed to calculate the radius of a mean tangent sphere. We only search for a height field intersection within that distance. Although this is a very crude approximation, it produces plausible silhouette impressions as long as the height variation in the texture is orders of magnitude lower than the topological variations in the mesh. The results of this silhouette rendering are shown in Figure 3.20.

To conclude, we propose a simple and efficient way to recreate the parallax at rendering time and to create a silhouette impression, only via operations on the (u,v)-coordinates inside the BTF texture shader. However, this is a very simplified and approximate approach that is only viable in the typical BTF setting. The height map that describes the material's meso-geometry should only contain very subtle height variations that are at a much finer scale than the object geometry. The parallax and silhouette mapping will fail or produce incorrect results if the meso-structure is too coarse compared to the mesh topology. In the common BTF paradigm, however, the material is macroscopically planar; any coarse surface variation is modeled in the object mesh, while the meso-structure only models very fine details.

The goal of our 3D reconstruction is not an exactly accurate micro-geometry, as this is often impossible to model with a surface. Instead, the goal is to minimize parallax in the BTF which will allow more efficient compression and reduced blurring and ghosting in the interpolation, and possibly further applications such as editing or synthesis. Given this meso-structure proxy, an approximate silhouette with efficient parallax mapping is sufficient to create the visual effects and fidelity that we desire.

3.4 Results and Discussion

In this Chapter, we presented our custom high-resolution BTF acquisition system and processing pipeline, designed to scan delicate materials [RRW19]. To conclude, we recapitulate the structure of a typical output of our pipeline and outline potential improvements. Using our custom BTF shader in Mitsuba, we can render the captured materials, with silhouettes, on arbitrary object geometries, in arbitrary scenes. Further rendering features, such as seamless tiling of the BTF texture or importance sampling of the BTF to speed up rendering convergence, are left to future work.

3.4.1 Output of the Processing Pipeline

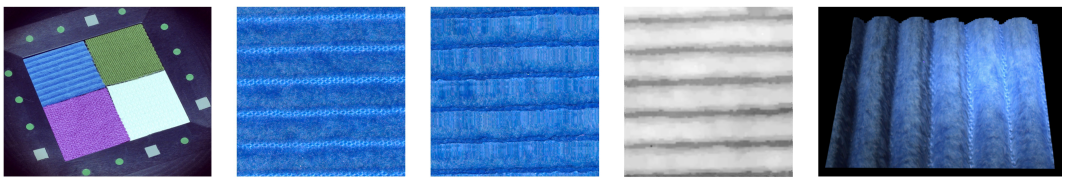


Figure 3.21: *From left to right*: Unprocessed HDR image from the raw dataset, extracted and rectified texture for an **orthogonal view**), extracted and rectified texture for a **grazing view**, reconstructed height field, BTF rendering with parallax mapping under point light illumination.

Figure 3.21 provides an overview of the output components from our material scanning pipeline. To speed up acquisition times and exploit the high spatial resolution of our images, we scan four textiles at once. Using the images under modeling light, we reconstruct a proxy height field. This geometry is used to extract and rectify the textures, of which we show an orthogonal and a grazing view for the corduroy sample. When extracting the grazing view, the footprint of each pixel in the original raw image is not square in the rectified view anymore. This anisotropic blur is very

noticeable in the extracted BTF texture. Finally, using the BTF and the height field for parallax mapping, we can texture any mesh with the material and render it with a realistic silhouette impression (Figure 3.21, rightmost image).

3.4.2 Capacities and Limitations

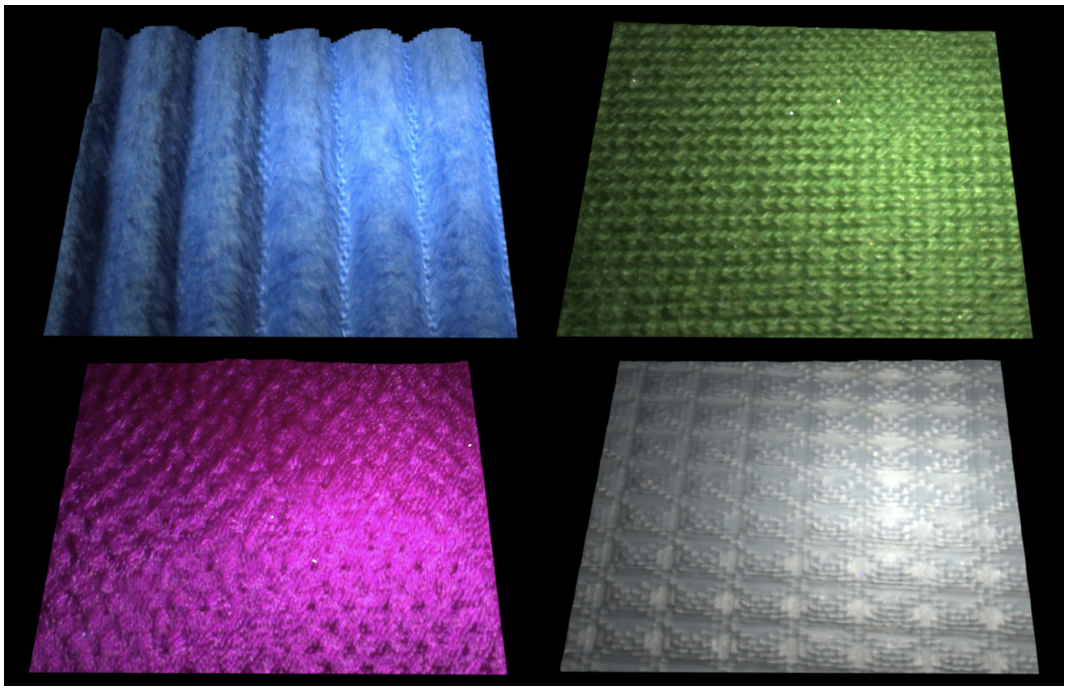


Figure 3.22: Rendering of a BTF-mapped square with materials captured with our setup: `buecord`, `greenwool`, `purpleweave` and `whitemesh`.

We show renderings (Figure 3.22) of the same scene (a single instance of texture mapped on a plane under a point light) with four BTFs captured simultaneously using our setup. The blue corduroy sample (top left) is at the limit of coarse height variations that the system can capture. Nevertheless, the silhouette is visible and there is no excessive blurring, which shows that the estimated geometry is plausible. If no geometry was used for the rectification, each spatial point on the sample would move around so much in the BTF textures, that the rendering would be completely blurred out because of the interpolation in texture space.

The color fidelity is high on all samples and the characteristic visual softness of textiles is not lost. The white mesh (bottom right) is qualitatively the least faithful, with a more diffuse and solid, almost metallic visual feel. The many specular glints

present in the original material are either not captured at the measured angles, or removed by the interpolation procedure at rendering time. The other two materials (green wool and purple sating weave) are slightly less specular and contain many other effects that are usually challenging to model (anisotropy, inter-shadowing and masking) – we manage to reproduce their appearance convincingly. For a catalog of all materials captured with our system, please confer Appendix B.

The main limitation of the capture system is currently acquisition time. The bottleneck is caused by the comparatively low power of the light source: a stronger emitter would allow quicker measurements and hence a denser sampling of the light and view hemispheres. Due to the relatively low illumination provided by the LED in our current setup, the shutter times of the HDR stack reach up to tens of seconds. We can slightly mitigate this by increasing the ISO, however this comes at the cost of increased noise in the measurements, which has strongly degraded the quality of some captured datasets. On the other hand, the model we present in Chapter 4 is able to combat this to some extent: since the noise is completely random, our model is unable to learn it, which effectively results in filtering out the noise.

Chapter 4

Neural Appearance Model for BTFs

The main challenge in BTF modeling 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 are able to consider more complex nonlinear dependencies in the high-dimensional distribution of BTF values.

In this Chapter, we propose a new strategy that compresses BTF data using an asymmetric encoder-decoder network architecture and compare it to traditional PCA (see Figure 4.1). Our approach is 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 non-linearly 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.

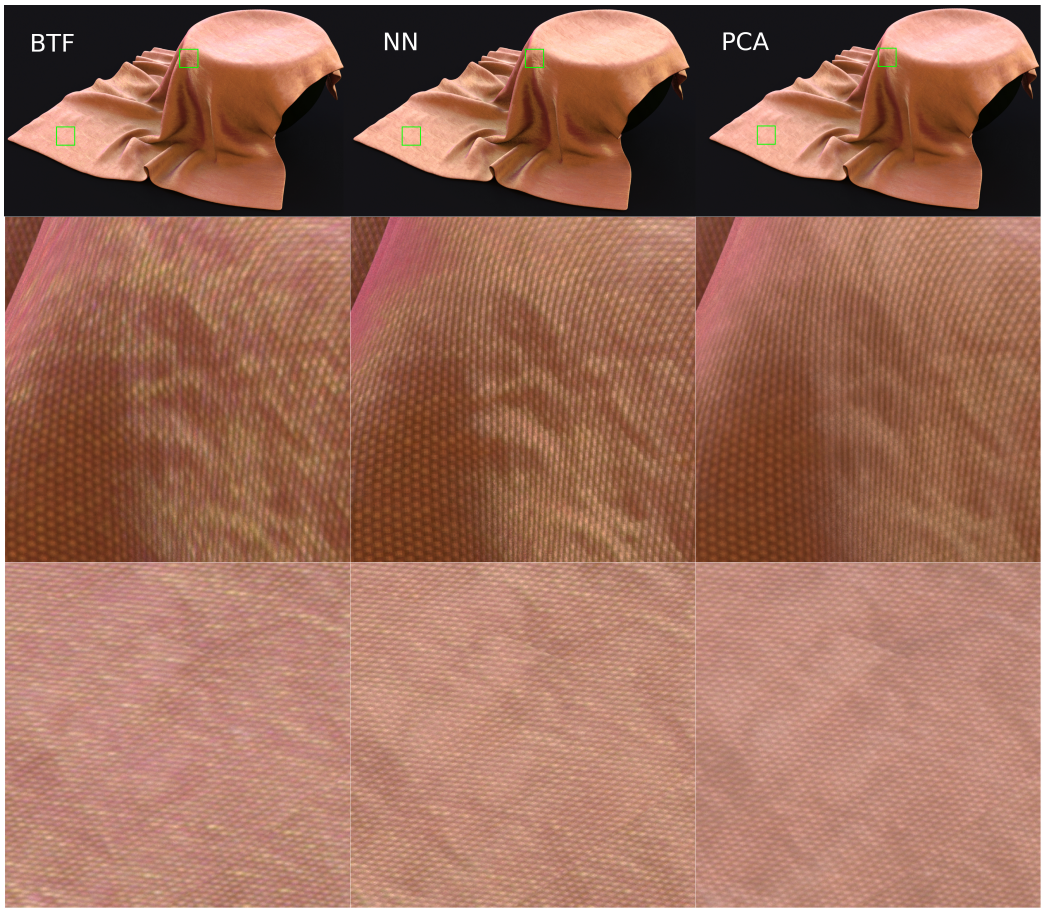


Figure 4.1: Our neural BTF model (middle) in comparison to rendering with the original or the PCA-compressed BTF, for a challenging specular fabric (`shantung`) captured with our setup. At the same compression ratio, our neural network-based BTF approximation is able to capture subtle surface variations and anisotropy that are blurred out by principal component analysis-based compression.

4.1 Existing Methods for BTF Compression

As a discrete set of samples from a 6-dimensional function, the BTF suffers from the curse of dimensionality, making not only dense acquisition, but also dense storage of such high-dimensional data prohibitive. Data compression has hence been a major research topic in BTF modeling community [MBK05, FH09b]. In addition to the background presented in Chapter 2, we now analyze different approaches to data compression, in the specific context of BTFs.

4.1.1 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 direc-

tion 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.

4.1.2 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 parametric 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 fabrics that do not do not occupy a clearly defined two-dimensional surface.

4.1.3 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-Lambertian materials.

Drawing from this background in BTF compression, and the recent advances in neural appearance modeling (see 2.4), we decide to explore applications of deep learning to BTF modeling. We harness two essential properties of auto-encoders to build an effective BTF material representation: an efficient adaptive latent representation with high compression ratio in conjunction with straight-forward interpolation, by making the decoder a continuous function of lighting and viewing directions.

4.2 Neural BTF Modeling

Partial evaluation of the BTF at a given surface position (texel) yields a 4-dimensional function $f(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ 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 lighting effects such as subsurface scattering.

In the discrete setting $\boldsymbol{p} = (x, y) \in \mathbb{N}^2$ is a pixel coordinate, so the ABRDF turns into a length-n vector of RGB values, with one entry for each captured combination of lighting and viewing angles. We pre-process 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.

4.2.1 BTF Compression using PCA

PCA-based compression techniques of BTFs typically arrange all ABRDFs as columns of a matrix $\mathbf{A} \in \mathbb{R}^{s \times n}$, where s is the number of BTF texels. The matrix is subsequently decomposed using a *Singular Value Decomposition* (SVD), which expresses the data a sum of outer products

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{4.1}$$

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 4.2 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 s texels, the matrices $\mathbf{U}$ and $\mathbf{V}$ will have $s \times k$ 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.

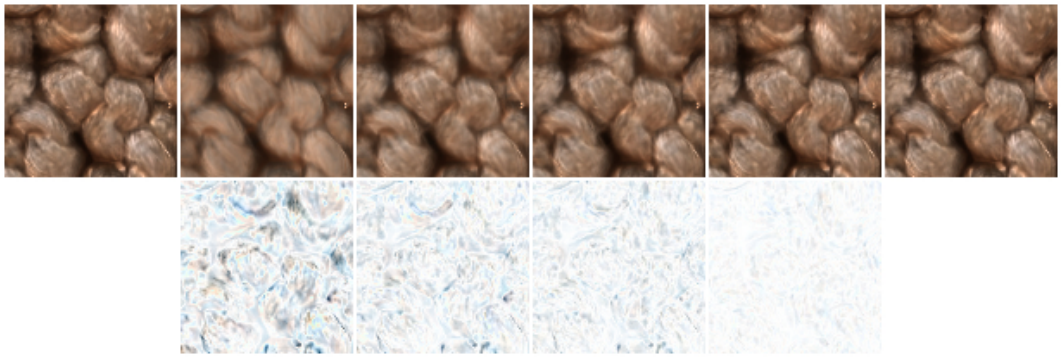


Figure 4.2: Top: Input BTF and PCA approximations using 8, 16, 32, 64, 128 singular values. **Bottom:** Negative image of difference magnitude, multiplied by 2 and gamma corrected. **Dataset:** `Carpet05` from [WGK14].

4.2.2 Neural BTF Compression

Our neural BTF compression strategy is inspired by auto-encoder networks [Ben09]. 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 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 auto-encoders 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 auto-encoder 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. Additionally, the ABRDF only provides reflectance values at the angles sampled during the BTF acquisition, so the queried angle pair would still need to be interpolated from the discrete set of ABRDF values. 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 4.3 illustrates our encoder and decoder, which we now discuss in turn.

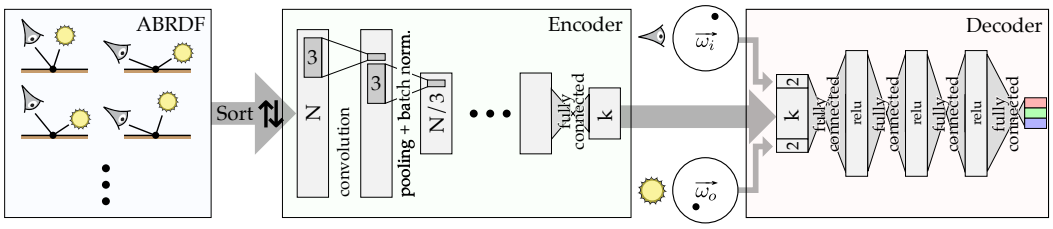


Figure 4.3: Diagram of our full neural pipeline architecture.

Neural Pipeline. 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 down-sampled 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 component-wise nonlinearities. The last layer outputs a single RGB color.

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 angular resolution of the input dataset: for the Bonn datasets (22,801 angles), we repeat the illustrated down-sampling layer sequence four times (each reducing the amount of data by a factor of 3×). For our own BTF datasets, which have twice the spatial resolution but only around 1,500 angular configurations, a single down-sampling block is sufficient.

This encoding architecture is comparatively simple and does not exploit the angular positional information in any manner. Additionally, the encoding network is very light, the number of parameters is comparatively low, meaning it cannot learn an overly complex encoding function. However, this is sufficient in our first approach to the problem, where we focus on the representational power of the decoding architecture. In this Chapter, we develop our neural BTF model, which is centered on the latent vectors and the decoder network. Techniques to fully exploit the possibilities the encoder network are explored in Chapter 5.

Decoder network. Our decoder architecture is shown on the right side of Figure 4.3. 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

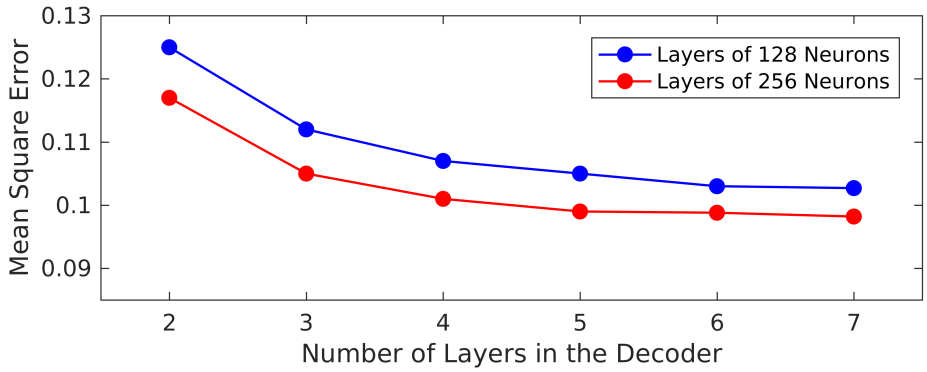


Figure 4.4: Test loss for identical networks except for the number of layers in the decoder. **Dataset:** `shantung` (from our database).

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 4.4 shows the L_2 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 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 $\mathbf{V}$ 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 4.5 shows the reconstruction error as the number 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

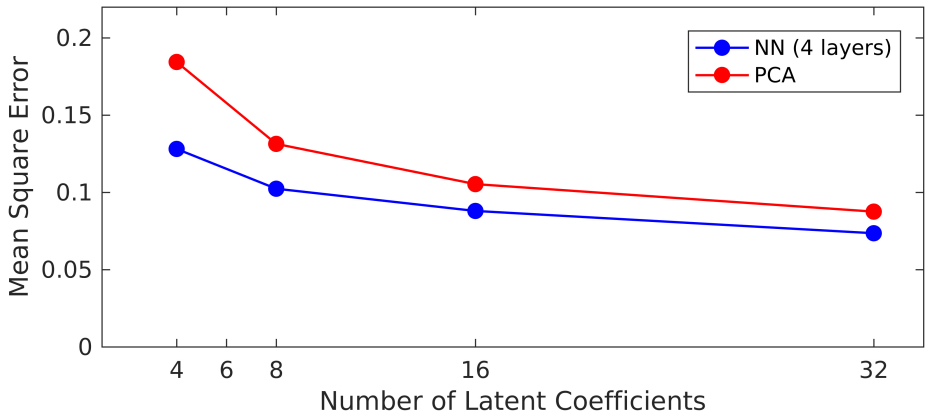


Figure 4.5: Error (on the preprocessed dataset) as a function of latent coefficients, for a PCA decoder matrix and a neural decoder of the same size. **Dataset:** `shantung` (from our database).

per texel as a simple SVBRDF model.

4.2.3 Comparison of Compression Performance

Our evaluation is based on two sources of data: BTFs from our own capture setup shown in Chapter 3 and publicly available datasets from the University of Bonn, UBO214 [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 the University of Bonn 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 4.1 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 4.2 reports reconstruction scores on a wider range of datasets from the Bonn database [WGK14].

We compare the compression performance of our representation to that of

	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) 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**
Compress. Ratio	**512.5**	**512.5**	**1090**	**1090**	**1090**	**1090**
PCA(16) 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**
Compress. Ratio	**256**	**256**	**545**	**545**	**545**	**545**
Ours(8) 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**
Compress. Ratio	**512.5**	**512.5**	**5897**	**5897**	**5897**	**5897**

Table 4.1: Reconstruction error (RMS), size of the components to store (in thousands of coefficients) 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.

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 auto-encoder network.

We perform comparisons on cropped datasets of 256×256 texels for our BTFs, and 100×100 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).

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

Table 4.1 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 B.3) 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.

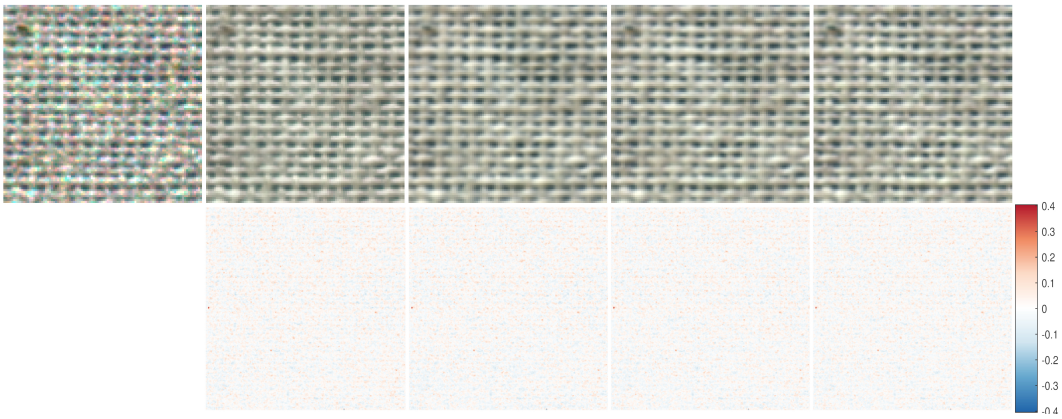


Figure 4.6: Top row (from left to right): Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles: 0, 90, 182.4, 35.1. **Bottom row:** Signed L_2 error of the reconstructions compared to the ground truth. **Dataset:** `cotton` from our database.

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 4.7, 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 UBO2014 datasets have already been pre-compressed using PCA with 101 coefficients – in the worst case, our comparison on these datasets is favorably biased towards PCA. Nevertheless, our approach achieves an over tenfold compression 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

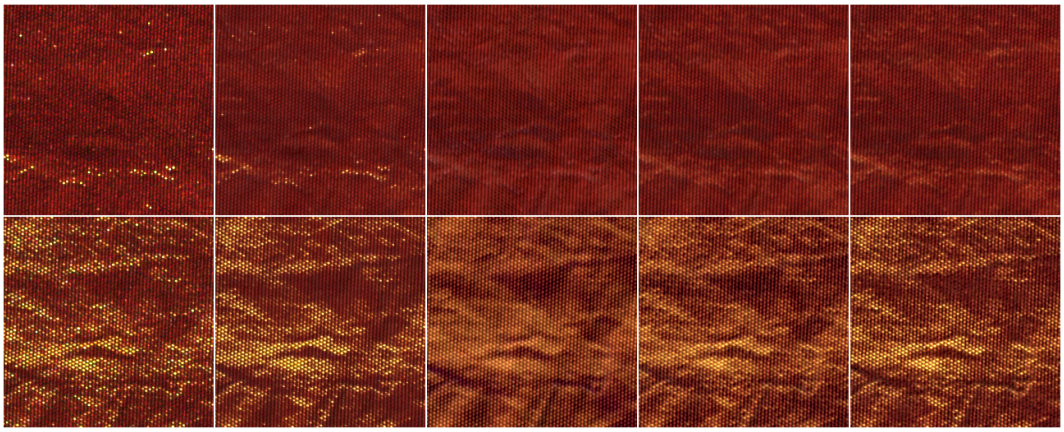


Figure 4.7: Columns from left to right: Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles in the **top row:** 0, 90, 182.4, 35.1. **Bottom row:** 222.5, 77.2, 44.9, 33.4. **Dataset:** Shantung from our database.

wide margin, at an over 5-fold increase in compression.

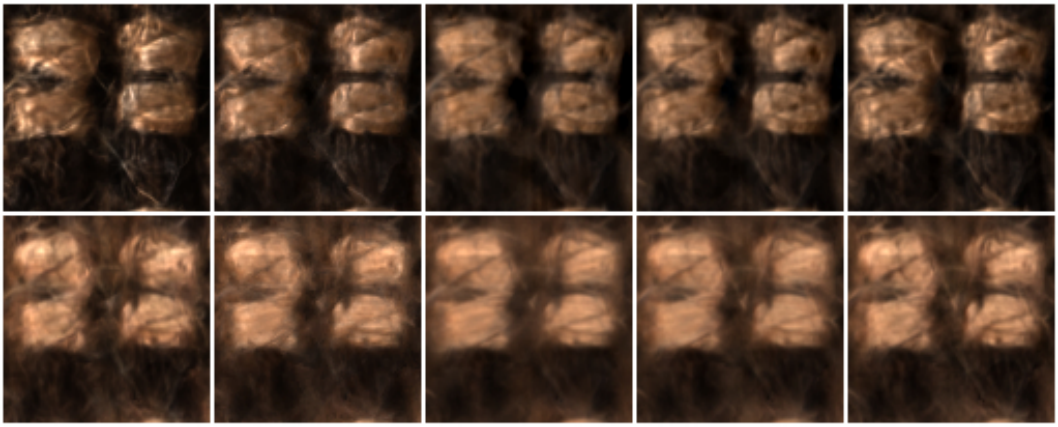


Figure 4.8: Columns from left to right: Ground truth, Neural Network, PCA (8), PCA (16) and PCA(32). View/Light azimuth/elevation angles in the **top row:** 0, 90, 180, 45. **Bottom row**: 270, 30, 0, 90. **Dataset:** `carpet07` from the Bonn Database.

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 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 4.7 shows).

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

Table 4.3: Comparison to Ruiters et al. [RK09] on the `Pulli` dataset.

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 4.3 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.

4.3 Learned Angular Interpolation

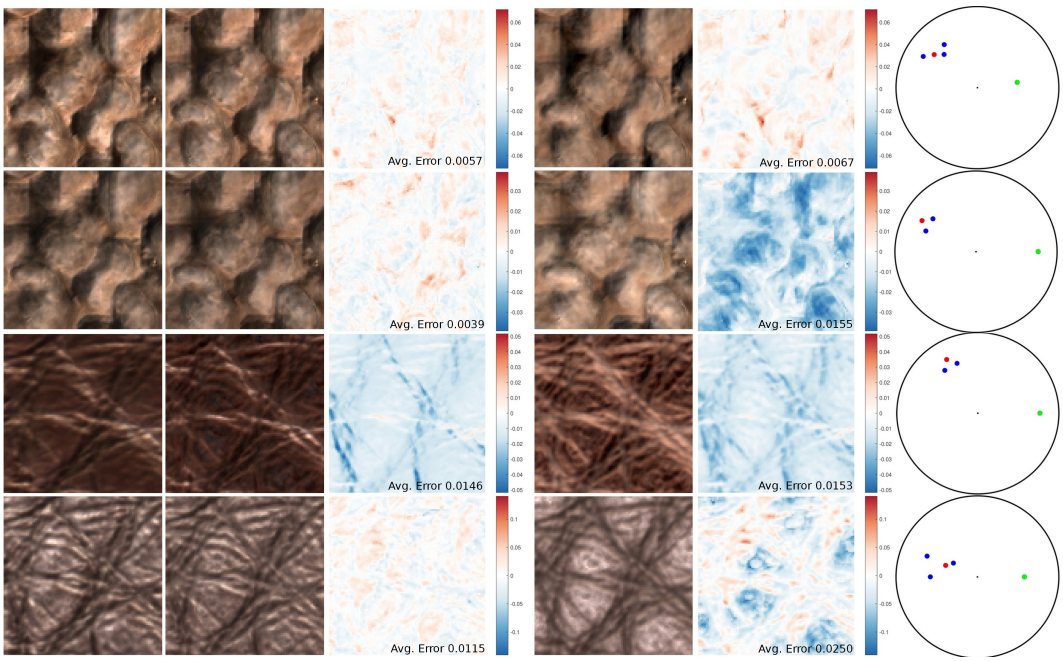


Figure 4.9: *From left to right*: ground truth, reconstruction with our method and with PCA, with error images. Angles displayed on the unit disk: view direction (green), light direction (red), and light directions used for PCA interpolation (blue). **Datasets:** `carpet05` and `leather04` from [WGK14].

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 (see Section 3.3.1). 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 in the angular domain that lie between measurement locations (*interpolation*), or even outside of the convex hull of the measurements (*extrapolation*). To evaluate this objectively, we train on subsets of the original BTFs and use the remaining textures as ground truth images for evaluation.

4.3.1 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 4.9 shows comparisons for 2 datasets from the UBO2014 database. 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

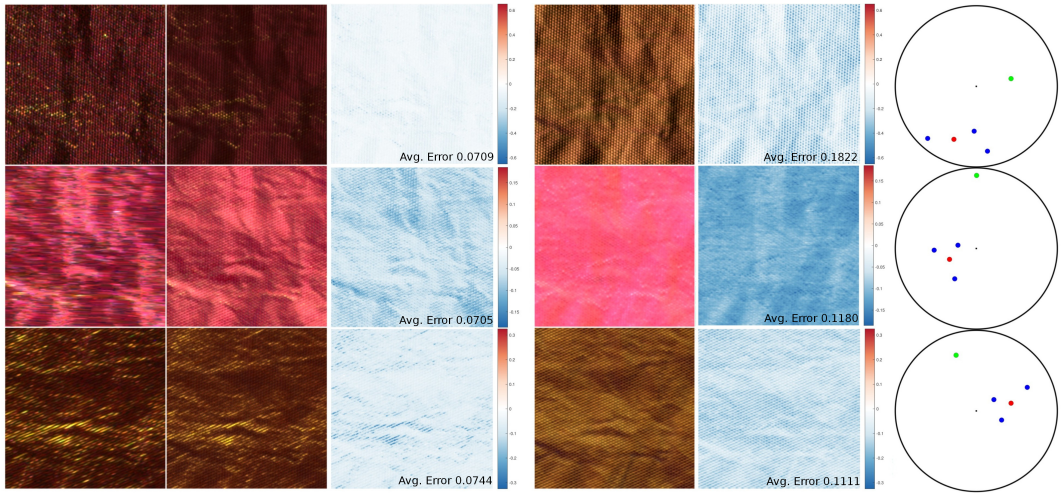


Figure 4.10: *From left to right*: ground truth, reconstruction with our method and with PCA, with respective error images. Angles are displayed on the unit disk: view direction (green), light direction (red), and light directions used for PCA interpolation (blue). **Dataset:** `shantung` (from our database).

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 4.9 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 4.10 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 4.9 and 4.10 only display characteristic examples for these materials, but we also generated interpolated images for many other configurations that are provided in the supplemental material of the publication [RJGW19].

4.3.2 Angular Plots

Another way of evaluating the interpolation performance is to consider a single texel for a fixed viewing angle (Figure 4.11), reducing the dimensionality from 6D to 2D. We plot the dependence on the lighting direction under stereographic projection onto a disk. For both materials, extrapolation 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 two 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. Similarly, for the `fabric04` dataset, our network produces the most plausible output, while all other approaches extrapolate unrealistic sheen in the region outside the convex hull of angles.

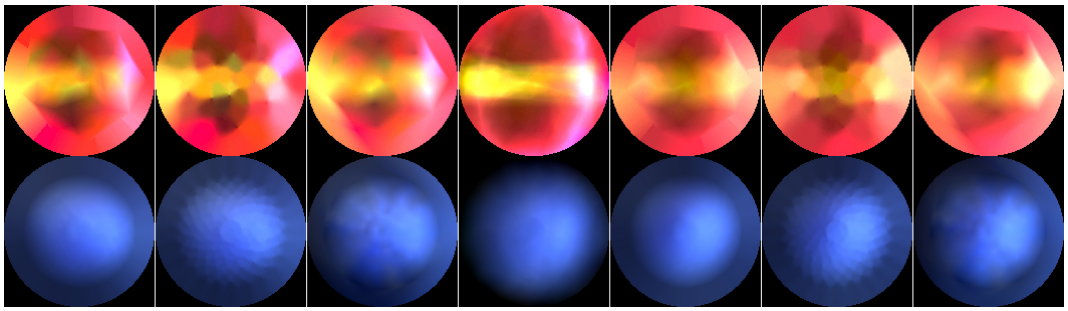


Figure 4.11: Angular plots of ABRDF slices (fixed view, varying lighting) with different interpolation strategies. *From left to right*: Original BTF (barycentric, lumigraph, natural neighbor), neural network, PCA (barycentric, lumigraph, natural-neighbor). **Dataset:** `shantung` (top), `fabric04` (bottom).

Figure 4.12: Renderings with Mitsuba's pathtracer (2000×1200 pixels, 128 samples per pixel, 10 processes, uniformly sampled BSDFs, environment lighting). Left to right: original BTF, our network and PCA. **Datasets** (left to right): `leather04`, `fabric04`, `cotton`.

4.3.3 Rendering Comparison

When comparing compression scores, we compared the reconstruction error with the ground truth at configurations where the appearance has been sampled. However, it is much more important to evaluate how the model behaves in between these samples, which only constitute a discrete set of a continuous space. As the renderings 4.12 suggest, our approach arguably behaves better in that space than even the original BTF, making the subjective improvement much higher than the scores established in Table 4.1.

We use the open source renderer Mitsuba [Jak10] to create high-resolution renderings of a cloth draped over a bowl (Figure 4.12). 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.

In Figure 4.1, we show additional close-up insets for a challenging material (dual-colored, anisotropic silk). Although we do not have access to a ground truth for this comparison, we can argue that the results generated with our network look much more plausible. The shadows, occlusions, and specular highlights tend to be blurred out by the linear interpolation with the original data, which explains the lower dynamic range and overall dulling of the material's appearance. Our approach however preserves these subtle effects and creates a more realistic image.

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 UBO2014 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 acceptable, but we do not claim superior rendering speeds. For additional visualizations, we refer to animations in the supplemental material of the publication [RJGW19]. 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.

4.4 Discussion

Choice of Baseline. Dimensionality reduction is a vast field in research with many contributions long before BTFs were introduced and generated a need for compression of ABRDFs. Early work on BTFs simply arranged the data in 2D matrix form and applied SVD [KMBK03]. Most factorization approaches are variations of this, with different pre-processing, but eventually applying SVD. Sattler et al. [SSK03] separate each view direction into a different matrix that is compressed with SVD, while the separation performed by Kim et al. [KCL18] acts in the reflectance data domain, between diffuse and glossy components. Mueller et al. [MMK03] propose *local PCA* – the ABRDFs are randomly clustered and PCA is applied independently to each region. Iteratively, the clusters are updated and the regions re-arranged to optimize the error of each PCA compression, This adaptive approach significantly outperforms naive PCA in terms of compression. Finally, the Bonn UBO2014 database is compressed using *Decorrelated Full Matrix Factorization* (DFMF) [Mül09]. The BTF data is pre-processed by converting from RGB into the YUV colorspace, applying a log-transform to the Y-channel (brightness), and dividing the U- and V-channel by the Y-channel. An SVD decomposition is then computed separately for each channel.

A few more involved dimensionality reduction methods have been explored; the common point is the preserved tensor representation of the BTF. Contrary to the previously presented SVD-based methods, the BTF is not flattened into a 2D matrix throughout the compression, allowing a more adequate treatment of the spatial and angular dimensions. Ruiters et al. [RK09] and Wang et al. [WWS+05] both

employ K-SVD based compression, which is a generalization of K-Means clustering, alternating between a sparse coding step that optimizes the sparse representation vectors and a dictionary update step. While this type of hierarchical dimensionality reduction exhibits good reconstruction at high compression ratios, the compression algorithm and the decompression computations are also more complex, which is something we would like to avoid as BTF rendering is already computationally expensive. A comparison of our compression against K-SVD is shown in Table 4.3.

In practice, however, most BTFs are compressed with SVD or a variation of it, because of its simplicity (both in compression and decompression) and efficiency. We chose PCA of the *log-transformed* and *whitened* BTF matrix as our baseline for comparisons because of this popularity and practicability, and because it is representative for most BTF factorization techniques that use some variant of SVD. Beyond performance comparisons, we want to highlight the methodological difference between our technique and linear dimensionality reduction: our neural encoding combines the factorization into latent maps with the continuous parametrization of SVBRDFs, entirely sidestepping the otherwise tedious matrix decompression and linear interpolation.

Further compression tricks like vector quantization [HFM10] can be applied to our neural latent maps and are orthogonal to our encoding method.

Choice of Encoder Architecture. In this Chapter's per-material training scenario, the exact architecture of the encoder has a limited effect on the learning and the results, and many other choices could have been made. The only task of the encoder is to map the ABRDFs to latent codes that can parameterize a successful regression by the decoder network, which overfits to the reflectance profiles.

The convolutional encoder fulfills a very simple task: the 1D convolutions downsample the input signals to a lower-dimensional representation (which still contains several hundreds of features), and the final fully-connected layer projects this into latent coordinates. Effectively, the encoder performs a classification of input texels into different latent regions. The main part of the learning is performed by the decoder, that effecticely remembers the most prominent appearances and learns to

reproduce them given a latent cue by the encoder.

We observed that adding more fully connected layers to the encoder does not improve reconstruction, so adding complexity to the encoding pipeline does not improve the overall result here. However, this simple classification of input texels still helps to simplify the learning a lot, as similar ABRDFs are not necessarily close to each other in the spatial domain, and the encoder provides a fast way of pre-processing the ABRDFs into coherent low-dimensional latent vectors.

One alternative would be to remove the encoder and initialize the latent vectors randomly. This is commonly referred to as an *auto-decoder* architecture, where the latent embeddings are optimized jointly with the decoder. While this would probably generate the same quality of results after training, it comes with the disadvantage that there is no quick way to convert new inputs to their latent embedding without optimizing. With our encoder, however, this is a matter of just one forward pass through the convolutions, done in milliseconds.

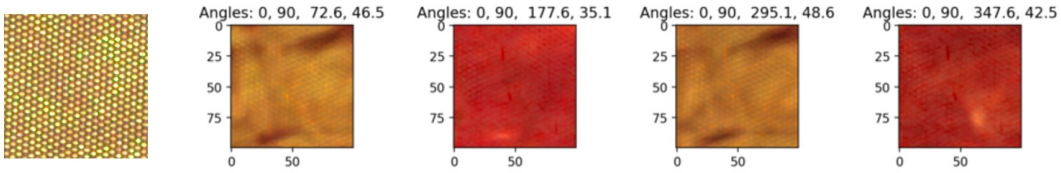


Figure 4.13: Visualizations of textures generated by a decoder (4 hidden layers of 256 neurons) trained on the uv-coordinates directly), for varying angles, compared to the ground truth texture (left). The reconstructions contain the angular anistropic color shift, but the spatial texture is significantly blurred due to the low-frequency bias of ReLU activations. **Dataset**: `shantung`.

Finally, it would also be possible to parameterize the network directly on the spatial (u,v)-coordinates, without any explicit conversion into an embedding space. This would mean the decoder has to learn the spatial arrangement of texels – while this is possible, it adds complexity to the task of the decoder. BTFs have repeating spatial structures, but that repetition is almost never exact in the pixel grid. Most real-world fabrics for instance will have slight stretching and wrinkles, meaning the material sample cannot be decomposed into exactly repeating tiles, making it hard for the network to make sense of the spatial structure.

We explored the approach in Figure 4.13. While the decoder is able to learn

shadowing effects and complex angular variations in reflectance (anisotropic color shift), it has a hard time learning accurate spatial details, even with a wider network (layers of 256 neurons). The resulting textures are very blurry, even with a bigger network than what we propose for our auto-encoder, which can be explained by the low-frequency bias of fully-connected networks with ReLU activations. Recent work has shown that a different parameterization can yield better results: in NeRF [MST+20], a positional encoding, based on Fourier-features of the spatial coordinates, is used to map the low-dimensional input parameters to a high-dimensional space that the decoder network can use to retain high frequencies. Alternatively, periodic activation functions like sines [SMB+20] have also been proven to exhibit the same properties.

While this could be used to achieve better results here, such an encoder-less architecture would still mean that we have to train on the full BTF texture to encode it. In most experiments however, we trained on a 100×100 pixel portion of the original BTF, and once trained, inferred the full resolution latent maps with the resulting network, which allowed for quicker training. Additionally, the latent maps can be used to drive texture synthesis, whereas with a *uv*-parameterized decoder, the prohibitively large original BTF would need to be used to drive the texture synthesis.

4.5 Conclusion

In this Chapter, we introduced a new 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. While matrix-factorization methods can reconstruct the sparse set of textures accurately, those details are lost during renderings when several textures are blended together. Linear interpolation between sparse reflectance samples blurs details and dulls the contrast while our networks inherently learned interpolation produces much more faithful results.

Our model is especially well suited for storage and transmission of BTFs because

of the small memory footprint of the network, as well as for photo-realistic renderings due to the improved interpolation. Future work could experiment with different error metrics for the network training (SSIM for instance) as well as different color spaces, which could potentially lead to perceptually more accurate results. With regards to compression, our method is also orthogonal to data-agnostic compression strategies such as quantization which could be used to further reduce the storage space of the latent maps and network layers.

In terms of practicality, the main limitation of the approach is that each material requires an individual learning process. A new network has to be trained for each material, which requires several hours of optimization. Additionally, all the learned knowledge is not shared between materials. We address this in Chapter 5, where we explore how to use a single, pre-trained network on all materials, and how the knowledge of the shared encoding space can be exploited.

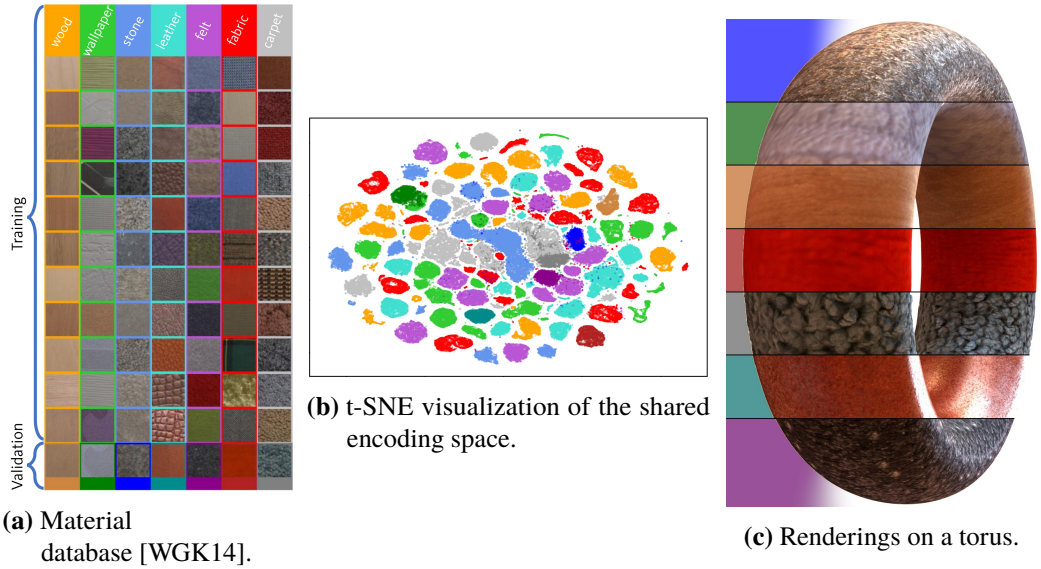
Chapter 5

Unified BTF Encoding to a Shared Parameter Space

In Chapter 4, we presented a material-specific auto-encoder architecture that works very well to compress and interpolate the specific BTF it was trained on, but does not generalize to unseen BTFs. This means that a new instance of the neural network has to be expensively re-trained for the purpose of encoding a different material BTF. The lack of a common parameter space for encoding ABRDFs (texels of a BTF) also means that the method cannot support applications such as appearance synthesis or interpolation in the latent space.

In this Chapter, we address all of these issues by introducing a unified BTF encoding architecture. Rather than repeating the procedure for every new BTF, we train a single, general network, that allows us to efficiently encode unseen materials via simple inference. Latent space operations are made possible because all materials are encoded in the same, shared space, by a single network.

In practice, we train a network to encode/decode BTF texels using 77 materials (Figure 5.1a) from the Bonn BTF database. Each texel's appearance is projected into a 32-dimensional latent basis. Figure 5.1c shows 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.



(a) Material database [WGK14].

(b) t-SNE visualization of the shared encoding space.

(c) Renderings on a torus.

Figure 5.1: Our general network can efficiently encode previously **unseen** materials to a shared latent basis.

5.1 Problem Analysis

Beyond the search for efficient reflectance models outlined in Chapter 2, substantial previous work has gone into searching for 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 crowd-sourced 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.

BTF data has spatial dimensions, meaning it contains information about the spatial layout and organization of ABRDFs, which could be exploited in the compression procedure. However, in order to keep input complexity low, we choose to process

each texel individually, without making use of the neighboring layout 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 inter-reflections. 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 and densely 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 elaborations on the deep learning concepts used in the following sections. Conceptually, our approach is close to an auto-encoder ([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 on 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.

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, notoriously used 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 to an arbitrary number of input photos. Kim et al. [KGT+17] on the other hand use moment-pooling to obtain the same independence. We draw inspiration from these approaches for our network architecture.

5.2 Encoding of ABRDFs with Arbitrary Sampling

In this section, we describe the specificities 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 5.2) projects an input ABRDF of arbitrary ordering and length n to a latent vector of small, fixed dimensionality. 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

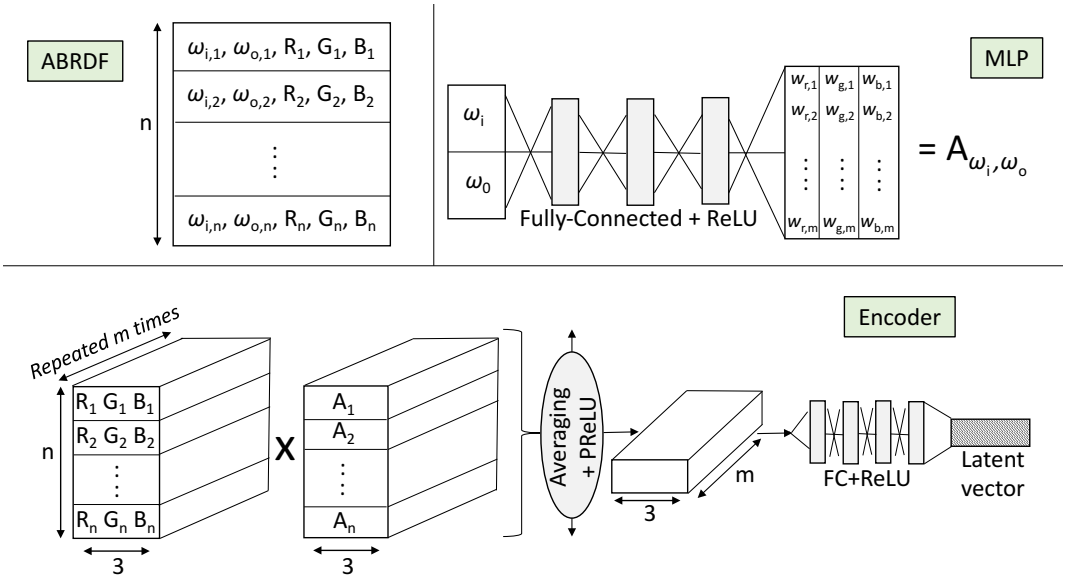


Figure 5.2: Our encoding architecture projects ABRDFs of arbitrary length and ordering to a fixed size latent vector.

angular filters. The remainder of our architecture consists of standard fully-connected networks with non-linearities in between. The decoding structure (Figure 5.3) 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) on the BTF texels.

5.2.1 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 of the product of the reflectance lobes with learned filters in angular space.

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 is constant, the recovered response would

be the mean reflectance, which is a good approximation of albedo.

Another possibly more intuitive way of looking at our encoding approach is as an approximation of a linear layer. One might want to straightforwardly apply a fully-connected layer to the list of RGB measurements. However, this is not possible because the input's ordering and length might change. 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 fully-connected layer, in the angular domain.

MLP. The MLP takes the angles (in stereographic parametrization, similarly to the decoder in Chapter 4) 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 $\mathbf{A}$. On the other side, the list of RGB reflectance values is expanded m times. Multiplying the resulting weight matrix element-wise 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 5.2. While Max-Pooling also satisfies the invariance 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.

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

Comparison of our Encoder with Chapter 4. The previously presented encoder [RJGW19] performs 1-dimensional convolutions and down-sampling (max pooling) over the ordered list of reflectance values, followed by fully-connected layers. 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 down-sampling is that the input ABRDFs are extremely large (almost 80,000 values), so straightforwardly using fully-connected layers is impracticable. Furthermore, a fully-connected layer would require fixing the angular sampling of the ABRDFs, which we want to remain flexible. Instead, 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 5.3) is also a fully-connected network with non-linear activations, following the same decoder design as in Section 4.2.2. 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.

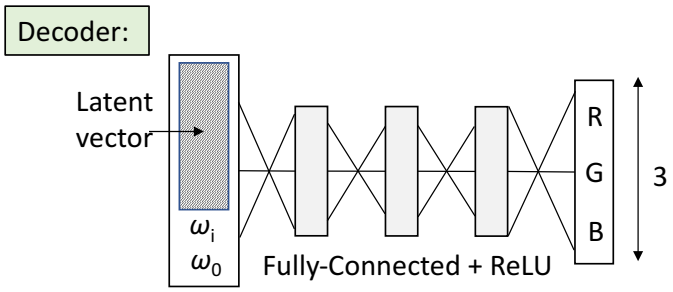


Figure 5.3: Our decoder architecture is identical to [RJGW19].

5.2.2 Implementation and Training Details

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 compensate the high dynamic range of measurements, a logarithmic transformation is applied to the values before the normalization.

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. 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 element-wise with each RGB reflectance measurement (expanded 800 times). The encoder only consists of a Parametric ReLU (*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 L_1 loss gives our encoding a more accurate average hue and preserves more contrast than the L_2 loss used in Chapter 4.

5.3 Performance Comparison with Over-fitting Network

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 Chapter 4, which we consequently refer to as *custom network*, as a new instance of the network is trained 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.

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 5.4 displays the texture reconstructions of each of the validation materials at one particular combination of light/view directions.

5.3.1 Comparisons in Texture Space

In Figure 5.4, we compare the ground truth with the custom network of Chapter 4 trained on all Bonn training materials, except the validation materials displayed in the figure, to the custom network over-fitted 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 number of latent dimensions to project and the same decoding budget, allowing us to assess how well each architecture is able to learn a more general embedding. As the custom network in

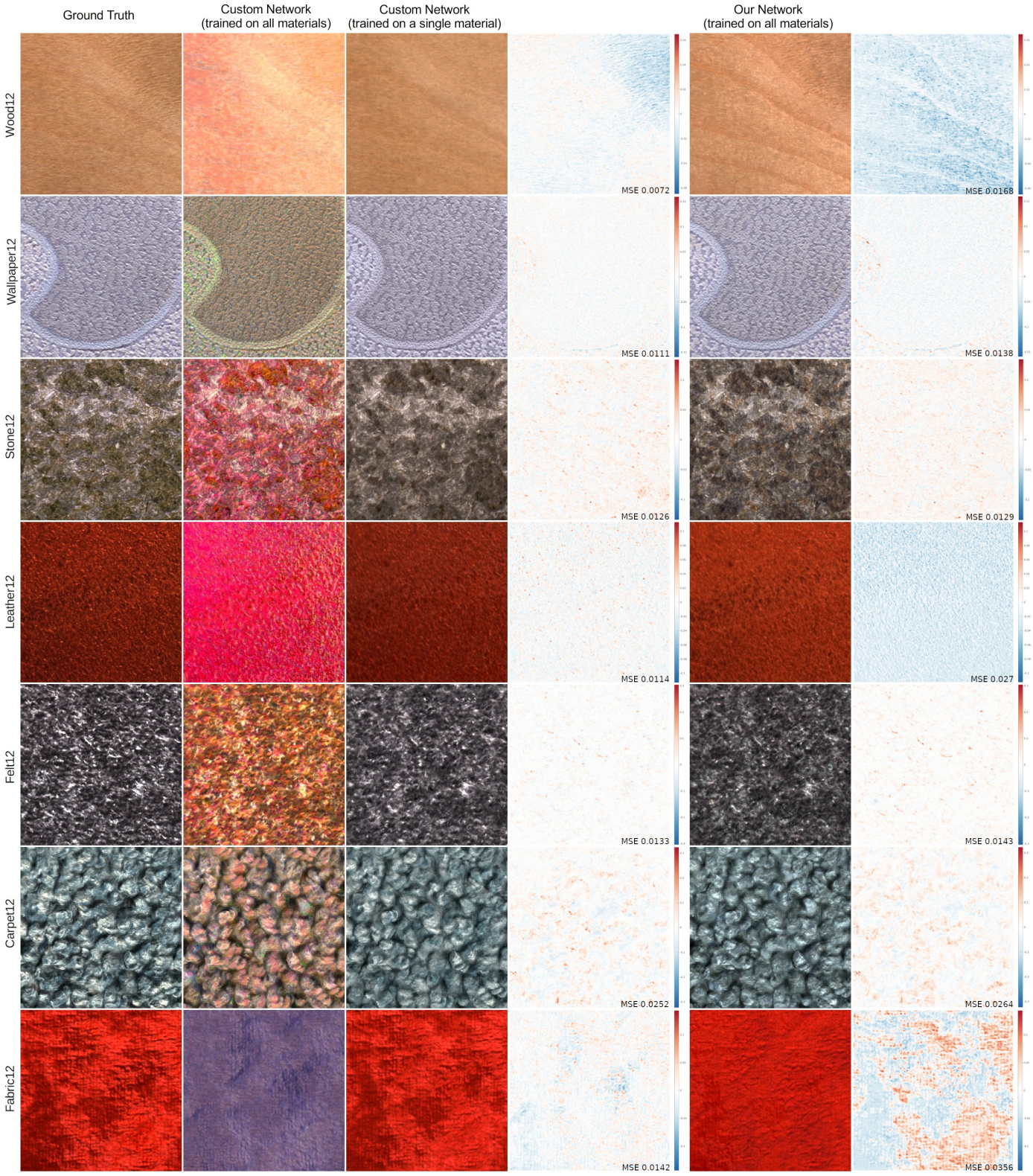


Figure 5.4: Comparison of the reconstruction using different architectures (at equal latent space and decoder size) with the original texture from BTF datasets kept for validation. Except for the custom network [RJGW19] over-fitted specifically to the respective dataset, the other networks have not seen the data at training time. View/light azimuth and elevation angles of the shown textures: 0°, 45°, 90°, 30°.

column 3 is over-fitted to the specific material, it represents the upper performance bound that a general architecture could reach.

On average, the custom network over-fitted 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 over-fitted 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, inter-shadowing, 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.

The custom network sets a standard objective for a reasonable reconstruction performance. When over-fitting 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 5.1 shows the average reconstruction error on the unseen datasets for our

network at varying latent sizes.

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

We compare our performance with that of the custom network. It is to be noted that we train our network on an L_1 loss on the log-transformed and whitened texels, while the custom network is trained with an L_2 loss on the texels preprocessed in the same way. From Table 5.1, we elect 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 over-fitting 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 heavier than those of Chapter 4, but the decoder network is shared between all materials, so doesn't require extra storage per material.

Regarding the size of the decoder, we showed in Chapter 4 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.

5.3.2 Comparisons on Renderings

When rendering with the original dataset, the ground truth data is inevitably corrupted by linear interpolation between nearest sampled directions (usually 9 light/view

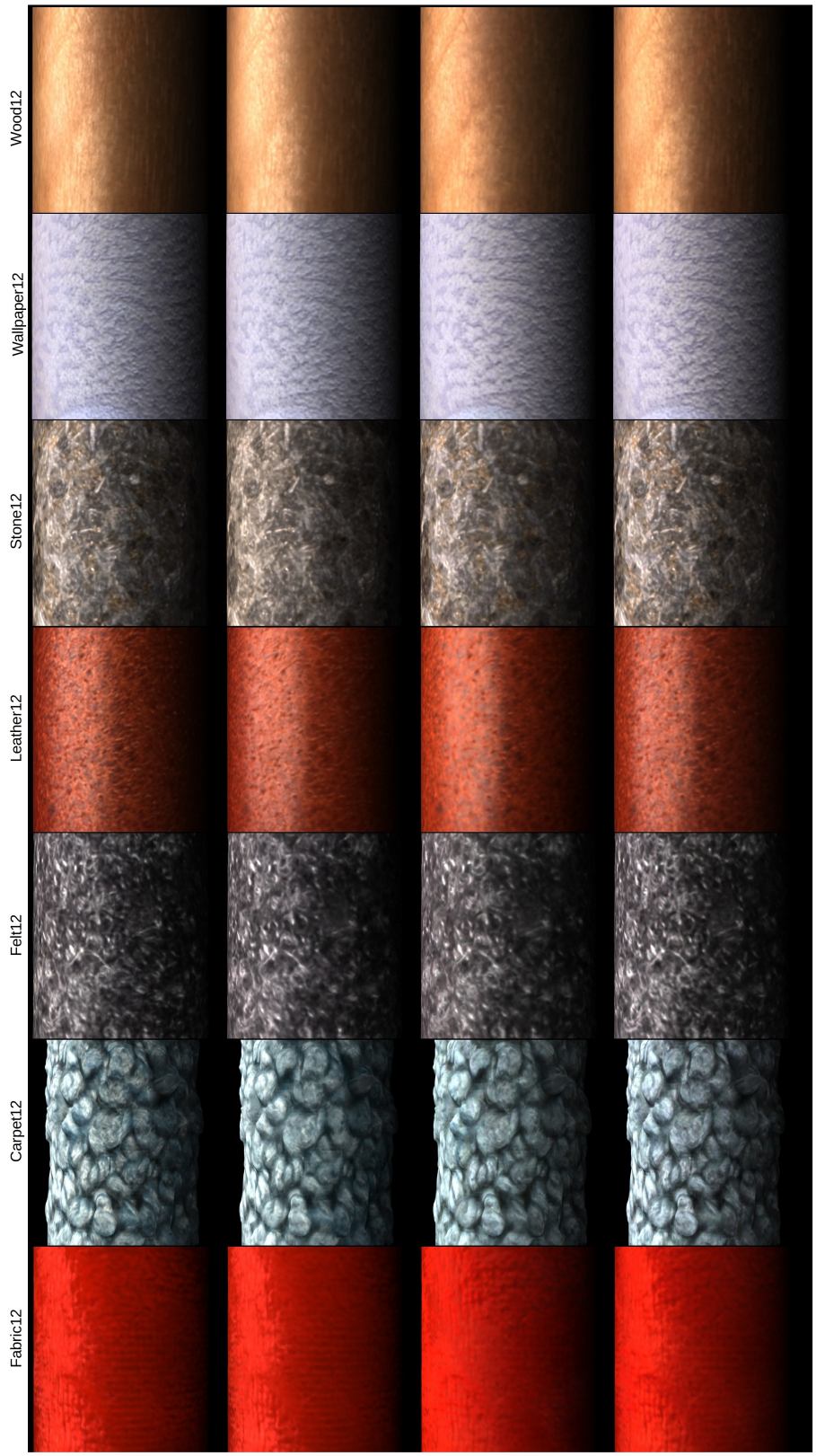


Figure 5.5: Renderings, *from left to right:* full BTF, custom network (over-fitted to the material), our general network (unseen material) with latent sizes 8 and 32. Textures mapped on a cylinder.

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 5.4) the reconstruction is not perfect. The stability of the neural interpolation was demonstrated in Section 4.3: 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 5.5, we compare renderings with the custom over-fitted network and with two instances of our architecture, to renderings with the original BTF. For reference, renderings of the BTF-mapped cylinders in Mitsuba, at 800×800 pixels, at 32 samples per pixel, parallelized on 10 processes, path-traced 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.

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 encoding 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 over-fitted network. Non-local effects like subsurface scattering and inter-shadowing 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 over-fitted 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, inter-shadowing, etc.). For visualization of temporal coherency, we provide animation frames of a BTF-textured plane under a moving point light source in the Appendix (C), comparing rendering with the original BTF to the custom network and our general network.

5.3.3 Evaluation on a Different Data Source

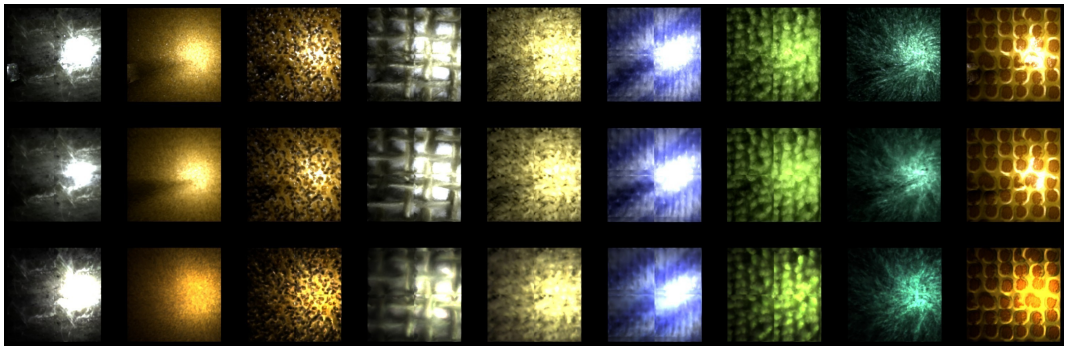


Figure 5.6: Point-lit renderings of BTF datasets from [FKH+18]. *From left to right:* materials 1 to 10. *Top to bottom:* Original BTF, [RJGW19] (over-fitted), our reconstruction (unseen material).

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 contain 6561 angular measurements VS 22801 for Bonn. The angular sampling of the UTIA datasets is hence less dense, but also distributed uniformly. However the materials are generally more specular and some of them contain transparent layers. As far as the authors know there is no height field parallax correction. Figure 5.6 show point-light renderings of the first ten materials of the UTIA MAM database, rendered using the original BTF, our custom network [RJGW19] over-fitted to the respective material, and our unified network's predicted latent maps. The most notable difference is that our network struggles to reconstruct the truncated specular highlights, because it has not seen any examples of those in the training data. The over-fitted network can learn those. Overall, we still observe the expected degradation compared to the over-fitted network, but beyond that our network generalizes plausibly to a new, unseen, data source.

5.4 Shared Latent Space

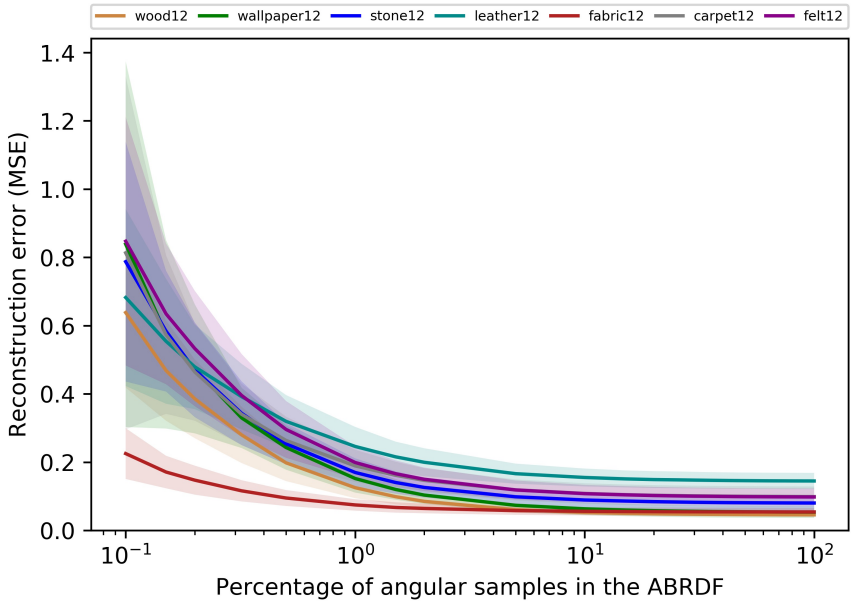


Figure 5.7: Mean and standard deviation of the reconstruction loss as a function of percentage of angular samples, computed over a random selection of pre-processed ABRDFs for each validation material.

5.4.1 Robustness to the Original 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 datasets of the UTIA MAM2014 database ([FKH+18]) contain 6561 angular measurements, which amounts 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 5.7. From less than 10% of angular samples onward (2280 randomly chosen out of the original 22801), the reconstruction is stable and has converged.

Figure 5.8 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

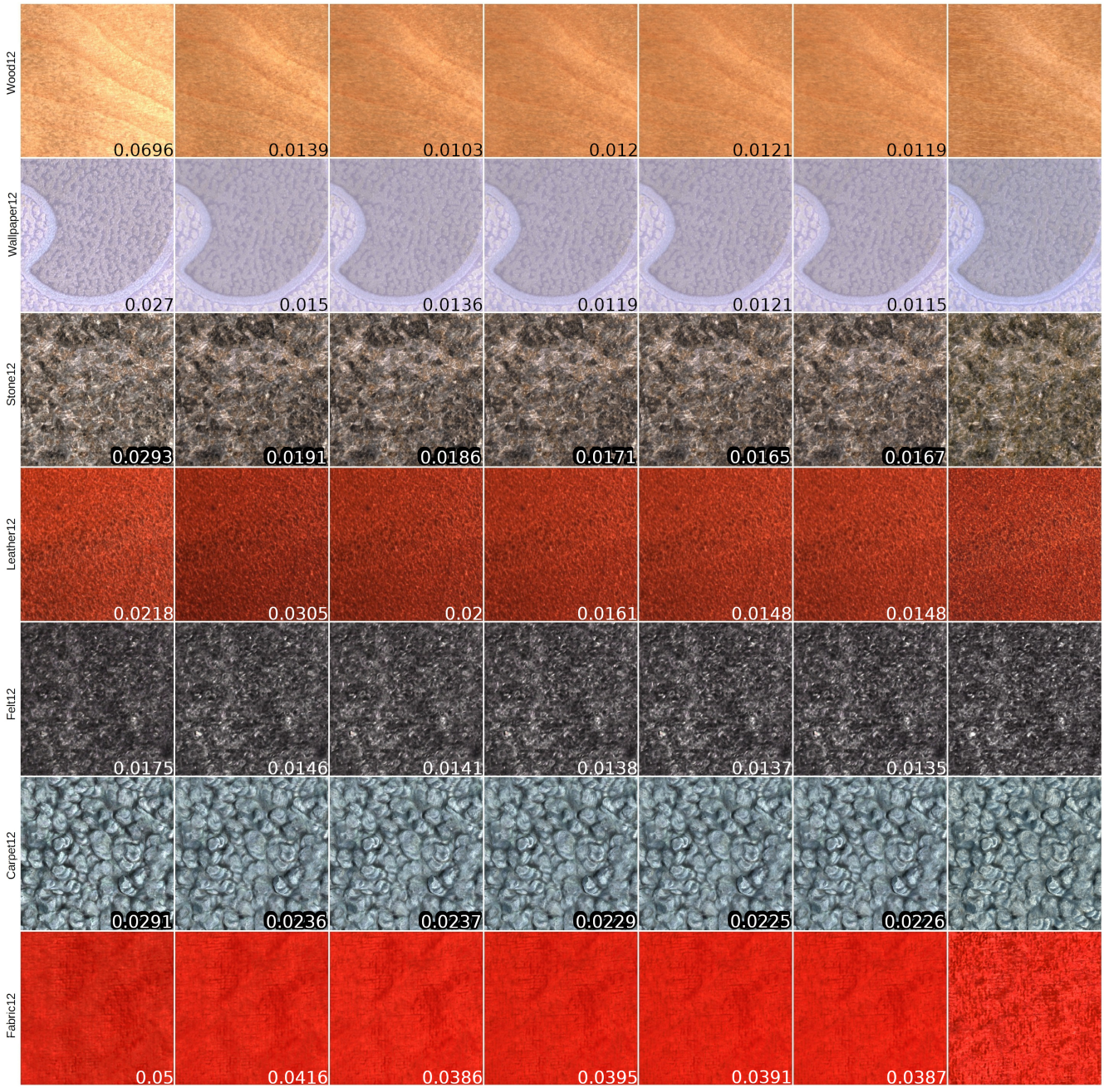


Figure 5.8: *From left to right:* Reconstruction using our network (latent size 32) at 1, 5, 10, 20, 50 and 100% of angular samples, with average reconstruction error. *Rightmost column:* Ground truth. **Angles** (view azimuth, elevation, light azimuth, elevation): 0°, 45°, 0°, 90°.

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 by a tenfold. Even datasets with a low-resolution angular sampling will benefit from this encoding that creates an appearance 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.

5.4.2 Filtering in Latent Space

An important technique to facilitate rendering at different scales is spatial down-filtering 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 effects of down-sampling in latent space before rendering, versus down-sampling in reconstructed texture space, in Figure 5.9. The difference is barely noticeable, filtering in latent space proves to be very stable. Furthermore, this saves computation time as it allows pre-computation of mipmaps and avoids having to run the decoder multiple times.

5.4.3 Texture Synthesis Using the 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 5.10 to enlarge the BTF by a factor of 5.

5.4.4 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 5.1b). We observe that within a BTF, the projections of texels are very well clustered. There is however not a lot of consistency with the semantic classes defined in the Bonn database. Nevertheless, this is to be

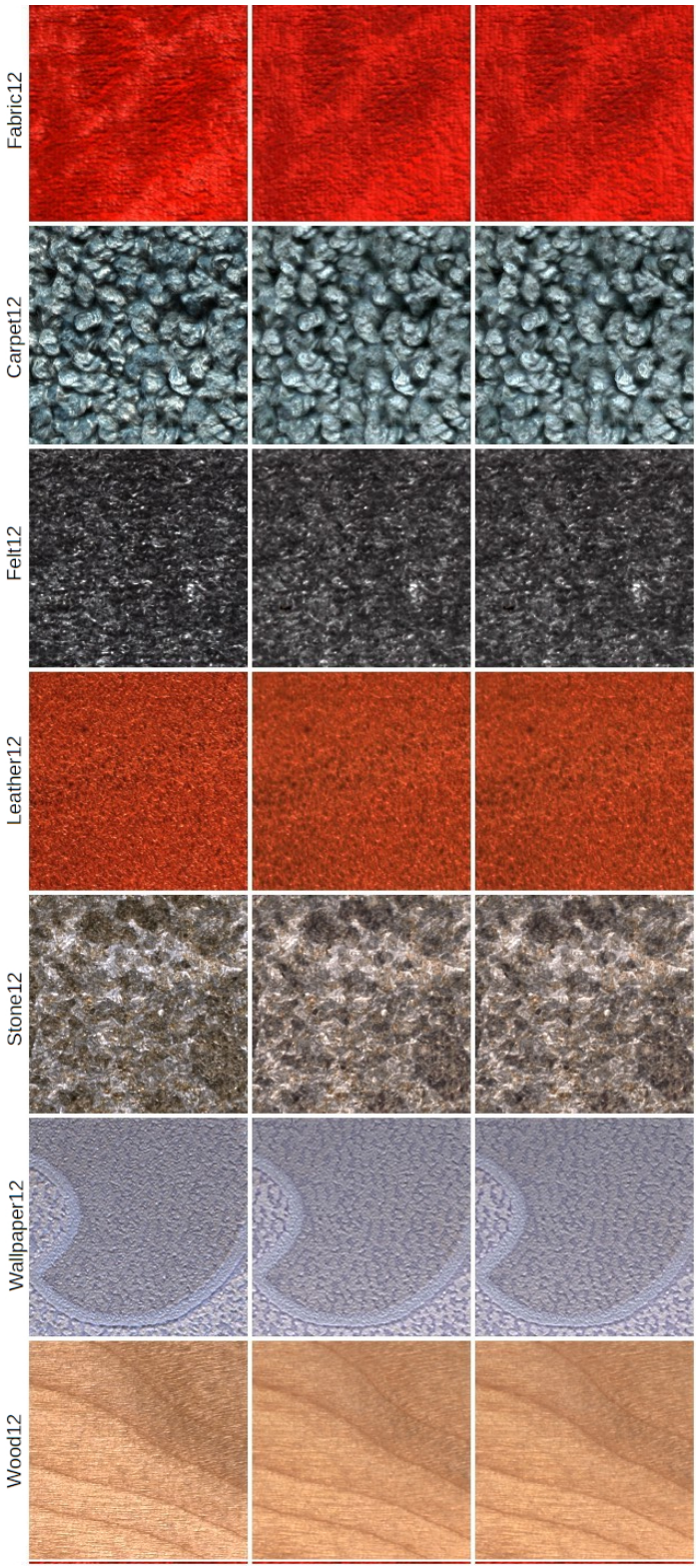


Figure 5.9: *Rows (top to bottom):* Ground truth, reconstruction filtered in image space, and reconstruction filtered in latent space. **Angles** (view azimuth, elevation, light azimuth, elevation): 0°, 90°, 270°, 30°.

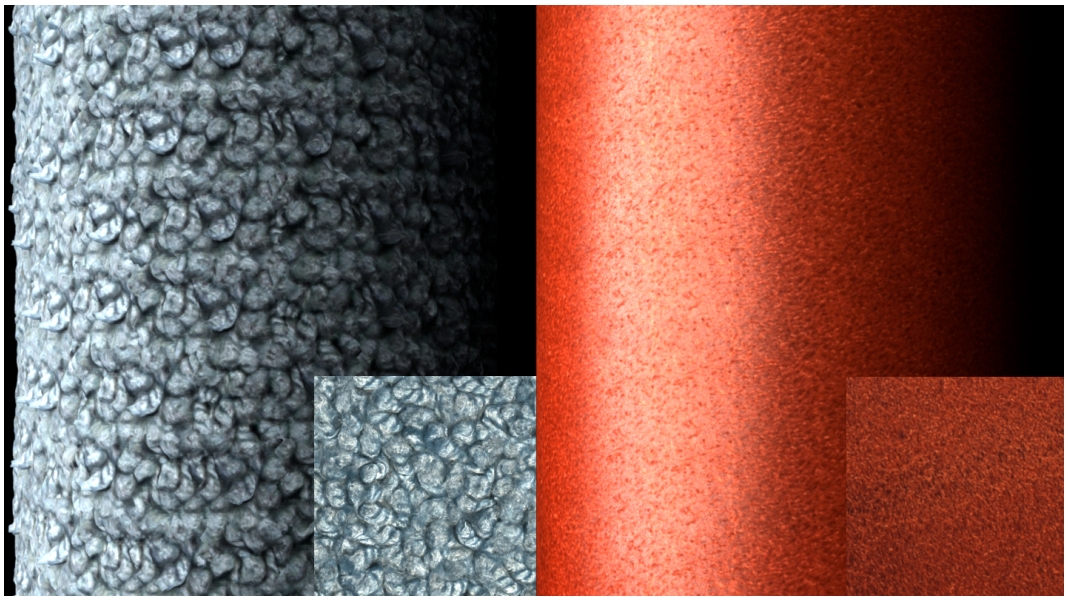


Figure 5.10: Renderings with textures synthesized in latent space. *Bottom corners:* Ground truth BTF texture.

expected as our clustering is relative to appearance, while the semantic classes refer rather to fabric/material types. For instance, some of the carpet ABRDFs are very close in albedo and reflectance shape to those from felt BTFs, even though they are in different semantic classes.

5.5 Discussion

Training Dataset Diversity. The training dataset consists of 77 BTFs, which on first sight is a low number for deep-learning standards, so it might seem surprising that the unified network can generalize to unseen materials. However, the training inputs to the network are not BTFs, but individual ABRDFs (BTF texels). Since the BTF resolution is 400×400 texels, this amounts to 160,000 data points per BTF. Of course, we need to take into consideration that most materials captured as BTFs have an inherently repeating structure (most fabrics do, for example), so there might be redundancy among the texels. However, since this structure does not perfectly tile, two texels that map to perfectly similar material points in the real-world will still be observed with a slightly rotated normal and tangent frame, and with sub-pixel shifted sample positions, meaning every texel adds new information to the learning and

provides an unseen data point. Finally, we also cheaply augment data by permutation of color channels during training, which allows to gain a 6x increase in variety. Last but not least, our decoder network remains small and is hence less likely to overfit to the training examples.

Validation Dataset. For a more complete picture of the generalization capabilities of the unified network, we show the visually closest matches in the training dataset to the materials used for validation in Figure 5.11. While there is some overlap (each validation material has parts that are similar to those in the training database), the validation set is still significantly different and unseen for the trained network. For instance, `carpet12` has no match in terms of hue, `fabric12` has a close match for hue but not for specularity (or in that case, with a very different green tint), and `leather12` has a very detailed, small-scale structure of ridges and specular bumps that cannot be found in the other leather datasets. Hence, if the unified network performs to an acceptable standard on the validation datasets, it means that it should generalize reasonably well.

Interpolation between BTFs. We demonstrated interpolation in the latent space for mipmapping purpose, among texels of the same material. But can we interpolate between materials?

The straightforward way to interpolate between textures would be to linearly blend the texels – here, this would mean linear interpolation between the latent texel parameters, as applied in Section 5.4.2. However, the graceful linear interpolation we observed for the mipmapping case does not necessarily translate to texels from disparate BTFs, as, unlike texels from within the same BTF, they are much further apart in latent space. The latent embedding is non-linear, so using a linear approximation for the interpolation will become more and more hazardous as the embedded positions are far apart.

Additionally, there is no guarantee that the resulting material will depict a plausible surface (see Figure 5.12), because BTF texels do not exist in isolation. They encode non-local effects that depend on a texel's neighborhood and are only consistent with the specific BTF's material. Per-point blending of texels alone does not guarantee

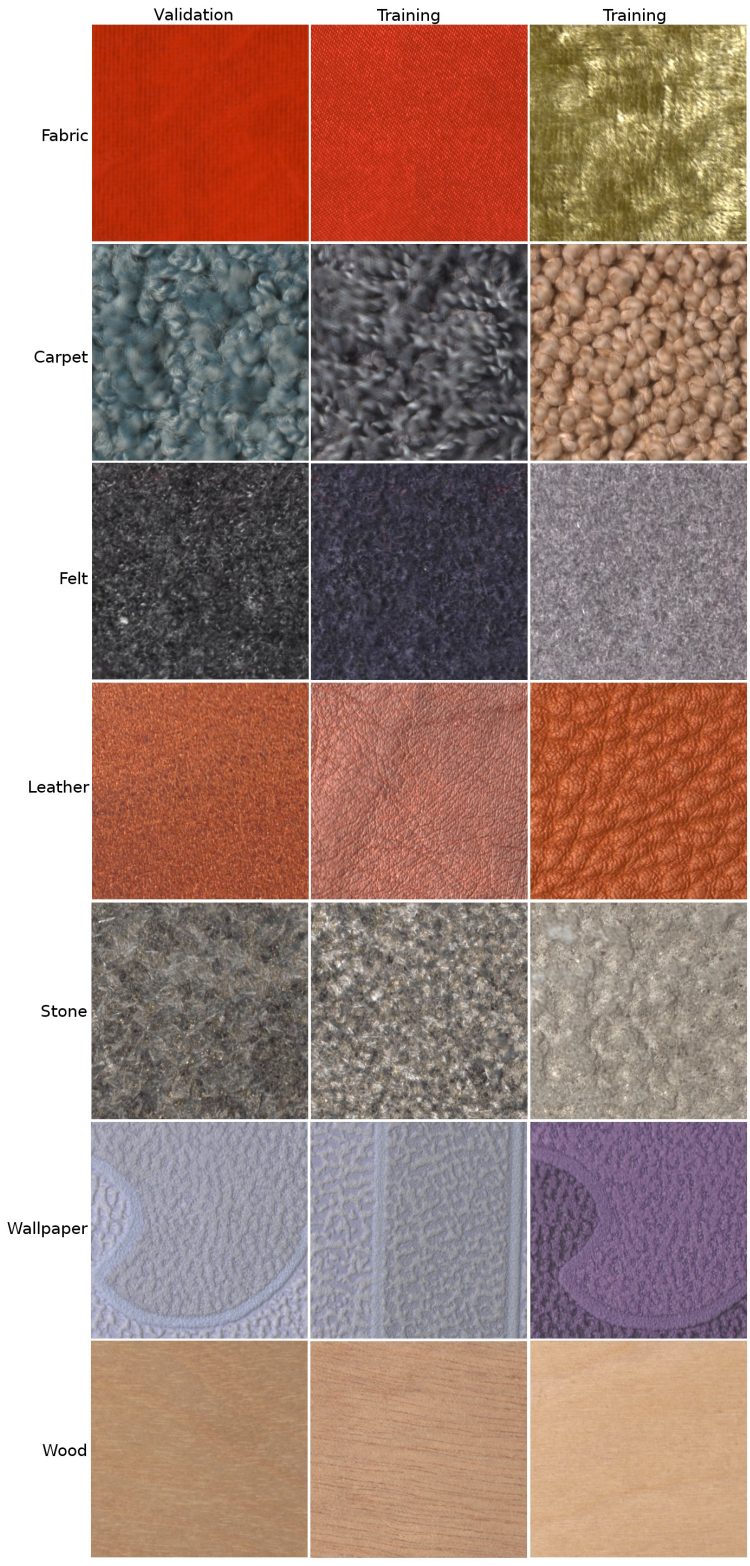


Figure 5.11: Validation BTFs and their two closest matches in the training BTFs (under frontal view and lighting).

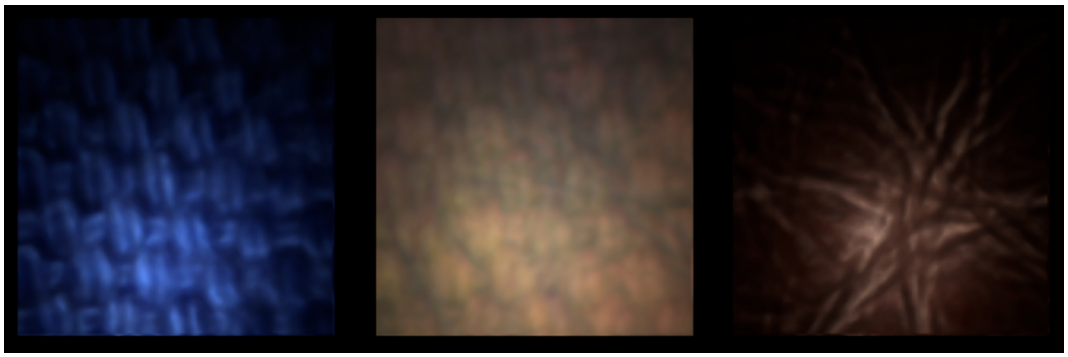


Figure 5.12: Failure example of linear interpolation between BTFs. **Datasets** (left to right): `fabric04`, `interpolated btf`, `leather04`.

that the resulting appearance is spatially coherent across these neighborhoods, nor that the resulting material corresponds to any meaningful mix of the two original BTFs.

More generally speaking, any approach to interpolate between BTFs will also have to solve the problems of interpolation between 2D textures. If linear interpolation is chosen, the result will exhibit the same artifacts as linear blending between 2D textures, that is blurring, ghosting, and general lack of plausibility since it will look like the two original materials superimposed. A more appropriate texture interpolation would need to take into account the spatial neighborhoods: patch-based texture synthesis algorithms [RSK13] achieve this by extracting small patches from the original textures, and preserving meaningful patch statistics in the synthesis step. More recently, CNNs have been applied very successfully to the texture synthesis [GEB15] and interpolation task [YBS+19], as they manage to preserve the neighborhood properties. In contrast, our neural encoding does not use convolutions and processes each texel separately. In order to take the spatial arrangements into account, a CNN-based texture synthesis and interpolation strategy could be employed on the latent maps, both for synthesis and interpolation of BTFs.

5.6 Conclusion

We presented a network architecture capable of handling and encoding unstructured angular reflectance measurements. Inspired by auto-encoders, our network projects ABRDFs into a low-dimensional latent space, analogous to analytic SVBRDF model

parameters, while the decoder network is analogous to the analytic SVBRDF 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 the custom network which requires a new network instance for every new material, encoding a new material with our method requires a simple evaluation of the encoder. Having a single auto-encoder instance also means that the latent space is shared between materials, i.e., texels are converted into the same domain. This allows exploration of the parameter space for applications such as latent mipmapping, texture synthesis etc.

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 necessarily have close neighbors in the training set (see 5.5). 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 the employed analytic model, which is opposed to our goal of staying flexible to learn all the components of real-world reflectance functions.

Chapter 6

Conclusion and Outlook

The aim of this thesis was to investigate ways to faithfully and efficiently render real-world materials with complex appearance, with a special focus on fabrics. The literature is filled with efficient techniques and clever approaches, but there is always a compromise and trade-off between practicability of capture and usage and accuracy of appearance reproduction, as well as often times restrictions to specific types of materials or appearances. In order to remain flexible and generalizable, we chose bidirectional texture functions as a starting point, and worked on improving their main drawback – practicability. Using BTFs in an online context or as an asset in a game was impossible before; with our neural model this becomes feasible.

Although public BTF databases exist, the number of available datasets is quite low, and there is no access to the raw data. In order to have full control over the data and the process, a custom acquisition hardware was designed as part of an industrial collaboration between UCL and Change of Paradigm Ltd., and I implemented the full calibration and data processing pipeline. The system is presented in Chapter 3 – boasting properties such as a high spatial resolution as well as arbitrary angular configurations and resolution. However, the first datasets captured so far have comparatively coarse angular resolution to compensate for the low intensity of the light source and reduce capture times.

The core of this thesis is centered on the development of a new representation for BTF data. The model should be accurate enough to preserve all the details contained in BTFs that make their use attractive (spatial details and non-local lighting effects,

such as subsurface scattering). At the same time, the model should combine this with the practicality and efficiency of analytical models. We found that a neural BTF model satisfies all of these conditions.

Our model uses a set of latent parameters that describes the appearance of a BTF texel. The latent vector can be appended to light and view directions, and used as input to the respective decoder network, which will output a reflectance value. In this regard, our neural model is very similar to an analytic BRDF model, which also takes a set of appearance parameters associated with view and light directions as input, and returns a reflectance value.

Chapter 4 introduces the neural BTF model and demonstrates its superior interpolation and representation capabilities. Similarly to analytic models, however, the material representation requires an optimization process to represent a new material – in our case, a new network needs to be trained to obtain the respective latent maps and the decoder. An additional difficulty in designing a method to convert several BTFs into the same representation, comes from the difference in format between different data sources. Data coming from different devices will have different angular or spatial resolution, as well as different angular configurations.

Chapter 5 shows that, at the cost of a slight loss in accuracy compared to the customized approach, new unseen materials can be encoded via inference of the encoder network rather than a full optimization. Our encoding method contains no prerequisites on angular and spatial resolution and ordering, and we demonstrate results on BTF data from various sources. The latent space is then shared between BTFs, opening this domain to further applications. Additionally, there is a single, shared decoder network now, allowing further optimizations of the rendering performance, for instance.

6.1 Future Work.

Currently, the main limitation for research on BTFs is the difficulty of dataset acquisition (due to complexity of the devices and the time and storage overheads involved) and the resulting lack of data, especially from the perspective of deep-

learning applications. This still hinders a systematic high-resolution (angular and spatial) large-scale acquisition of materials. The availability of a much larger set of BTFs would improve the quality of our unified model: the more data our general network is trained on, the better the inference. This means the network will not only reconstruct unseen materials more accurately, but the latent space will also behave better as it will be probed in a much denser way during training.

Conceptually, the main difference between our neural model and previous analytic models is that the parameter space and the appearance function are *learned*, rather than chosen by the author or the user. While this has the advantage of increased flexibility, it comes with the drawback that the latent space and the decoder cannot be intuitively understood. Analytic BRDF model parameters are always linked to quantities that have a visual meaning, such as glossiness or albedo, making the material representations easy and intuitive to edit. This is the main use case that our model currently does not support, but it would be very desirable for artistic control, in the film and game industries, for instance.

This limitation touches on a big drawback of neural network applications – the lack of analytical understanding of the outputs. The achieved scores (or errors) are very impressive, but often sacrifice the idea of gaining any information on the problem, which more traditional approaches are often based on. However, the emergence of deep learning as a widespread tool is still recent and promises many more advances and realizations. In the case of our unified BTF encoding space, the straight-forward follow-up is to compensate this lack of understanding by additional learning, to map back from latent parameters to quantities that we understand visually. Given an existing latent space and a set of meaningful appearance parameters [SGM+16, GGPL20], can we learn a mapping that will allow intuitive latent space editing of neural BTFs?

Appendix A

Glossary

ABRDF	**A**pparent **B**idirectional **R**eflectance **D**istribution **F**unction
BRDF	**B**idirectional **R**eflectance **D**istribution **F**unction
BSDF	**B**idirectional **S**cattering **D**istribution **F**unction
BSSRDF	**B**idirectional **S**cattering **S**urface **R**eflectance **D**istribution **F**unction
BTF	**B**idirectional **T**exture **F**unction
CNN	**C**onvolutional **N**eural **N**etwork
MLP	**M**ulti **L**ayer **P**erceptron
NN	**N**eural **N**etwork
PTM	**P**olynomial **T**exture **M**ap
PCA	**P**rincipal **C**omponent **A**nalysis
ReLU	**R**ectified **L**inear **U**nit
SVBRDF	**S**patially **V**arying **B**idirectional **R**eflectance **D**istribution **F**unction

Appendix B

Datasets Captured with our Setup

In this Chapter, we show datasets captured with our setup [RRW19], in three formats: the original BTF, the per-material representation [RJGW19] and the unified model [RGJW20]. We show all the datasets (6 in total) we were able to capture successfully and completely. For quicker captures and due to restricted access to the capture system, we captured four datasets at once. At full resolution, we can, without loss of quality, extract a $2{,}000 \times 2{,}000$ pixel BTF texture of the 10×10 cm frame contents given the native camera resolution. When capturing four datasets at once, we extract a 4×4 cm region of each sample, at a 800×800 texels resolution.

Of the 8 fully captured materials, 2 BTFs were not successfully extracted as the bracketed shutter settings and corresponding ISO were not adequate for those materials, resulting in HDR reconstructions with too much noise. We hence only show 6 BTFs in this Appendix. We respectively captured the materials (`greenwool`, `bluecord`, `purpleweave`, `whitemesh`) and (`shantung`, `cotton`) together. The specific light-view direction combinations are not identical in both captures although roughly at similar angular resolution, but both the per-material and the unified approach generalize to any angular sampling.

The custom networks achieve very good fidelity in the renderings of all materials. For the unified network, the challenge is especially difficult since these BTFs have a low angular sampling compared to the training set, combined with materials that exhibit complex appearance (such as drastic anisotropy for `shantung` or high specularity for `purpleweave`), that the network has not been extensively exposed to in the training set.

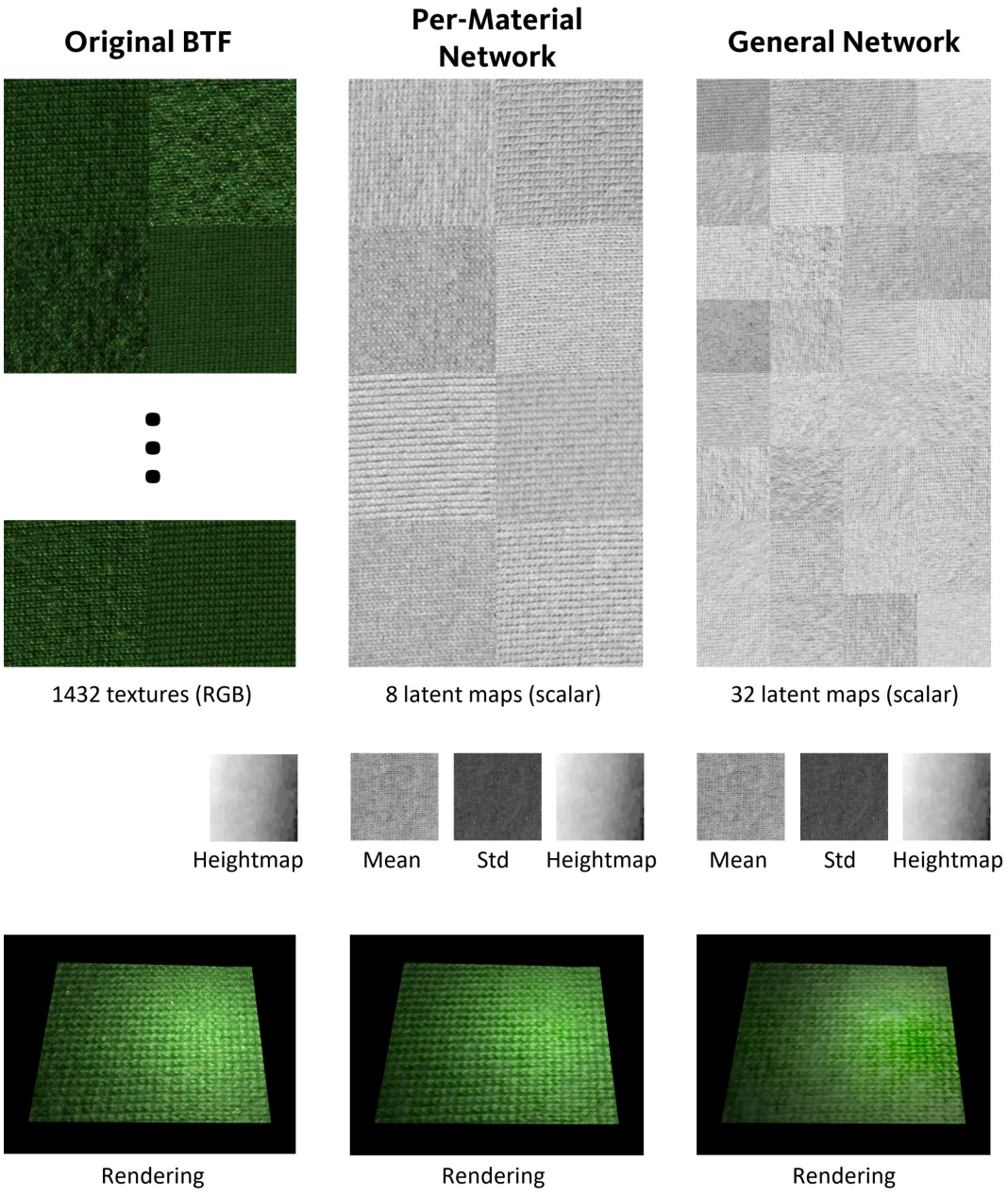


Figure B.1: Green wool sample (`greenwool`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

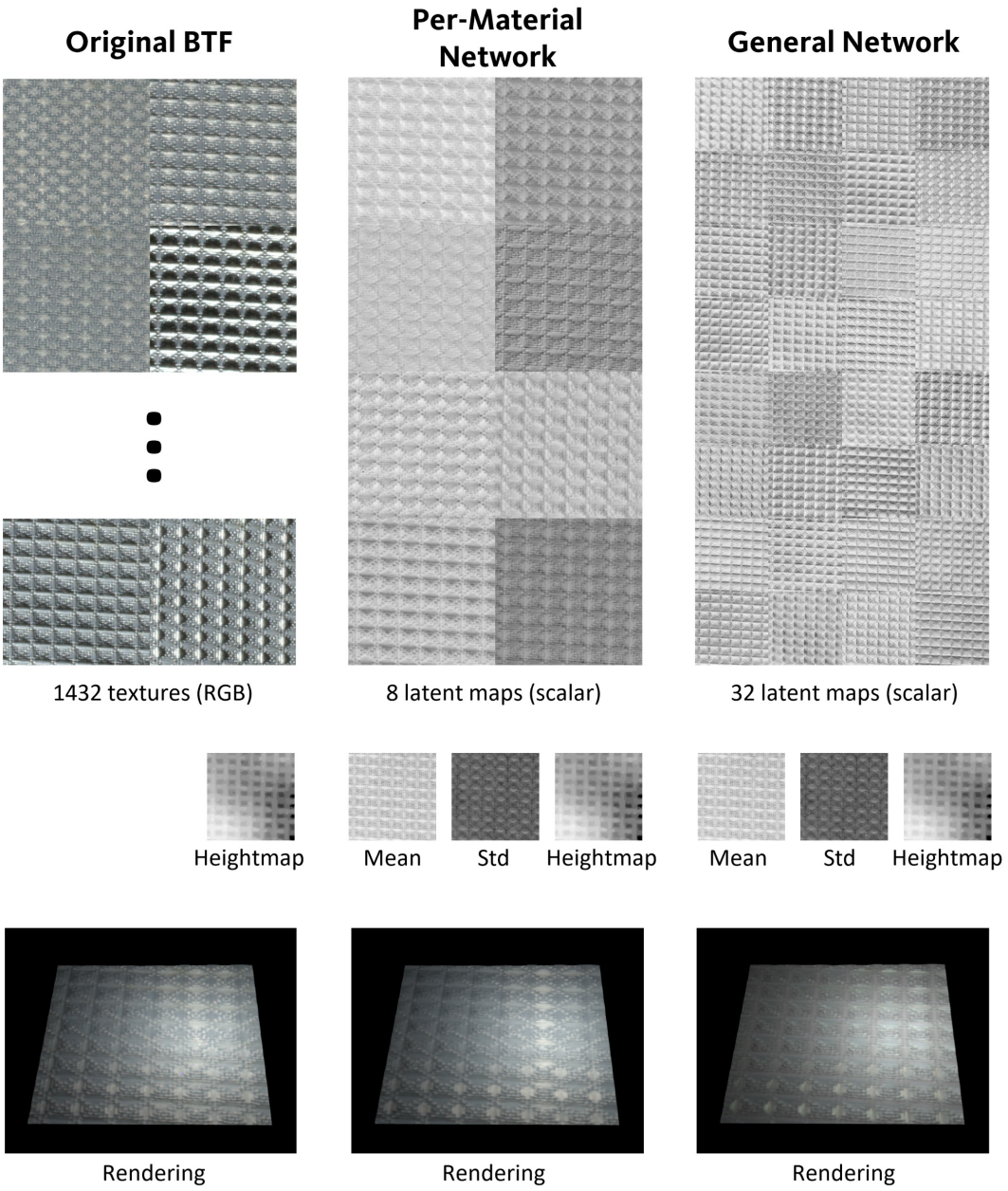


Figure B.2: White mesh sample (`whitemesh`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

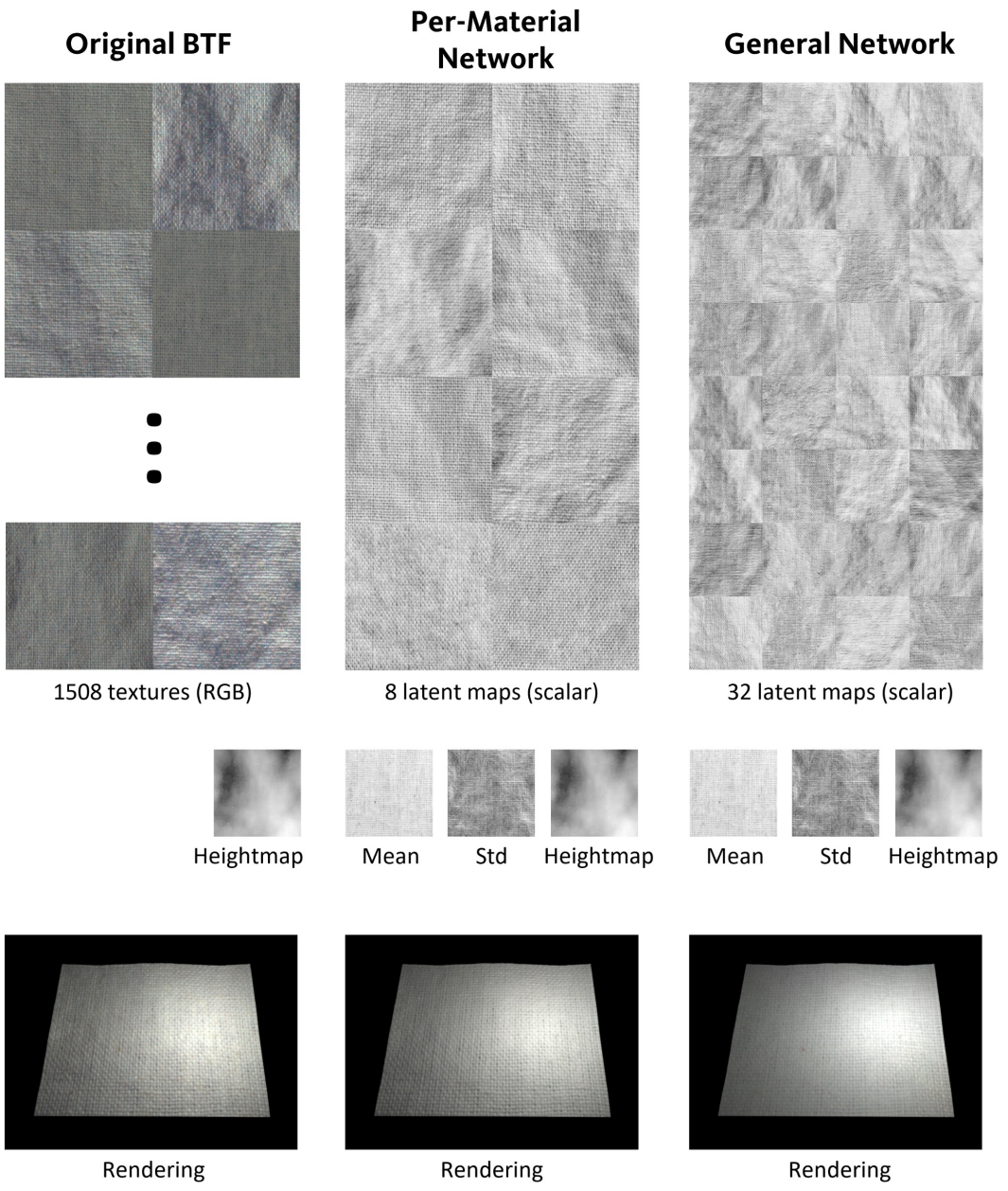


Figure B.3: Grey cotton weave sample (`cotton`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

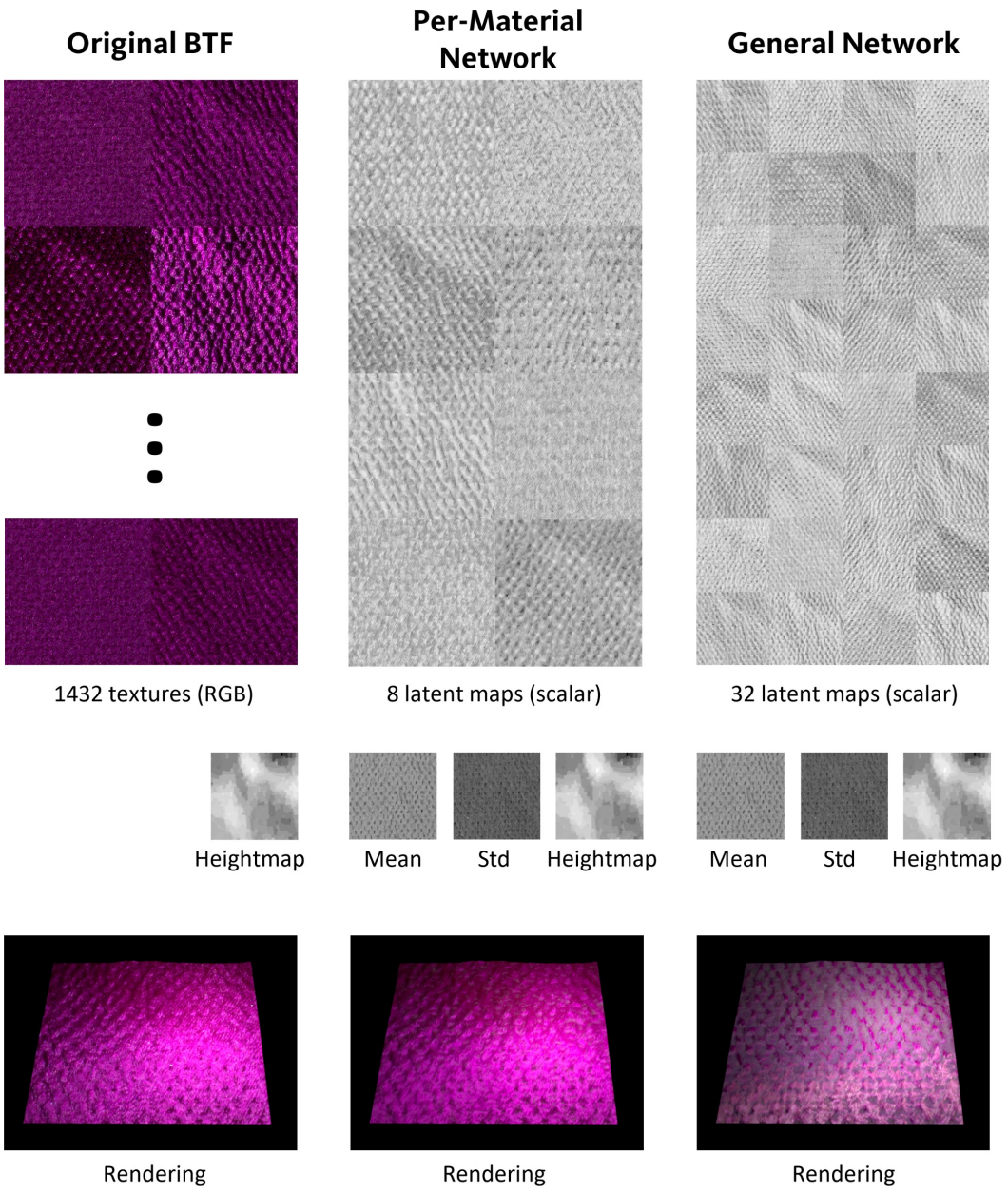


Figure B.4: Shiny purple weave sample (`purpleweave`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

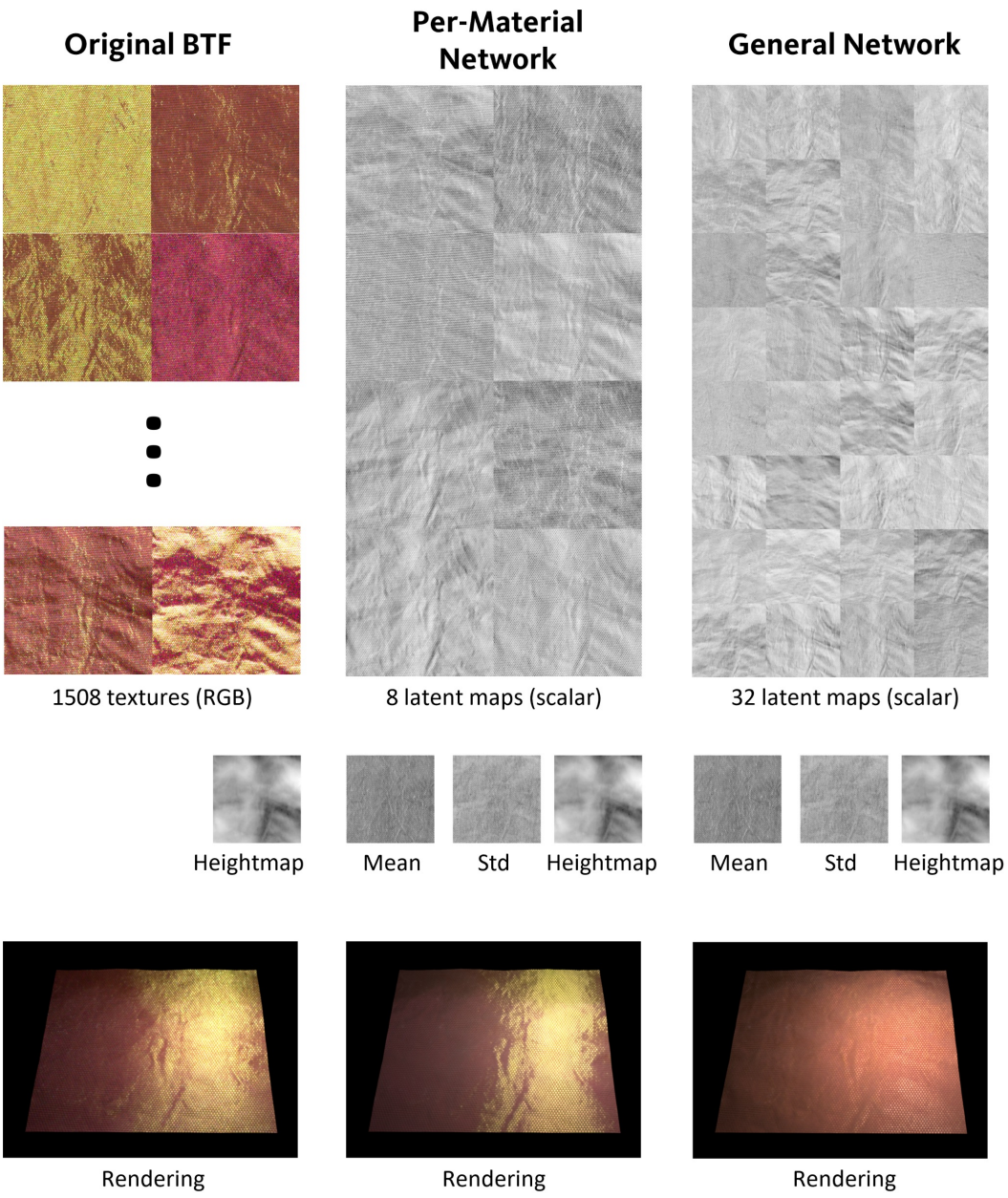


Figure B.5: Dual-colored, anisotropic shantung silk sample (`shantung`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

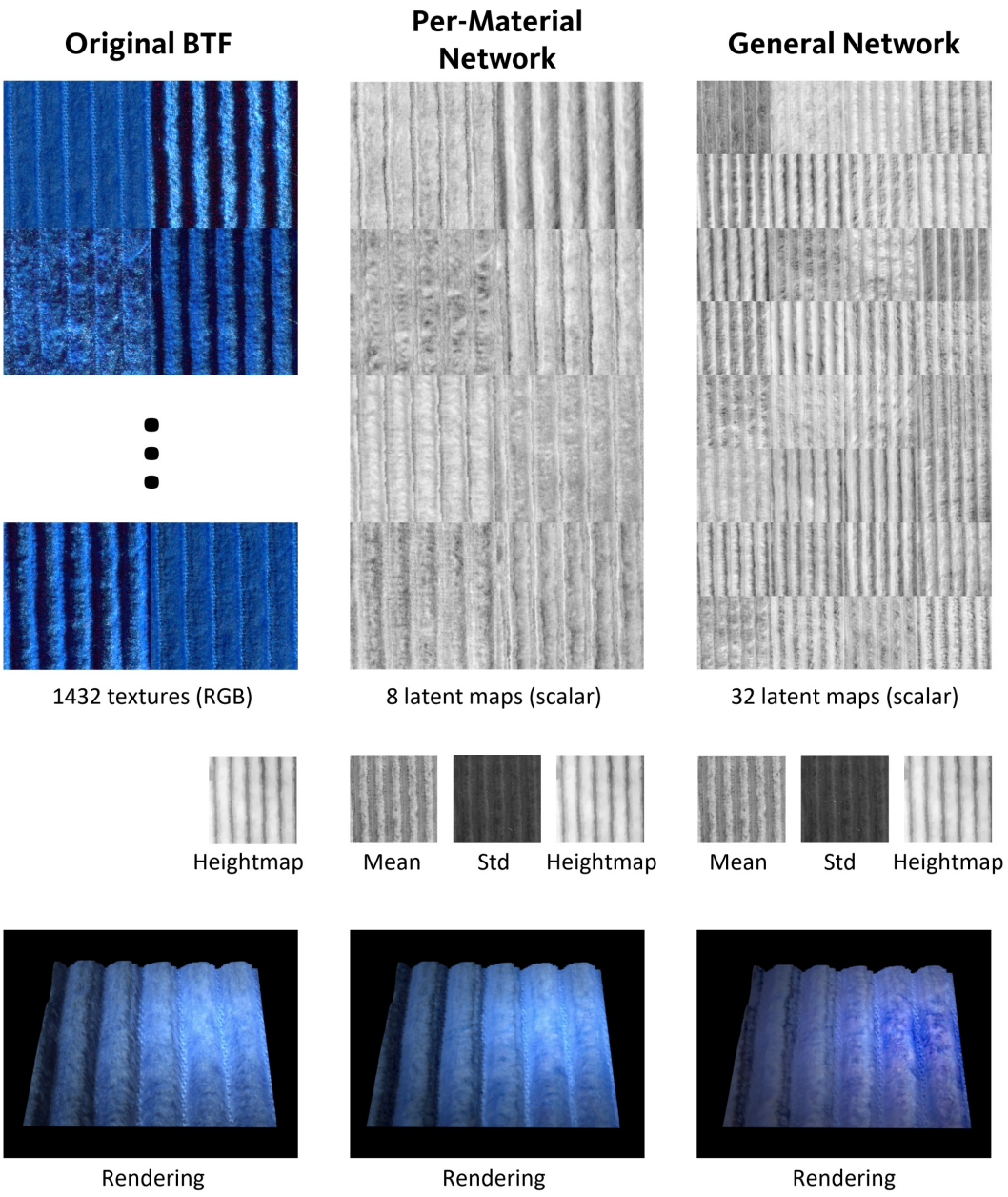


Figure B.6: Blue corduroy sample (`bluecord`): Textures are extracted at 800×800 pixels. All scalar maps are gamma-corrected for display.

Appendix C

Animated Rendering Frames for Validation Materials

In this section, we display a collection of still frames from rendered animations of a BTF textured plane under moving point light illumination. The plane rotates as well, which is important to observe the fidelity to anisotropic highlights. We compare three different rendering techniques: the original BTF, the overfitted custom network of Chapter 4, and the general network presented in Chapter 5. Results are shown for all seven materials from the Bonn UBO2014 database that were used for validation (unseen by the general network).

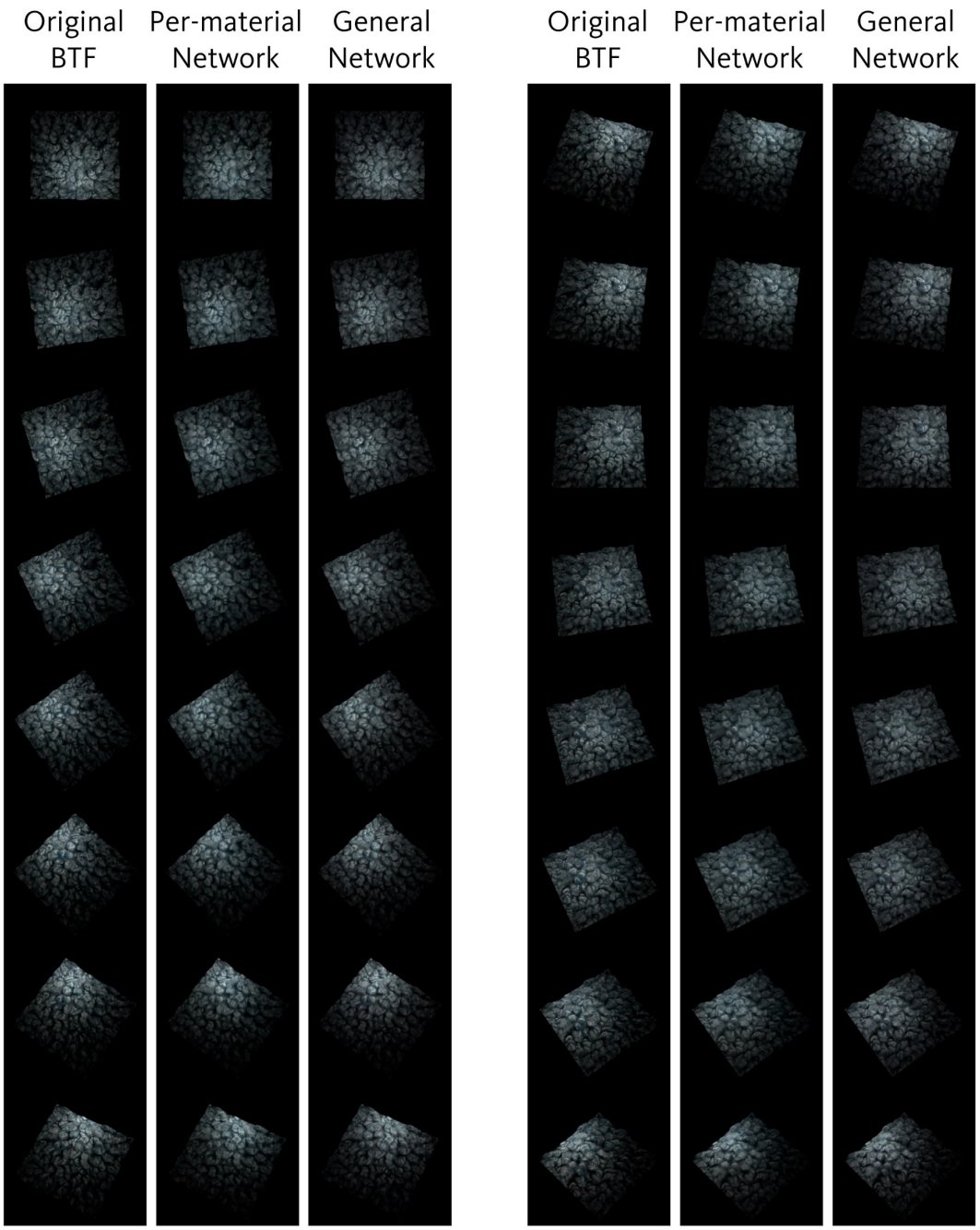


Figure C.1: Frames of the animation of a rotating plane under moving point-light illumination, for the `carpet12` dataset.

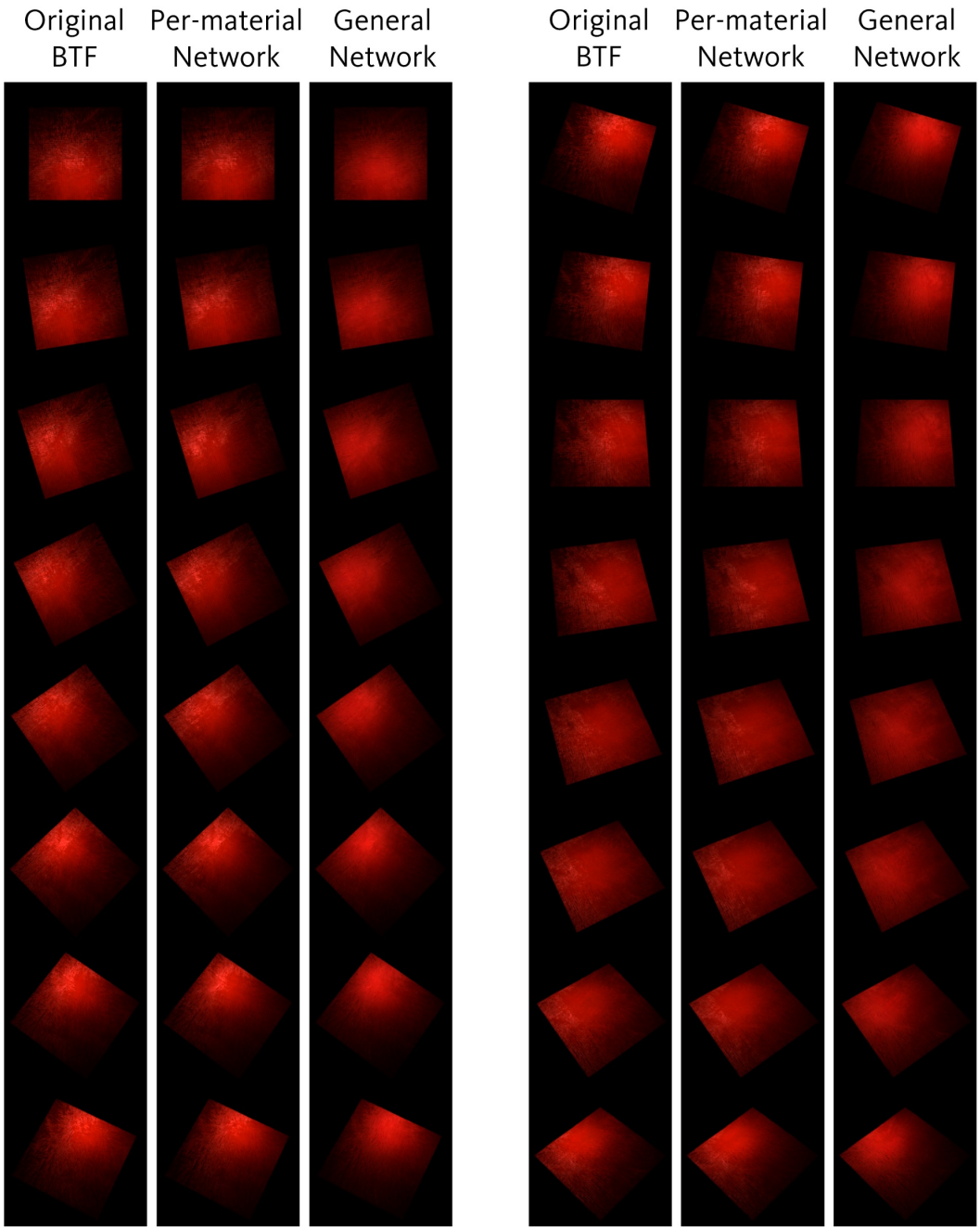


Figure C.2: Frames of the animation of a rotating plane under moving point-light illumination, for the `fabric12` dataset.

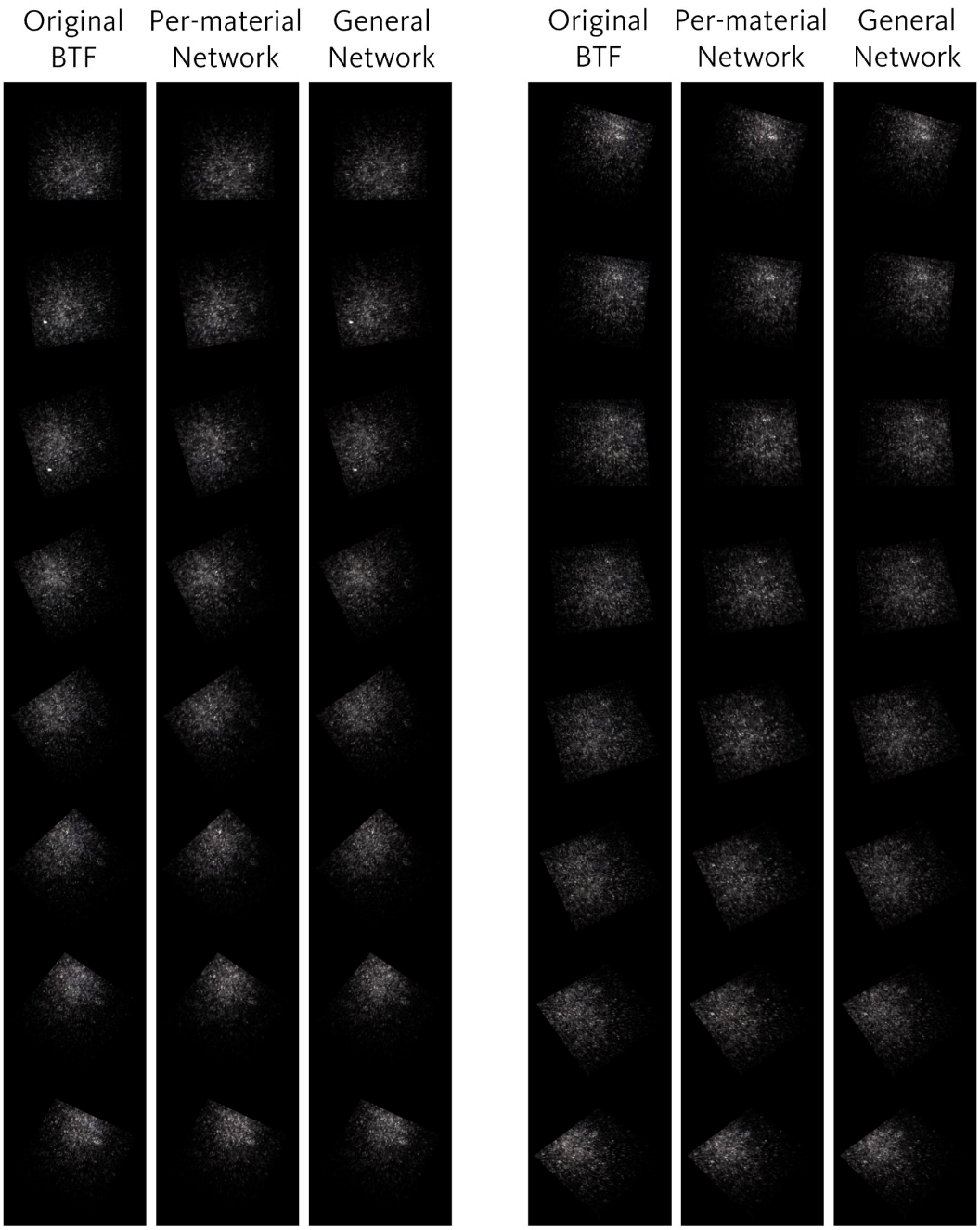


Figure C.3: Frames of the animation of a rotating plane under moving point-light illumination, for the `felt12` dataset.

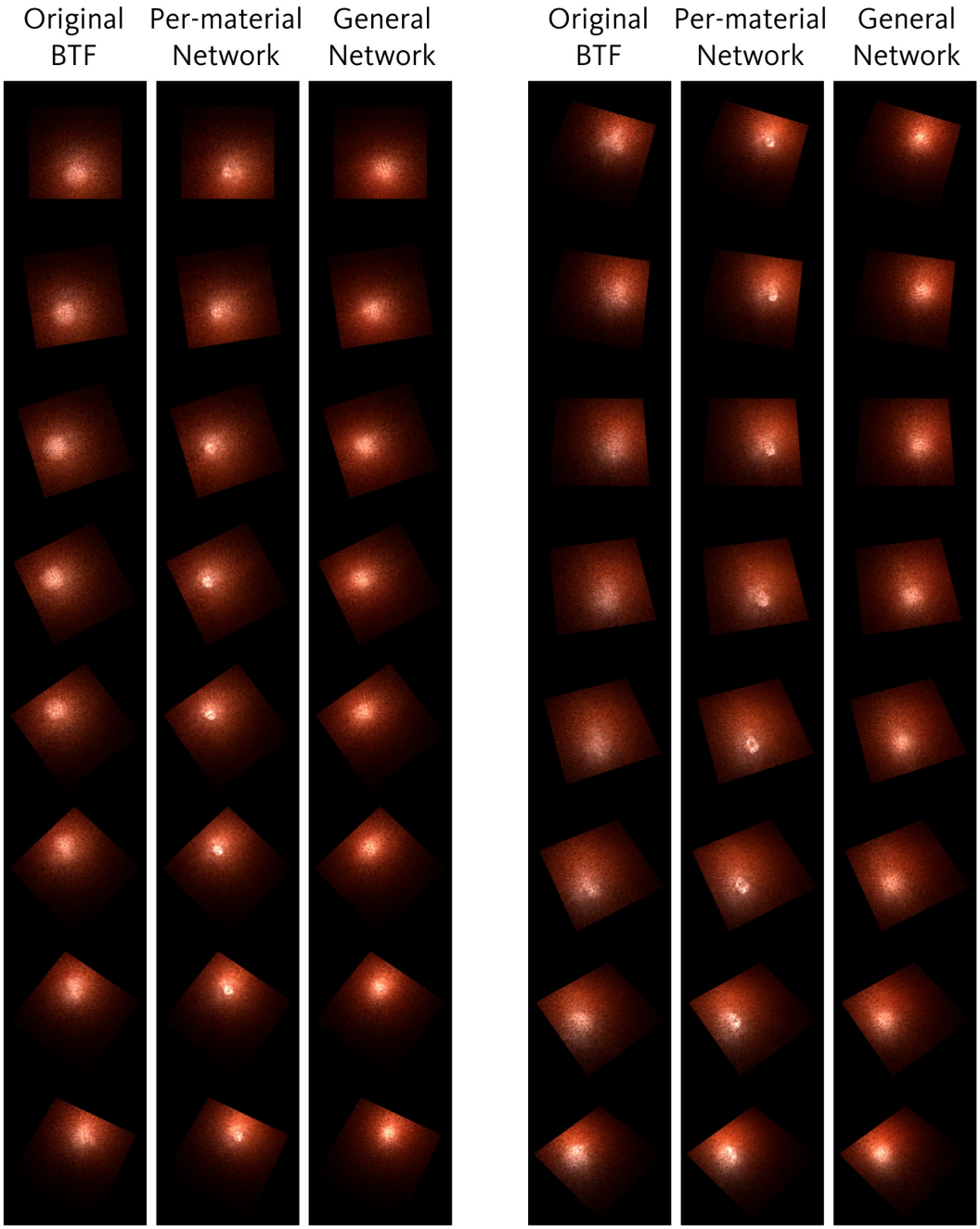


Figure C.4: Frames of the animation of a rotating plane under moving point-light illumination, for the `leather12` dataset.

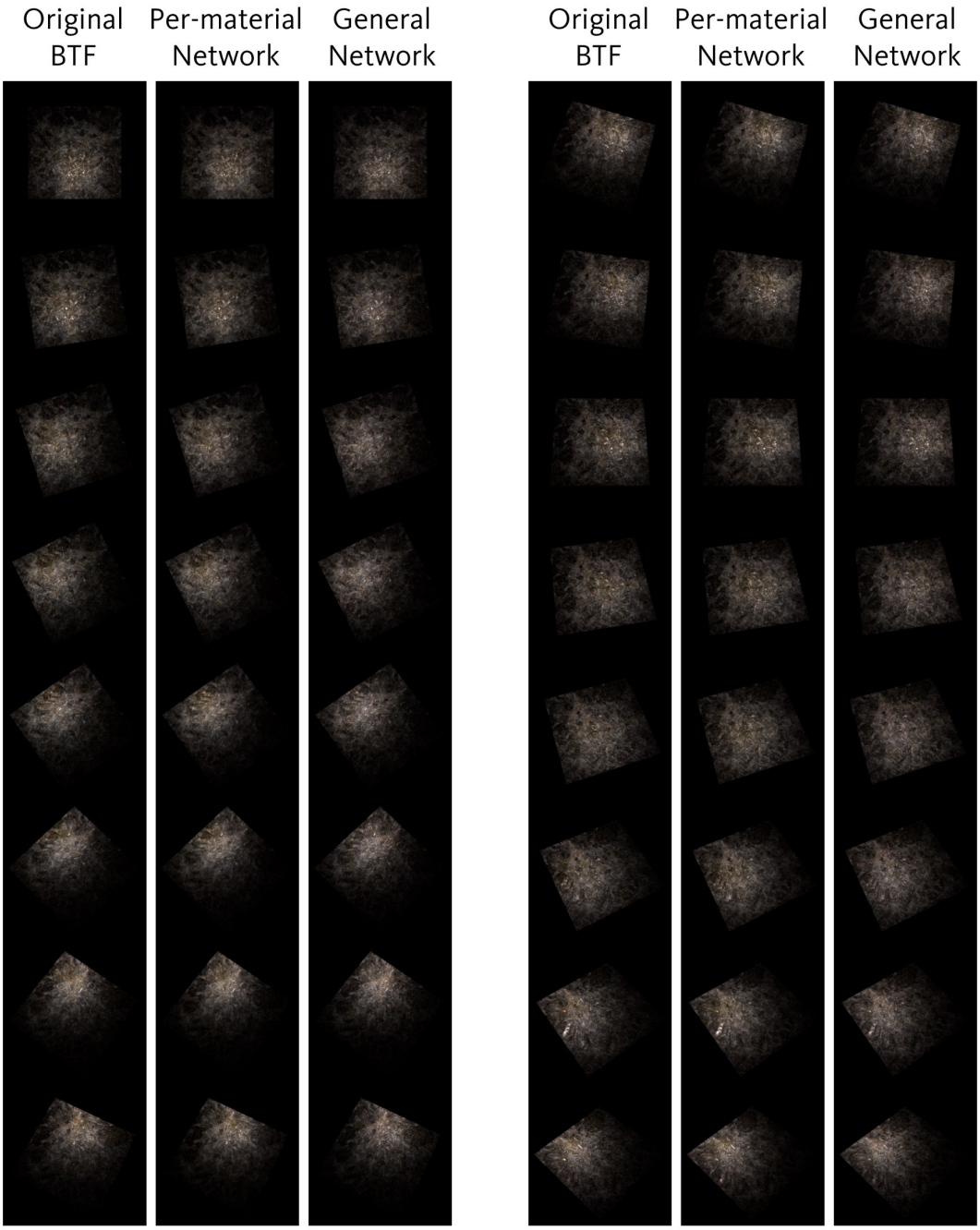


Figure C.5: Frames of the animation of a rotating plane under moving point-light illumination, for the `stone12` dataset.

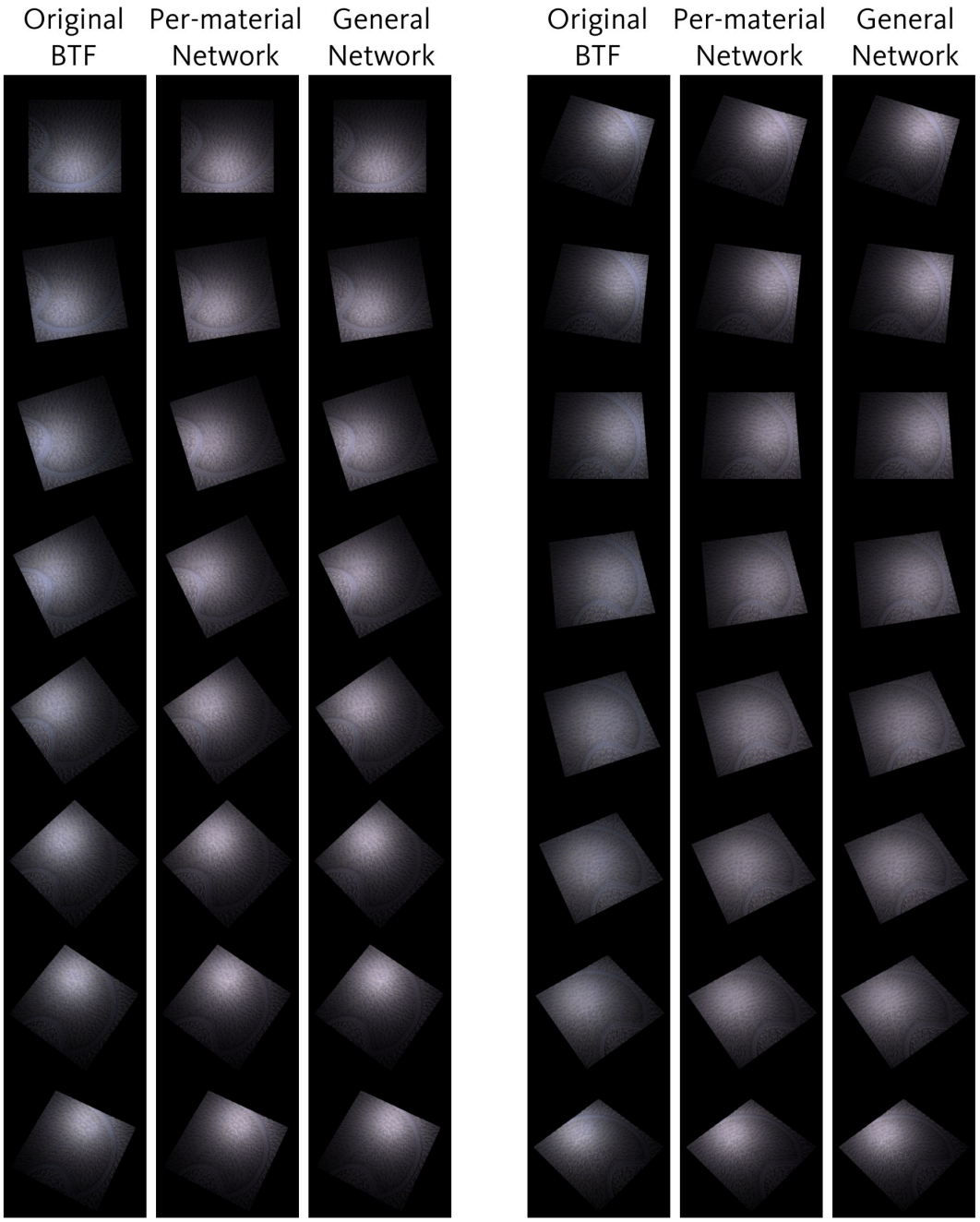


Figure C.6: Frames of the animation of a rotating plane under moving point-light illumination, for the `wallpaper12` dataset.

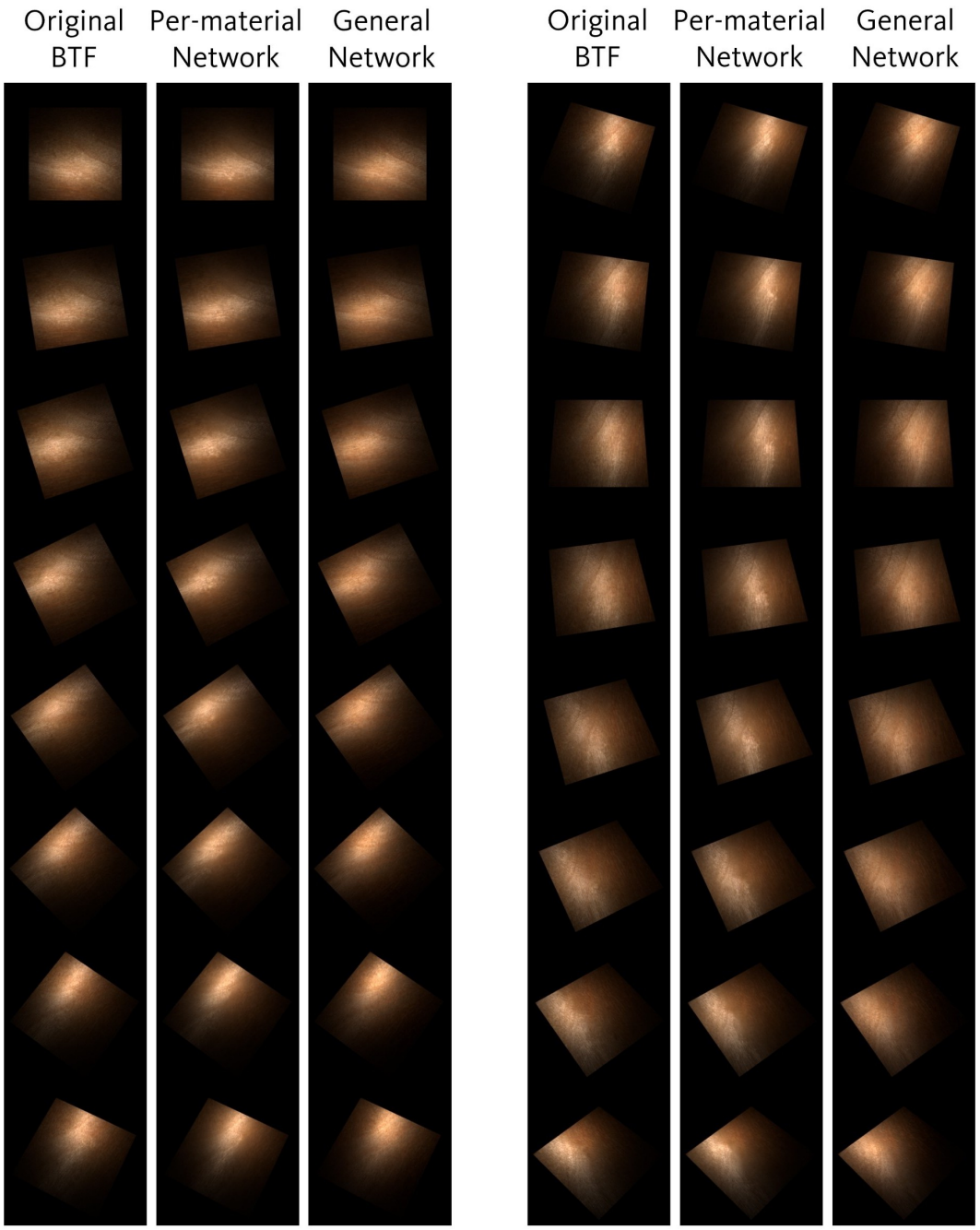


Figure C.7: Frames of the animation of a rotating plane under moving point-light illumination, for the `wood12` dataset.

Bibliography

[AAL16] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 35(4), July 2016.

[ACG+17] Carlos Aliaga, Carlos Castillo, Diego Gutierrez, Miguel A. Otaduy, Jorge Lopez-Moreno, and Adrian Jarabo. An appearance model for textile fibers. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 36(4):35–45, 2017.

[Ami02] Isaac Amidror. Scattered data interpolation methods for electronic imaging systems: a survey. *Journal of Electronic Imaging*, 11(2):157 – 176, 2002.

[AWL13] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Practical SVBRDF capture in the frequency domain. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4):110:1–110:12, August 2013.

[AWL15] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 34(4), July 2015.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 28, page 425–432, 2001.

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[BK88] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4–5):291–294, September 1988.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124–1137, Sep. 2004.

[BL19] Mark Boss and Hendrik P. A. Lensch. Single image brdf parameter estimation with a conditional adversarial network. *arXiv:1910.05148*, 2019.

[BLPW14] Adam Brady, Jason Lawrence, Pieter Peers, and Westley Weimer. Genbrdf: Discovering new analytic brdfs with genetic programming. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4), July 2014.

[BMSR19] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. Neural view-interpolation for sparse light field video. *arXiv:1910.13921*, 2019.

[CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 21(3):447–456, July 2002.

[CCZ+20] Z. Chen, A. Chen, G. Zhang, C. Wang, Y. Ji, K. N. Kutulakos, and J. Yu. A neural rendering framework for free-viewpoint relighting. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5598–5609, 2020.

[Che95] Shenchang Eric Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 29–38, 1995.

[CHH+17] J. Cáp, J. Hošek, V. Havran, Š. Němcová, and K. Macúchová. Optomechanical design of rotary kaleidoscope for bidirectional texture function acquisition. *Applied Optics*, 56(26):7373, 2017.

[Com18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 279–288, 1993.

[CWZ+18] Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. Deep surface light fields. *Proc. ACM Computer Graphics Interactive Techniques*, 1(1), July 2018.

[DAD+18] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4), July 2018.

[DAD+19] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Flexible svbrdf capture with a multi-image deep network. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 38(4), July 2019.

[DHT+00] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 145–156, 2000.

[DJ18] Jonathan Dupuy and Wenzel Jakob. An adaptive parameterization for efficient material acquisition and rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6), December 2018.

[Don19] Yue Dong. Deep appearance modeling: A survey. *Visual Informatics*, 3(2):59 – 68, 2019.

[DvGNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[DW04] Kristin Dana and Jing Wang. Device for convenient measurement of spatially varying bidirectional reflectance. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 21:1–12, 02 2004.

[EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 341–346, 2001.

[EJRB+18] S. Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari Morcos, Marta Garnelo, Avraham Ruderman, Andrei Rusu, Ivo Danihelka, Karol Gregor, David Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360:1204–1210, 06 2018.

[EV14] Petr Egert and Havran Vlastimil. Parallel BTF Compression with Multi-Level Vector Quantization in OpenCL. In *Pacific Graphics Short Papers*, 2014.

[FCGH08] Jiří Filip, Michael J. Chantler, Patrick R. Green, and Michal Haindl. A psychophysically validated metric for bidirectional texture data

reduction. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 27(5), December 2008.

[FH09a] J. Filip and M. Haindl. Bidirectional texture function modeling: A state of the art survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1921–1940, 2009.

[FH09b] Jiři Filip and Michal Haindl. Bidirectional texture function modeling: A state of the art survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(11):1921–1940, November 2009.

[FKH+18] J. Filip, M. Kolafová, M. Havlíček, R. Vávra, M. Haindl, and Rushmeier H. Evaluating Physical and Rendered Material Appearance. *The Visual Computer (Computer Graphics International 2018)*, 2018.

[FL00] Brian V. Funt and Benjamin C. Lewis. Diagonal versus affine transformations for color correction. *Journal of the Optical Society of America A*, 17(11):2108–2112, November 2000.

[FVHK17] J. Filip, R. Vávra, M. Havlíček, and M. Krupička. Predicting visual perception of material structure in virtual environments. *Computer Graphics Forum*, 36(1):89–100, 2017.

[GAHO07] A. Ghosh, S. Achutha, W. Heidrich, and M. O'Toole. Brdf acquisition with basis illumination. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, volume 90, pages 183–197, 2007.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. www.deeplearningbook.org.

[GCP+09] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Estimating specular roughness and anisotropy from second order spherical gradient illumination. In *Proc. Eurographics Symposium on Rendering*, page 1161–1170, 2009.

[GCP+10] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Circularly polarized spherical illumination reflectometry. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 29(6), December 2010.

[GEB15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 262–270, Cambridge, MA, USA, 2015. MIT Press.

[GGG+16] D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. BRDF representation and acquisition. In *Computer Graphics Forum (Eurographics State of the Art Reports)*, page 625–650, 2016.

[GGPL20] Jie Guo, Yanwen Guo, Jingui Pan, and Wenzhou Lu. Brdf analysis with directional statistics and its applications. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1476–1489, March 2020.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 43–54, 1996.

[GLD+19] DUAN GAO, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[GTHD03] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear light source reflectometry. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):749–758, July 2003.

[GTLL06] Gaurav Garg, Eino-Ville Talvala, Marc Levoy, and Hendrik P. Lensch. Symmetric photography: Exploiting data-sparseness in reflectance

fields. In *Proc. Eurographics Symposium on Rendering*, page 251–262, 2006.

[HF07] Michal Haindl and Jiri Filip. Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(10):1859–1865, October 2007.

[HFA04] Michal Haindl, Jiri Filip, and Michael Arnold. BTF image space utmost compression and modelling method. In *Proceedings of the International Conference on Pattern Recognition (ICPR'04)*, volume 3, pages 194–197, 2004.

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

[HH11] Martin Hatka and Michal Haindl. BTF Rendering in Blender. *Proceedings of the 10th International Conference on Virtual Reality Continuum and its Applications in Industry*, pages 265–272, 2011.

[HHN+17] Vlastimil Havran, Jan Hosek, Sarka Nemcova, Jiri Cap, and Jiri Bittner. Lightdrum–Portable Light Stage for Accurate BTF Measurement on Site. *Sensors*, 17(3), 2017.

[HLZ10] Michael Holroyd, Jason Lawrence, and Todd Zickler. A coaxial optical scanner for synchronous acquisition of 3d geometry and surface reflectance. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 29(4), July 2010.

[HM12] Vávra R. Haindl M., Filip J. Digital material appearance: the curse of tera-bytes. *ERCIM News*, (90):49–50, 2012.

[HMR19] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proc. the Conference on*

Computer Vision and Pattern Recognition (CVPR), pages 8353–8361, June 2019.

[HMRR19] P. Hermosilla, S. Maisch, T. Ritschel, and Timo Ropinski. Deep-learning the latent space of light transport. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 38:207–217, 07 2019.

[HP03] Jefferson Y. Han and Ken Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):741–748, July 2003.

[HPP+18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6), December 2018.

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

[IM06] Piti Irawan and Stephen Marschner. A Simple, Accurate Texture Model for Woven Cotton Cloth. *Technical report PCG-06-01, Program of Computer Graphics, Cornell University*, pages 1–8, June 2006.

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

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 511–518, 2001.

[JWD+14] Adrian Jarabo, Hongzhi Wu, Julie Dorsey, Holly Rushmeier, and Diego Gutierrez. Effects of approximate filtering on the appearance of bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 20(6):880–892, June 2014.

[Kaj86] James T. Kajiya. The rendering equation. In *Computer Graphics (Proc. SIGGRAPH), ACM*, page 143–150, 1986.

[KCL18] Yong Hwi Kim, Junho Choi, and Kwan Heng Lee. An efficient method for specular-enhanced BTF compression. *Computer & Graphics*, 75:1–10, June 2018.

[KCW+18] Kaizhang Kang, Zimin Chen, Jiaping Wang, Kun Zhou, and Hongzhi Wu. Efficient reflectance capture using an autoencoder. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4), July 2018.

[KGT+17] Kihwan Kim, Jinwei Gu, Stephen Tyree, Pavlo Molchanov, Matthias Nießner, and Jan Kautz. A lightweight approach for on-the-fly reflectance estimation. *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 20–28, 2017.

[KHX+19] Alexandr Kuznetsov, Miloš Hašan, Zexiang Xu, Ling-Qi Yan, Bruce Walter, Nima Khademi Kalantari, Steve Marschner, and Ravi Ramamoorthi. Learning generative models for rendering specular microgeometry. *ACM Trans. Graph.*, 38(6), November 2019.

[KK89] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Computer Graphics (Proc. SIGGRAPH), ACM*, page 271–280, 1989.

[KM06] Naoki Kawai and Kazuo Matsufuji. Azimuth-rotated vector quantization for BTF compression. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, 2006.

[KMBK03] Melissa L. Koudelka, Sebastian Magda, Peter N. Belhumeur, and David J. Kriegman. Acquisition, compression, and synthesis of bidirectional texture functions. In *In ICCV 03 Workshop on Texture Analysis and Synthesis*, 2003.

[Kol10] Vladimir Kolmogorov. Max-flow/min-cut. http://mouse.cs.uwaterloo.ca/code/maxflow-v3.01.zip, 2010.

[KWKT15] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2539–2547. Curran Associates Inc., 2015.

[KXH+19] Kaizhang Kang, Cihui Xie, Chengan He, Mingqi Yi, Minyi Gu, Zimin Chen, Kun Zhou, and Hongzhi Wu. Learning efficient illumination multiplexing for joint capture of reflectance and shape. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 38(6), November 2019.

[KZG18] Christos Kampouris, Stefanos Zafeiriou, and Abhijeet Ghosh. Diffuse-specular separation using binary spherical gradient illumination. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, page 1–10, 2018.

[LBAD+06] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 735–745, 2006.

[LDPT17] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4):45:1–45:11, July 2017.

[LG05] Hugo Ledoux and Christopher Gold. An efficient natural neighbour interpolation algorithm for geoscientific modelling. In *Developments in Spatial Data Handling*, pages 97–108, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM*

Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), page 31–42, 1996.

[LHWD17] Chloe LeGendre, Loc Hyunh, Shanhe Wang, and Paul Debevec. Modeling vellus facial hair from asperity scattering silhouettes. In *ACM SIGGRAPH 2017 Talks*, 2017.

[LMS+19] Manuel Lagunas, Sandra Malpica, Ana Serrano, Elena Garces, Diego Gutierrez, and Belen Masia. A similarity measure for material appearance. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[LSC18] Zhengqin Li, Kalyan Sunkavalli, and Manmohan Krishna Chandraker. Materials for masses: SVBRDF acquisition with a single mobile phone image. *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[LSS+19] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4):65:1–65:14, July 2019.

[LSSS18] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):68:1–68:13, July 2018.

[LXR+18] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6), December 2018.

[LZB17] Fujun Luan, Shuang Zhao, and Kavita Bala. Fiber-level on-the-fly procedural textiles. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 36(4):123–135, July 2017.

[MBK05] Gero Müller, Gerhard H. Bendels, and Reinhard Klein. 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*, VAST'05, pages 13–20. Eurographics Association, 2005.

[MBR+13] R. Marques, C. Bouville, M. Ribardière, L. P. Santos, and K. Bouatouch. Spherical fibonacci point sets for illumination integrals. *Computer Graphics Forum*, 32(8):134–143, 2013.

[MCT+05] Wan-Chun Ma, Sung-Hsiang Chao, Yu-Ting Tseng, Yung-Yu Chuang, Chun-Fa Chang, Bing-Yu Chen, and Ming Ouhyoung. Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 187–194. ACM, 2005.

[MG09] N. Menzel and M. Guthe. g-BRDFs: An Intuitive and Editable BTF Representation. *Computer Graphics Forum*, 2009.

[MGW01] Tom Malzbender, Dan Gelb, and Hans Wolters. Polynomial texture maps. *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 519–528, 2001.

[MHP+19] Abhimitra Meka, Christian Häne, Rohit Pandey, Michael Zollhöfer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. Deep reflectance fields: High-quality facial reflectance field inference from color gradient illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[MHRK19] S. Merzbach, M. Hermann, M. Rump, and R. Klein. Learned fitting

of spatially varying BRDFs. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 38(4):193–205, 2019.

[MLH02] David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '02, pages 79–88. Eurographics Association, 2002.

[MM06] Jonathan T. Moon and Stephen R. Marschner. Simulating multiple scattering in hair using a photon mapping approach. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):1067–1074, July 2006.

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003*, pages 271–280. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[MMR+19] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(5), October 2019.

[MMS+04] G Mueller, J Meseth, M Sattler, R Sarlette, and R Klein. Acquisition, Synthesis and Rendering of Bidirectional Texture Functions. *Eurographics 2004. STAR: State of the art report*, 24(1):83–109, 2004.

[MPBM03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, page 759–769, 2003.

[MRLF19] M. Maximov, T. Ritschel, L. Leal-Taixé, and M. Fritz. Deep appearance maps. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 8728–8737, 2019.

[MRP98] Gavin Miller, Steven Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, pages 281–292. Springer Vienna, 1998.

[MSOC+19] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv:2003.08934*, 2020.

[MSY07] Y. Mukaigawa, K. Sumino, and Y. Yagi. High-speed measurement of brdf using an ellipsoidal mirror and a projector. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[MTK+10] Yasuhiro Mukaigawa, Seiichi Tagawa, Jaewon Kim, Ramesh Raskar, Yasuyuki Matsushita, and Yasushi Yagi. Hemispherical confocal imaging using turtleback reflector. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ACCV'10, page 336–349. Springer-Verlag, 2010.

[MUGL08] Rosana Montes, Carlos Ureña, Rubén García, and Miguel Lastra. Generic brdf sampling - a sampling method for global illumination. In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications - Volume 1: GRAPP, (VISIGRAPP 2008)*, pages 191–198. INSTICC, SciTePress, 2008.

[Mül09] Gero Müller. *Data-Driven Methods for Compression and Editing*

of Spatially Varying Appearance. Dissertation, Universität Bonn, December 2009.

[MWLT00] Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, and Kenneth E. Torrance. Image-based bidirectional reflectance distribution function measurement. *Appl. Opt.*, 39(16):2592–2600, Jun 2000.

[NDM05] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proc. Eurographics Symposium on Rendering*, EGSR '05, page 117–126. Eurographics Association, 2005.

[NPLBY18] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems (Proc. NeurIPS)*, volume 31, 2018.

[NRH+77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, Oct 1977.

[NZLVG05] Alexander Neubeck, Alexey Zalesny, Van Luc, and Luc Van Gool. 3d texture reconstruction from extensive BTF data. *4th International Workshop on Texture Analysis and Synthesis*, pages 13–18, 10 2005.

[OP05] Manuel M. Oliveira and Fabio Policarpo. An Efficient Representation for Surface Details. *Proceedings of the 14th International Multimedia Modeling Conference*, 55(51):1–8, 2005.

[QSMG17] C. R. Qi, H. Su, Kaichun Mo, and L. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[RDL+15] Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo.

Image based relighting using neural networks. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 34(4), July 2015.

[RGJW20] Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. Unified neural encoding of BTFs. *Computer Graphics Forum (Proc. Eurographics)*, 39(2):167–178, July 2020.

[RJGW19] Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. Neural BTF compression and interpolation. *Computer Graphics Forum (Proc. Eurographics)*, 38(2):235–244, 2019.

[RK09] Roland Ruiters and Reinhard Klein. BTF compression via sparse tensor decomposition. *Proc. Eurographics Symposium on Rendering*, pages 1181–1188, 2009.

[Roy99] Sébastien Roy. Stereo Without Epipolar Lines: A Maximum-Flow Formulation. *International Journal of Computer Vision*, 34(2):147–161, aug 1999.

[RPG16] J. Riviere, P. Peers, and A. Ghosh. Mobile surface reflectometry. *Computer Graphics Forum*, 35(1):191–202, February 2016.

[RRW19] Gilles Rainer, Kevin Rathbone, and Tim Weyrich. High-resolution BTF capture for delicate materials. In *Conference on Visual Media Production (CVMP) Posters*, December 2019.

[RSK13] Roland Ruiters, Christopher Schwartz, and Reinhard Klein. Example-based interpolation and synthesis of bidirectional texture functions. *Computer Graphics Forum (Proceedings of the Eurographics 2013)*, 32(2):361–370, May 2013.

[RWG+13] Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. Global illumination with radiance regression functions. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4):130:1–130:12, July 2013.

[Sch96] D. Scharstein. Stereo vision for view synthesis. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 852–858, June 1996.

[Sch15] Christopher Schwartz. *Acquisition, Transmission and Rendering of Objects with Optically Complicated Material Appearance*. Dissertation, Universität Bonn, 2015.

[SD96] Steven M. Seitz and Charles R. Dyer. View morphing. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 21–30, 1996.

[SGM+16] Ana Serrano, Diego Gutierrez, Karol Myszkowski, Hans-Peter Seidel, and Belen Masia. An intuitive control space for material appearance. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 35(6), November 2016.

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

[SJR18] Tiancheng Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. Connecting measured BRDFs to analytic BRDFs by data-driven diffuse-specular separation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6):273:1–273:15, December 2018.

[SMB+20] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[SMG20] Paul Sanzenbacher, Lars Mescheder, and Andreas Geiger. Learning neural light transport. *arXiv:2006.03427*, 2020.

[SPS13] Nuno Silva and Luís Paulo Santos. Interactive high fidelity visualization of complex materials on the GPU. *Computer & Graphics, Technical Section*, 37(7):809–819, November 2013.

[SS97] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *ACM Proc. the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 251–258, 1997.

[SSK03] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Efficient and realistic visualization of cloth. In *Proc. Eurographics Workshop on Rendering*, page 167–177, 2003.

[SSN18] Cyril Soler, Kartic Subr, and Derek Nowrouzezahrai. A versatile parameterization for measured material manifolds. *Computer Graphics Forum (Proc. Eurographics)*, 37(2):135–144, 2018.

[SSW+14] Christopher Schwartz, Ralf Sarlette, Michael Weinmann, Martin Rump, and Reinhard Klein. Design and implementation of practical bidirectional texture function measurement devices focusing on the developments at the university of bonn. *Sensors*, 14(5), April 2014.

[SSWK13] Christopher Schwartz, Ralf Sarlette, Michael Weinmann, and Reinhard Klein. Dome II: A parallelized BTF acquisition system. In Holly Rushmeier and Reinhard Klein, editors, *Proceedings of the Eurographics Workshop on Material Appearance Modeling: Issues and Acquisition*, pages 25–31. Eurographics Association, June 2013.

[Sta02] Stanford. Stanford graphics lab spherical gantry, 2002. http://graphics.stanford.edu/projects/gantry/.

[SWRK11] Christopher Schwartz, Michael Weinmann, Roland Ruiters, and Reinhard Klein. Integrated high-quality acquisition of geometry and appearance for cultural heritage. In *Proceedings of the International Symposium on Virtual Reality, Archeology and Cultural Heritage VAST*, pages 25–32. Eurographics Association, October 2011.

[SZW19] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene

representations. In *Advances in Neural Information Processing Systems (Proc. NeurIPS*, 2019.

[TANS05] M Tsuchida, Hiroyuki Arai, M Nishiko, and Yoshiyuki Sakaguchi. Development of BRDF and BTF measurement and computer-aided design systems based on multispectral imaging. *Proc. AIC05*, 1, 01 2005.

[TFG+13] Borom Tunwattanapong, Graham Fyffe, Paul Graham, Jay Busch, Xueming Yu, Abhijeet Ghosh, and Paul Debevec. Acquiring reflectance and shape from continuous spherical harmonic illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4), July 2013.

[TFLS11] Yu-Ting Tsai, Kuei-Li Fang, Wen-Chieh Lin, and Zen-Chung Shih. Modeling bidirectional texture functions with multivariate spherical radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(7):1356–1369, July 2011.

[TFT+20] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (Eurographics State of the Art Reports)*, 2020.

[TZL+02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 21(3):665–672, July 2002.

[TZN19] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019.

[VF16] R. Vávra and J. Filip. Minimal sampling for effective acquisition of anisotropic brdfs. *Computer Graphics Forum (Proc. Pacific Graphics)*, 35(7):299–309, October 2016.

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions in Image Processing*, 13(4):600–612, April 2004.

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

[WGK14] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a BTF database. *Proc. European Conference on Computer Vision (ECCV)*, pages 156–171, 2014.

[WLL+08] Tim Weyrich, Jason Lawrence, Hendrik Lensch, Szymon Rusinkiewicz, and Todd Zickler. Principles of appearance acquisition and representation. *Foundations and Trends in Computer Graphics and Vision*, 4(2):75–191, 2008.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. Eurographics Symposium on Rendering*, EGSR'07, page 195–206. Eurographics Association, 2007.

[WMP+06] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, and Markus Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):1013–1024, 2006.

[WSRK11] Michael Weinmann, Christopher Schwartz, Roland Ruiters, and Reinhard Klein. A multi-camera, multi-projector super-resolution frame-

work for structured light. In *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 397–404. IEEE Computer Society's Conference Publishing Services, May 2011.

[WWHL07] R. Peter Weistroffer, Kristen R. Walcott, Greg Humphreys, and Jason Lawrence. Efficient basis decomposition for scattered reflectance data. In *Proc. Eurographics Symposium on Rendering*, volume 18 of *EGSR'07*, pages 207–218. Eurographics Association, 2007.

[WWS+05] Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):527–535, July 2005.

[XSHR18] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):126:1–126:13, July 2018.

[YBS+19] Ning Yu, Connelly Barnes, Eli Shechtman, Sohrab Amirghodsi, and Michal Lukáč. Texture mixer: A network for controllable synthesis and interpolation of texture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[ZFWW18] Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. Gaussian material synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):76:1–76:14, July 2018.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(11):1330–1334, November 2000.

[ZJMB12] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala.

Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph.*, 31(4), July 2012.

[ZJMB14] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Building volumetric appearance models of fabric using micro ct imaging. *Commun. ACM*, 57(11):98–105, October 2014.

[ZRL+09] Arno Zinke, Martin Rump, Tomás Lay, Andreas Weber, Anton Andriyenko, and Reinhard Klein. A practical approach for photometric acquisition of hair color. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 28(5):1–9, December 2009.

[ZW07] Arno Zinke and Andreas Weber. Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):342–356, March 2007.

[ZYWK08] Arno Zinke, Cem Yuksel, Andreas Weber, and John Keyser. Dual scattering approximation for fast multiple scattering in hair. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3):1–10, August 2008.