Data-Driven Analysis and Interpolation of Optical Material Properties

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Dipl.-Inf. Roland Artur Ruiters

aus Köln

Bonn, Oktober 2013

Universität Bonn
Institut für Informatik II
Friedrich-Ebert-Allee 144, D-53113 Bonn

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms Universität Bonn

Dekan: Prof. Dr. U.-G. Meißner

1. Referent: Prof. Dr. Reinhard Klein
2. Referent: Prof. Dr. Christian Bauckhage

Tag der Promotion: 15.12.2014
Erscheinungsjahr 2015

Contents

ZUSAMMENFASSUNG

Die digitale Reproduktion des charakteristischen Erscheinungsbildes eines Materials ist von erheblicher Bedeutung für die Erzeugung von fotorealistischen Bildern. Aufgrund der hohen Komplexität vieler realer Materialien ist ihre Modellierung jedoch ein schwieriges Problem. Daher wurde in den letzten Jahren viel an datengetriebenen Techniken geforscht. Bei diesen Verfahren wird das optische Erscheinungsbild einer realen Materialprobe vermessen. Dies ermöglicht deren akkurate Wiedergabe und damit die Erzeugung von synthetischen Bildern, die nur schwer von Fotografien des Materials zu unterscheiden sind. Dennoch haben diese Verfahren noch keine breite Anwendung in der Praxis gefunden. Dies liegt hauptsächlich daran, dass die Messungen nach wie vor aufwändig und teuer und die resultierenden Datensätze groß und schwierig zu verarbeiten und editieren sind. Diese Dissertation besteht aus zwei Teilen, die sich diesen Problemen durch die Entwicklung von effizienten Repräsentationen und Interpolationstechniken für optische Materialeigenschaften widmen.

Der erste Teil beschäftigt sich mit Repräsentationen für optische Materialeigenschaften und Techniken zur Rekonstruktion dieser Repräsentationen anhand von Messungen. Zuerst wird dabei eine deutlich schnellere Kompression der klassischen auf einer PCA basierenden Repräsentation für BTFs untersucht. Hierfür wird eine GPU-beschleunigte Technik, um die PCA von großen Datenmatrizen schnell zu berechnen, beschrieben. Weiterhin werden kompakte Repräsentationen für BRDFs, basierende auf einer PARAFC Tensorfaktorisierung, und für BTFs, basierend auf einer dünnbesetzten Tensorzerlegung, eingeführt. Schließlich werden Techniken, um das Reflexionsverhalten eines Materials anhand einer dünnen und unregelmäßig abgetasteten Messung zu rekonstruieren, entwickelt. Hierbei werden sowohl eine Höhenkarte zusammen mit eine Mixtur analytischer BRDFs als auch eine Summe separierbarer Funktionen als mögliche Repräsentationen untersucht.

Im zweiten Teil werden Materialeditiertechniken, die auf der Interpolation zwischen mehreren vermessenen Materialproben basieren, erforscht. Hierfür werden datengetriebene Interpolationstechniken für BRDFs, Texturen und BTFs entwickelt. Es wird gezeigt, dass es möglich ist, mit diesen Techniken plausible Interpolationssequenzen zu erzeugen. Dies gelingt sogar für Materialien mit komplexer Featuretopologie, räumlich variierendem Reflexionsverhalten und einer Mesostruktur, die starke Parallaxen verursacht. Diese Techniken bilden die Grundlage für intuitive und leistungsfähige Anwendungen zum Editieren von Materialien, welche es einem Benutzer ermöglichen, neue Materialien zu designen, indem er die Charakteristika mehrerer vermessener Proben kombiniert.

Abstract

Reproducing the characteristic appearance of materials digitally is of considerable importance for the creation of photo-realistic images. However, due to the high complexity of many real-world materials, modeling them accurately is a difficult problem. Therefore, data-driven techniques have received considerable attention in the last years. In these approaches, the optical material properties from actual material samples are measured. This enables their faithful reproduction and the creation of synthetic images which are difficult to distinguish from actual photographs of the material. Still, these techniques have not yet found wide-spread practical application. This is mainly due to the fact that the measurement process is still time-consuming and expensive and that the resulting datasets are large and difficult to process and edit. This dissertation is split into two parts which address these questions by providing more efficient representations and interpolation techniques for optical material properties.

The first part is concerned with representations for optical material properties and techniques to derive these representations from measurements. First, we investigate considerable improvements of the compression speed of the classical PCA based representation of BTFs. For this, we describe a GPU accelerated technique to compute the PCA of very large data-matrices. Then, we introduce a compact representation for BRDFs, based on a PARAFAC tensor decomposition, and for BTFs, based on a sparse tensor decomposition. Finally, we develop techniques to reconstruct the reflectance behavior of a material from a sparse and irregular input sampling either using a representation via a heightfield and a mixture of analytical BRDFs or by fitting a sum of separable functions to the sparse samples.

In the second part, we explore material editing approaches based on the interpolation between measured exemplars. For this, we develop data-driven interpolation techniques for BRDFs, textures and BTFs. We demonstrate that it is possible to create believable interpolation sequences even for materials with complex feature topology, spatially varying reflectance behavior and a meso-structure resulting in strong parallaxes. These techniques provide the foundation for an intuitive and powerful material editing approach, which gives the end user the ability to create a new material by combining the characteristics of several measured samples.

ACKNOWLEDGEMENTS

I would like to thank everybody who has helped and enabled me to write this thesis. First and foremost, this is my supervisor Prof. Dr. Reinhard Klein who provided the initial vision for this thesis, supported and guided me throughout its development, but also allowed me the freedom to explore ideas of my own.

Second, I have to thank all the members of the working group for the help they provided and the friendly and creative atmosphere and especially my co-authors Christopher Schwartz, Michael Weinmann, Prof. Dr. Renato Pajarola, Martin Rump, Dr. Ruwen Schnabel, Dr. Susanne Suter, Aljosa Osep, Ralf Sarlette, and Dr. Arno Zinke without whom this work would never have been possible. I have to particularly thank Ralf Sarlette for building the BTF acquisition devices, which provided the data this thesis is built upon. I also would like to thank my office mates Sebastian Dunkel, Kai Schröder, Christopher Schwartz and Raoul Wessel for all the help and support they provided and sometimes also for the patience and endurance they had with me. And in particular, I thank Raoul Wessel for long discussions about important topics such as fusion reactors, interstellar travel, strong AI, Dyson Spheres and Matrioshka Brains, and in general everything related to saving (and destroying) the earth.

I would also like to express my gratitude to Prof. Christian Bauckhage for being the second reviewer and to Prof. Dr. Jürgen Gall and Prof. Dr. Marc Alexander Schweitzer for kindly agreeing to serve as members of my thesis committee. Furthermore, I have to thank the Deutsche Forschungsgemeinschaft, the European Union and the Universität Bonn for providing the funding enabling my research.

And finally, I have to thank my parents for the support they provided.

CHAPTER 1

INTRODUCTION

Due to the tremendous progress in computer graphics, today it is possible to create renderings which are almost indistinguishable from photographs. However, with advancements in rendering technology and the increasing fidelity of the created images, content creation has become more and more demanding. Whereas for the first computer generated renderings the scene description consisted only of a few geometric primitives and material parameters, today objects are described by millions of polygons and very sophisticated material descriptions are used. For a truly photo-realistic image, an accurate description of all aspects of reality which influence the final appearance would be required. Manually creating all the input data at this level of detail is very labor-intensive and thus in practice often infeasible.

To solve this problem, data-driven techniques have become ubiquitous. Instead of creating the scene description manually, the necessary data is measured from real-world counterparts and then integrated into the image creation process. This paradigm finds application for nearly all types of input that are required. 3D-scanning is used to acquire the geometry of objects, *High Dynamic Range* (HDR) light probes allow to obtain the illumination conditions in a real-world scene, motion capture from actors enables the realistic animation of characters and even the distortion of lenses or the appearance of lens-flares are measured and integrated into the rendering process.

Though this approach finds application in many areas, the focus of this thesis will be on the representation of the reflectance properties of materials. Here, the acquisition from real-world samples is also an established technique. Starting with measurement of individual material parameters and the use of textures acquired via photographs it culminated in sophisticated acquisition devices which provide detailed measurements of the appearance of a material under a multitude of view and illumination conditions.

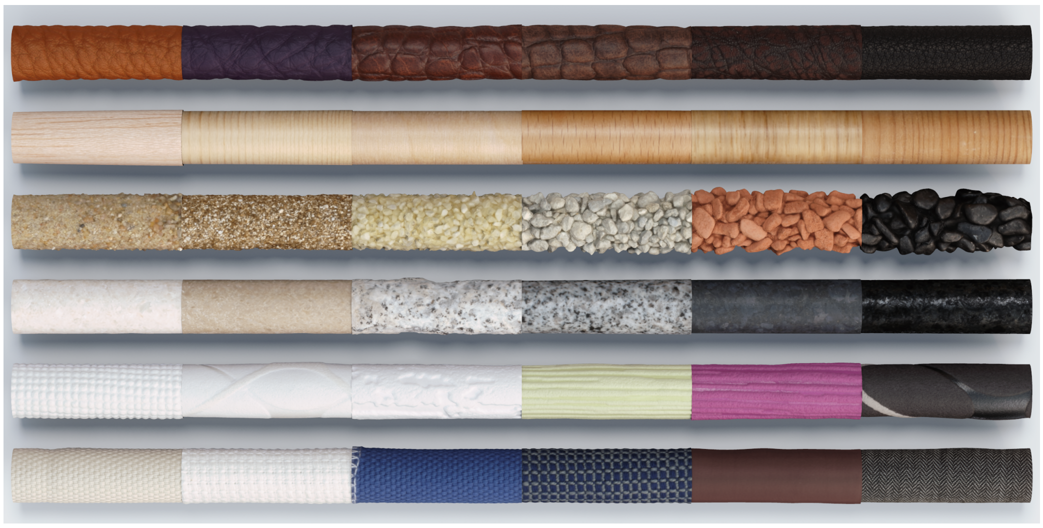


Figure 1.1: *Rendering of a selection of measured BTFs (using heightfields to reproduce the silhouette as described in Chapter 10).*

This effort is necessary due to the high complexity of an accurate description of a material. The amount of light that is reflected by a surface towards an observer can vary both with the light and the view direction. This reflectance behavior can also exhibit a complex spatial variation and in many cases the appearance of a material is actually characterized by small geometric features on the surface such as bumps, ridges or scratches, which can result in complex interreflections, occlusions and shadowing. Some materials also show strong sub-surface scattering increasing the complexity even further.

As an example, Figure 1.1 shows a rendering of several material samples acquired from real-world counterparts. To measure the reflectance behavior of these samples, images under a large number of illumination and view directions were taken. This approach enables the realistic reproduction of the appearance of a wide range of materials, without the need to manually model each of them separately.

The efficient and accurate measurement is a challenging problem and an active area of research in its own right. However, this thesis is mainly concerned with processing, representing and editing the acquired data. Usually, the measured data cannot be used directly for rendering. Instead, it is necessary to derive a representation which has to solve a number of problems:

Compact Representation A material measurement can result in a very large dataset, easily containing several hundred GBs of raw data. To allow for efficient use in rendering applications, it is thus necessary to find a representation which is far more compact. This representation should fulfill several requirements. Ideally, it should accurately reproduce the measured data, be as compact as possible, allow for efficient random access during rendering and last but not least it should be possible to derive this representation with acceptable computational effort from the measurement.

Incomplete Sampling The sampling provided by a measurement device will necessarily be incomplete. Due to constrains on the complexity of the device and the acceptable acquisition time, there will always be limitations in the resolution and angular sampling a measurement can provide. Furthermore, when acquiring the reflectance behavior of objects, for some parts on the surface there will be missing data due to occlusions and shadows during the measurement process. When deriving a representation for later rendering, it is thus necessary to impute these missing data in a reasonable way.

Material Editing In many use cases, the goal of computer graphics is not the exact reproduction of the real world, but the creation of new content based on the acquired data. Therefore, the chosen representation should offer the user the possibility to edit the acquired data in a meaningful way. A very intuitive and powerful approach for this is the use of interpolation techniques, which provide the end user with the ability to create a material by combining the characteristics of several measured samples. By interpolating between a number of acquired samples, it is possible to create a ”material space” which can then be explored by the user.

1.1 Thesis Outline and Main Contributions

This thesis is divided into two main parts, where the first part is concerned with **material representations** (Chapters 3–7) and the second part describes **material interpolation** techniques (Chapters 8–10). The main scientific contributions in Part I are:

- **Parallelized Matrix Factorization** Over many years, the use of matrix factorization techniques has proven to be one of the most successful approaches to compress reflectance data. However, computing these factorizations for the large datasets that result from measurements can be a very time consuming process. In Chapter 3, we introduce a technique which subdivides the matrix factorization into several independent sub-problems each of which can be solved on a GPU. This provides a considerable speed-up compared to a sequential CPU implementation and thus makes the use of factorization based compression techniques feasible, even for very large datasets.

- **BTF Compression via Sparse Tensor Decomposition** Several tensor approximation based techniques for the compression of BTF datasets have been proposed. However, these techniques suffer from the fact that random access into the decomposed tensor is very expensive and thus these techniques are not well-suited for rendering. In Chapter 4, we show that by utilizing a decomposition of a tensor representing the BTF as several sparse tensors, very high compression rates and at the same time a good decompression performance can be achieved.
- **Heightfield and SVBRDF Reconstruction** Representing a material as a heightfield and a spatially varying linear-combination of analytical basis BRDFs provides a very compact material representation, which easily allows for editing operations such as changes in color or specularity. However, deriving such a representation from measured data is a challenging problem. In Chapter 5, we describe an optimization technique that combines photometric and multi-view stereo to reconstruct an accurate heightfield and estimates the parameters of the basis materials and their distribution. Additionally, it performs simulations of the light exchange to compensate for the influence of interreflections.
- **PARAFAC based BRDF Representation** Accurately representing a measured BRDF via an analytical model results in a very difficult optimization problem. Data-driven representations on the other hand often require far more storage space. In Chapter 6, we show that a homogeneous BRDF can be represented very compactly via a PARAFAC tensor approximation. For this, the data has to be represented via a half-angle/difference-angle parameterization and a suitable weighting to compensate for the large dynamic range of BRDFs has to be applied.
- **Surface Reflectance from Sparse and Irregular Samples** The PARAFAC based representation introduced in the previous chapter is only suitable for homogeneous materials and requires a dense reflectance measurement as input. In Chapter 7, we generalize this representation to spatially varying materials and demonstrate a technique to reconstruct the representations from a much sparser and irregularly sampled measurement. For this, we introduce additional regularization constraints which enforce smoothness and exploit spatial coherence.

Part II introduces data-driven interpolation techniques for BRDFs, textures and finally BTFs. Its main contributions are:

- **BRDF Interpolation via Dynamic Time Warping** While the proposed PARAFAC based BRDF representation allows for a very compact storage and accurate representation of the acquired materials, an intuitive editing is not directly possible. A straightforward linear interpolation between measured materials can result in unintuitive results. In Chapter 8, we introduce a technique, which performs perceptually plausible interpolations by using dynamic time warping to align BRDF features. Based on this approach, an intuitive user interface for the interactive exploration of the space spanned by a database of BRDFs becomes possible.
- **Patch-based Texture Interpolation** Interpolating between two textures is a challenging problem. When the two textures have different feature topologies it is not possible to bring these features into alignment and thus interpolation via linear blending or warping is not possible. In Chapter 9, we introduce a technique relying on patch-based texture synthesis to perform this interpolation. Instead of aligning all features in the input images, we find corresponding patches for which aligning the features is much easier. These interpolated patches are then finally used to reassemble the interpolated texture.
- **BTF Interpolation** The patch-based interpolation technique presented in the previous chapter was limited to textures. In Chapter 10, we extend this approach to the interpolation of measured BTFs. A straightforward application of the algorithm would not be feasible due to the large size of the BTFs. However, by taking advantage of a factorized representation and utilizing a heightfield to compensate parallaxes, it is possible to create plausible interpolations between a wide range of different materials.

In addition to these chapters, in Chapter 2, a short summary of the background, important previous work and an overview of the utilized notations is provided. Part III finally concludes the thesis and discusses remaining open questions and avenues of future research.

1.2 Publications

As is common in computer graphics, most of the work presented in this thesis has previously been published. In particular, it is based on the following publications:

- Roland Ruiters, Martin Rump, and Reinhard Klein. Parallelized Matrix Factorization for fast BTF Compression. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 25–32, March 2009.
- Roland Ruiters and Reinhard Klein. Heightfield and spatially varying BRDF Reconstruction for Materials with Interreflections. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2):513–522, April 2009.
- Roland Ruiters and Reinhard Klein. BTF Compression via Sparse Tensor Decomposition. *Computer Graphics Forum (Proceedings of EGSR)*, 28(4):1181–1188, July 2009.
- Roland Ruiters, Ruwen Schnabel, and Reinhard Klein. Patch-based Texture Interpolation. *Computer Graphics Forum (Proceedings of EGSR)*, 29(4):1421–1429, June 2010.
- Roland Ruiters and Reinhard Klein. A compact and editable representation for measured BRDFs. Technical Report CG-2010-1, University of Bonn, December 2010.
- Roland Ruiters, Christopher Schwartz, and Reinhard Klein. Data Driven Surface Reflectance from Sparse and Irregular Samples. *Computer Graphics Forum (Proceedings of Eurographics)*, 31(2):315–324, May 2012.
- Roland Ruiters, Christopher Schwartz, and Reinhard Klein. Example-based Interpolation and Synthesis of Bidirectional Texture Functions. *Computer Graphics Forum (Proceedings of Eurographics)*, 32(2), 2013.
- Roland Ruiters and Reinhard Klein. BTF based Material Representations: Current Challenges. In *Eurographics Workshop on Material Appearance Modeling: Issues and Acquisition*, pages 17–20. Eurographics Association, June 2013.

Parts of the work from Chapters 4, 6, and 7 have also been presented by the author in a tutorial at the Eurographics 2013:

- Renato Pajarola, Susanne K. Suter, and Roland Ruiters. Tensor approximation in visualization and computer graphics. In *Eurographics 2013 - Tutorials*, number T6, May 2013.

The work in Chapter 7 and Chapter 10 is also partially based on techniques described in the publications below. In these papers, however, the main focus is on the acquisition of geometry and reflectance of objects or streaming and rendering of BTFs instead on material acquisition or editing and therefore they are not included into this thesis.

- Michael Weinmann, Christopher Schwartz, Roland Ruiters, and Reinhard Klein. A Multi-Camera, Multi-Projector Super-Resolution Framework for Structured Light. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 397–404, May 2011.
- Christopher Schwartz, Michael Weinmann, Roland Ruiters, and Reinhard Klein. Integrated High-Quality Acquisition of Geometry and Appearance for Cultural Heritage. In *The 12th International Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST)*, pages 25–32, October 2011.
- Christopher Schwartz, Roland Ruiters, Michael Weinmann, and Reinhard Klein. WebGL-based Streaming and Presentation Framework for Bidirectional Texture Functions. In *The 12th International Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST)*, pages 113–120, October 2011. Best Paper Award.
- Christopher Schwartz, Michael Weinmann, Roland Ruiters, Arno Zinke, Ralf Sarlette, and Reinhard Klein. Capturing Shape and Reflectance of Food. In *SIGGRAPH Asia 2011 Sketches*, SA '11, pages 28:1–28:2, December 2011.
- Michael Weinmann, Roland Ruiters, Aljosa Osep, Christopher Schwartz, and Reinhard Klein. Fusing Structured Light Consistency and Helmholtz Normals for 3D Reconstruction. In *Proceedings of the British Machine Vision Conference*, pages 108.1–108.12, September 2012.
- Christopher Schwartz, Roland Ruiters, Michael Weinmann, and Reinhard Klein. WebGL-based Streaming and Presentation of Objects with Bidirectional Texture Functions. *Journal on Computing and Cultural Heritage (JOCCH)*, 6(3):11:1–11:21, July 2013.
- Christopher Schwartz, Roland Ruiters, and Reinhard Klein. Level-of-Detail Streaming and Rendering using Bidirectional Sparse Virtual Texture Functions. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, 32(7):345–354, October 2013.
- Michael Weinmann, Aljosa Osep, Roland Ruiters, and Reinhard Klein. Multi-View Normal Field Integration for 3D Reconstruction of Mirroring Objects. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

CHAPTER 2

PRELIMINARIES

In every-day life, the term *material* is usually used to describe the substance an object is made of. A large number of attributes are associated with this term, which far surpass its optical appearance. Some examples are the haptic impression a human has when touching the object, how hard or soft it is, whether it is elastic or brittle, its thermal conductivity, or even the sounds generated when the material collides with something else. Furthermore, the term material is actually strongly dependent on the scale where the distinction between the actual geometry of the object and its material takes place. For example, sand can be considered as the material a sandcastle is made of, but it could also be described as granules made of silica.

Computer graphics usually takes a far more limited view. Here, the term material is used to describe the way incoming light interacts with an object of a given geometry. Still, this can encompass a very wide range of interactions, ranging from reflection, transmission or scattering to effects like fluorescence, phosphorescence or even changes in polarization and phase. Again, what is considered as material strongly depends on the scale. Generally speaking, every aspect influencing the appearance of an object that is not explicitly modeled in the geometry has to be represented in the material. However, it is important to keep in mind that this distinction can be made at different scales depending on the stage of the image creation process. For example, a displacement map might be considered as part of a material by the artist, but would be handled as geometry by the rendering system.

Though a wide range of different material descriptions exists in computer graphics, most of them can be regarded as a function ρ which specifies the proportionality between the incoming and outgoing light. The main difference between these descriptions is on which parameters of the incoming and the outgoing light the proportionality depends. A wide range of parameters, such as position on the surface or in the volume, direction of the light, wavelength or color channel, polarization and point in time can be taken into account. Each of these parameters can be specified both for the incoming and the outgoing light separately. With each additional parameter, the dimensionality of ρ increases further, making a complete characterization of the material increasingly difficult. This is especially the case if the behavior has to be measured from a real-world sample. Apart from the effort for the measurement and storage of this function, the rendering complexity also grows exponentially with the number of parameters taken into account. According to the superposition principle, the amount of outgoing light is determined by integrating over all the parameters characterizing the incoming light. For these reasons, a wide range of different material descriptions exists, each of which depends on a different set of parameters and thus offers a different trade-off between complexity and the range of materials that can be described this way.

In the scope of this thesis, we will mainly focus on *Bidirectional Reflectance Distribution Functions* (BRDFs), *Spatially Varying Bidirectional Reflectance Distribution Functions* (SVBRDFs) and *Bidirectional Texture Functions* (BTFs). The BRDF was introduced by Nicodemus [Nic65] and describes a homogeneous, opaque surface. The BRDF can be considered as a statistical description of microscopic surface features, like tiny ridges or facets, which are so small that they cannot be seen individually but which still influence the reflectance behavior.

The BRDF is a four-dimensional function ρ that depends on the direction of the incoming light ω_i and the direction of the reflected light ω_o. It is defined as the ratio between the radiance of the reflected light dL_o $[\mathrm{W}/(\mathrm{m}^2\,\mathrm{sr})]$ and the irradiance of the incoming light dE_i $[\mathrm{W}/\mathrm{m}^2]$:

$$\rho(\omega_i, \omega_o) = \frac{\mathrm{d}L_o(\omega_o)}{\mathrm{d}E_i(\omega_i)} \tag{2.1}$$

When the surface of an object is not homogeneous, it is necessary to store a different BRDF for every point $\mathbf{x}$ on its surface. This results in the six-dimensional SVBRDF $\rho(\mathbf{x}, \omega_i, \omega_o)$. The definitions of both the BRDF and the SVBRDF are based on the assumption, that the surface is completely opaque and that the light is therefore reflected at exactly the position where it hits the surface. Non-local light interactions are not taken into account. To enforce physical correctness, the function ρ thus has to be reciprocal ($\rho(\omega_i, \omega_o) = \rho(\omega_o, \omega_i)$) and energy conserving ($\forall \omega_o \in \Omega_o : \int_{\Omega_i} \rho(\omega_i, \omega_o) \cos(\theta_i), \mathrm{d}\omega_i < 1$, where Ω_i and Ω_o denote the hemispheres of all incoming and outgoing light directions respectively).

In contrast to the SVBRDF, which describes a physical model, the BTF, introduced by Dana et al. [DvGNK97], is a data-driven representation. It is usually acquired by a measurement from a real-world sample. For this, images of a material sample are taken from multiple view directions and under multiple illumination directions. The BTF can thus be regarded as a view and light direction dependent texture. Formally, it is also defined as a six-dimensional function $\rho(\mathbf{x}, \omega_i, \omega_o)$ but, in contrast to an SVBRDF, reciprocity and energy conservation are not required and thus non-local effects can be included exactly as they were observed during the measurement.

For example, many materials have small-scale geometric features, which can still be seen individually but are difficult to describe via geometry. These features are often called *meso-structure*, as they lie in between the microscopic structure, that is small enough to be described purely statistical via a BRDF, and the macroscopic structure, that is described explicitly via the geometry. These features can introduce a wide range of non-local effects into the BTF, where the emitted light at one point is no longer strictly proportional to the incoming light at the same point but actually depends on the light reaching the material at different points. For example, when seen from different view directions, features will move in the image due to parallax and some parts of the material might be occluded by others. When the light direction changes, shadows can move and interreflections or sub-surface scattering can result in a brightening of different parts of the material.

Since all of these effects are included in the measurement of the original material sample, a BTF is able to preserve the distinctive appearance of a material very faithfully. This is even possible for very complex materials for which an exact simulation of these non-local effects would be very challenging. However, a correct reproduction of the appearance of a surface is only achieved this way under the assumption of homogeneous parallel incoming light. In practice, as long as the scale of these non-local effects remains small compared to the variation of the light in the scene it provides a good approximation of the actual appearance.

Since the amount of light that is reflected depends on the wavelength of the light, it is additionally necessary to store BRDFs, SVBRDFs and BTFs in dependence on the wavelength of the light. For a correct color reproduction, it would be necessary to store multispectral or even hyperspectral datasets. However, in the scope of this thesis, we will only consider RGB colors and thus only three color channels are stored in all our datasets. We will also not consider fluorescence or phosphorescence. Furthermore, we do only take sub-surface scattering implicitly into account where it was captured within a BTF, but do not model it explicitly, which would require the use of a *Bidirectional Scattering-Surface Reflectance Distribution Function* (BSSRDF) [NRH*77].

2.1 Reflectance Acquisition

Since a large number of different devices for the acquisition of BRDFs, SVBRDF and BTFs have been suggested and a full description is not possible within the scope of this work, we refer to the following surveys [MMS*04, WLL*09, FH09]. However, all datasets used in this thesis were actually acquired with only four different devices. In the following, we will provide a short description and the most important specifications of these devices.

2.1.1 Gonioreflectometer at the University of Bonn

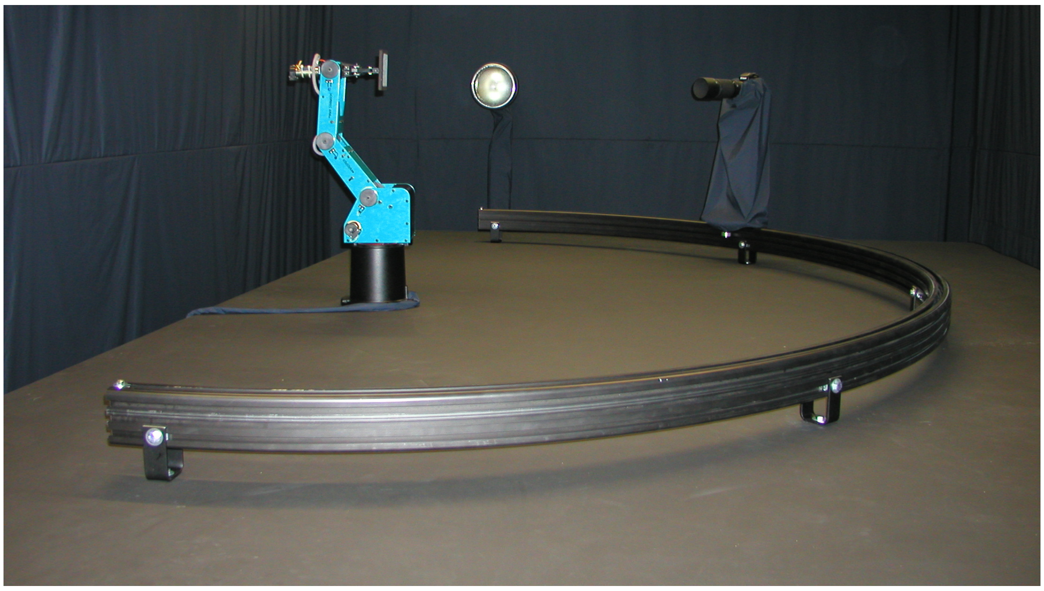


Figure 2.1: *Gonioreflectometer-like setup at the University of Bonn for the acquisition of BTFs*

This BTF acquisition device, first described in [SSK03], is a gonioreflectometer-like setup which allows to independently position both a camera (initially a Kodak DCS 760 with 6 MP, later upgraded to a Kodak DCS Pro 14N with 16 MP) and a light source (Hydrargyrum Medium Arc Length Iodide HMI lamp) with respect to the material sample. For this, the sample is moved by a robot arm (Intellitek SCORBOT-ER4u) and the camera is mounted on a rail with a radius of 170 cm. Although this configuration allows for nearly arbitrary samplings of view and light direction, all datasets used in this thesis use a sampling with all combinations of 81 view and 81 light directions distributed approximately uniformly on a hemisphere. This results in a total of $6{,}561$ images. These images were captured in LDR and had a spatial resolution of about $100\,\mu\mathrm{m}$ per pixel (for the *top-view*, the case where the view direction is orthogonal to the sample). The maximum sample size is $10\,\mathrm{cm} \times 10\,\mathrm{cm}$. A measurement with the Kodak DCS Pro 14N of one material with this setup takes about $14\,\mathrm{h}$ and results in about $77\,\mathrm{GB}$ of losslessly compressed raw data. After post-processing, this results in a BTF dataset with a typical resolution of $81 \times 81 \times 256 \times 256$ and thus an uncompressed size of about 2.5 GB. The acquisition device has later been updated to capture spectral BTFs [RSK10b], but for this thesis only RGB datasets are used.

2.1.2 MERL BRDF Database

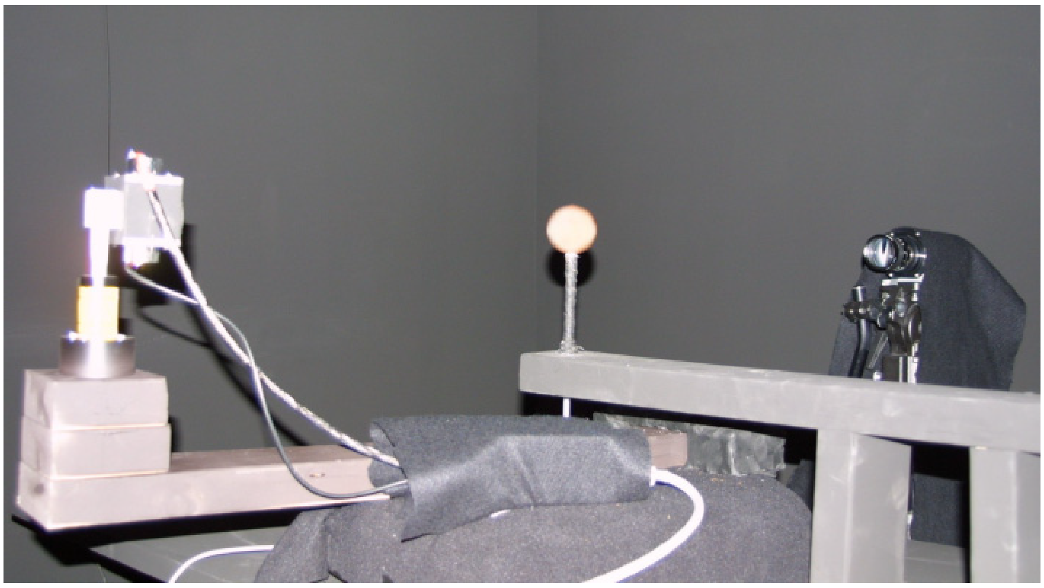


Figure 2.2: *Acquisition device that was used for the acquisition of the isotropic MERL BRDF database (image from [MPBM03a])*

A large database with 100 isotropic BRDFs has been made publicly available for research purposes by the Mitsubishi Research Laboratories at http://www.merl.com/brdf/. Details on the acquisition device and the processing of the data can be found in [MPBM03a, MPBM03b, Mat03]. The device is based on the setup first proposed in [MWLT00]. It consists of a fixed camera (QImaging Retiga 1300, 1.3 MP, 10-bit, RGB) and a light source (Hamamatsu SQ Xenon lamp) which can be rotated around the sample. By using a spherical, homogeneous material sample this setup is capable of capturing a large number of BRDF samples with each image. The measurement process then requires about four hours and results in a total of 330 HDR images per sample. From these images, $20 - 30$

million BRDF samples are extracted and then resampled to a regular grid with a resolution of $180 \times 90 \times 90$. To increase the resolution at the highlight, the parameterization [Rus98] via half-angle and difference-angle is employed and additionally the square-root is applied to θ_h. When stored as uncompressed single precision floating point values, one isotropic BRDF in this format requires about $16.7\,\mathrm{MB}$.

2.1.3 Multi-View Dome at the University of Bonn

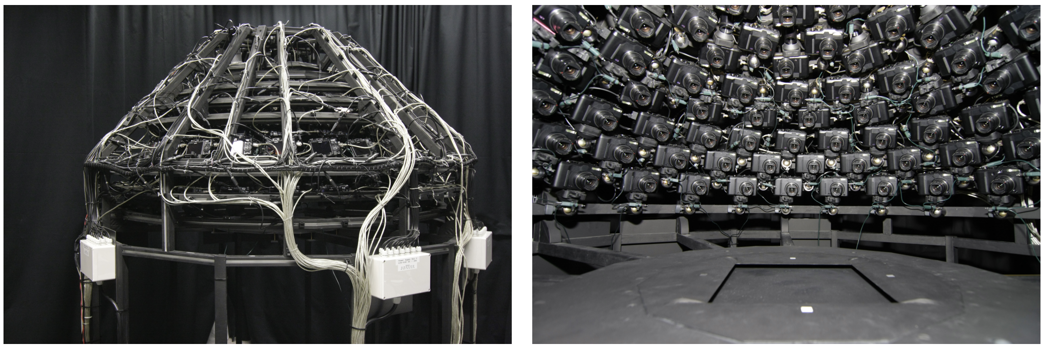


Figure 2.3: *Massively parallel BTF acquisition setup at the University of Bonn*

This massively parallel setup was built to improve the acquisition time of the sequential gonioreflectometer setup and at the same time improve the angular sampling. It was first described in [MMS*04] (additional details are given in [MBK05] and [RMS*08]) and consists of 151 consumer cameras mounted on a hemispherical gantry with a radius of approximately $80\,\mathrm{cm}$. In the initial version, Canon PowerShot A75s with $3.2\,\mathrm{MP}$ were used, which have later been upgraded to Canon PowerShot G9s with $12\,\mathrm{MP}$. The flashes in these cameras serve as light sources to illuminate the sample. Thus a measurement contains a total of $151 \times 151 = 22{,}801$ images. The sample-holder has a size of $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ giving a resolution of approximately $50\,\mu\mathrm{m}$ per pixel. However, since the cameras are equipped with a zoom optic, even higher resolution are possible for smaller samples. By combining different flash intensity and camera sensitivity (ISO value) settings, it is possible to capture HDR sequences with this setup. A full measurement with four HDR shots takes about $2\,\mathrm{h}$ and generates about $290\,\mathrm{GB}$ of JPEG compressed raw data. Depending on the sample size and selected resolution, this results in datasets with sizes ranging from $151 \times 151 \times 128 \times 128$ (uncompressed $2\,\mathrm{GB}$) to $151 \times 151 \times 512 \times 512$ (uncompressed $33\,\mathrm{GB}$) after postprocessing.

This setup is also equipped with 8 LG HS200G projectors to enable structured light geometry reconstructions. We developed a technique that takes advantage of the fact that the same sample is illuminated from several projectors to perform projector-super-resolution [WSRK11]. This helps to offset the fact that due to the construction of the setup only small projectors with a low resolution at a rather high distance to the sample can be used. This technique allows us to acquire geometry and reflectance of objects in one measurement [SWRK11]. However, this geometry reconstruction can also be used to derive a heightfield of a planar material sample (see Chapter 10 for more details). The geometry acquisition requires approximately $75\,\mathrm{min}$.

2.1.4 Mobile Dome with Camera-Arc at the University of Bonn

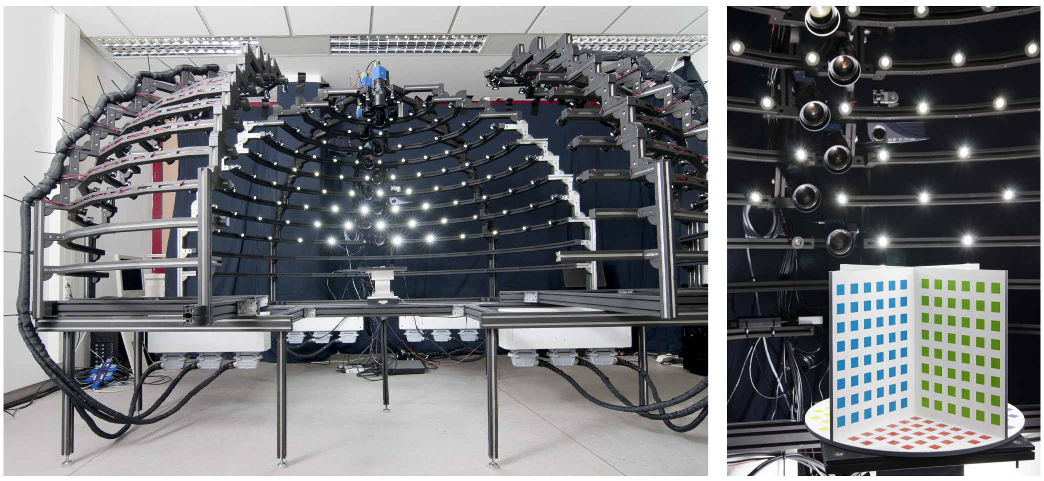


Figure 2.4: *Movable reflectance acquisition setup at the University of Bonn*

As a compromise between the sequential gonioreflectometer and the massively-parallel setup, an acquisition device consisting of 198 LED light sources placed on a hemispherical gantry with a inner radius of about $1\,\mathrm{m}$ and 11 cameras (SVS-Vistek svs4022COGE, $4\,\mathrm{MP}$, 12-bit, RGB) arranged in an arc, has been built at the University of Bonn. To compensate for the one-dimensional view sampling, a turn-table is used to rotate the sample. This way, it is possible to use high-quality cameras and lenses, which would be prohibitively costly in a fully parallel setup. The cameras can be equipped either with $50\,\mathrm{mm}$ or $100\,\mathrm{mm}$ Zeiss lenses which provide a resolution of $125.0\,\mu\mathrm{m}$ or $67\,\mu\mathrm{m}$ per pixel, respectively. HDR sequences can be created by combining images with different exposure times. Since the step-size of the turn-table settings can be chosen by the user and the number and length of exposures required depends on the material, the measurement times with this setup can vary considerably. We typically use $15°$ steps of the turn-table and an exposure series with $3-4$ exposures between usually $3\,\mathrm{ms}$ and $3\,\mathrm{s}$, depending on the dynamic range of the material. Measurement times between $4\ \mathrm{h}$ and $10\ \mathrm{h}$ are required in these cases. When four exposures are taken, this results in $209{,}088$ images with a losslessly compressed size of about $800\ \mathrm{GB}$.

This setup is also equipped with four LG HS200G projectors to perform 3D reconstructions. Similar to the reflectance acquisition, here also a varying number of turn-table and exposure settings can be used. For the geometry acquisition, we usually use $45°$ turn-table steps. Depending on the number of exposures, a total geometry acquisition time between $1.5\ \mathrm{h}$ and $3\ \mathrm{h}$ is then required. In this setup, a challenge arises due to the turn-table. Though it is possible to reconstruct one independent geometry per turn-table setting and to join the individual measurements afterwards, this prevents the use of super-resolution techniques like the one we used for the massively-parallel setup. To improve upon this, we developed a technique [WRO*12] for this setup which combines structured light and Helmholtz stereopsis and unites the information from all available turn-table settings in one common optimization problem. Even though this setup can be used to create BTFs from planar samples, in this thesis only datasets captured for 3D objects are used in Chapter 7.

2.2 Reflectance Representations

In this section, we will shortly review existing work on representations for measured surface reflectance. We will describe both techniques based on the fitting of analytical models to measurements and data-driven representations which directly use the measured data for rendering. For a more comprehensive overview, we refer to recent surveys [MMS*04, HF11].

2.2.1 Analytical BRDF Models

A very compact representation of a measured homogeneous BRDF can be obtained by fitting an analytical BRDF model to the measured data. This results in a representation which typical only requires a small number of parameters. Depending on the utilized analytical model, there can be further advantages. Many models guarantee important physical properties such as conservation of energy and reciprocity or provide analytic equations for importance sampling. A large number of different analytical models exist, which all have distinctive advantages and disadvantages. They vary on the one hand in the number and classes of materials that can be represented by a given model and on the other hand by the computational complexity of the model and the number of its parameters. For a very comprehensive survey, we refer to [MSUA12].

Several of these models have been fitted to measured material samples (e.g. in [HTSG91, War92, LFTG97, NDM05, WW07]). However, automatically fitting the models to a BRDF measurement is a difficult problem. Since many models are only suitable for a subset of all materials, one has first to decide on the correct one. Once a model has been chosen, a non-linear optimization problem has to be solved to find its parameters. This is prone to local minima, especially for complex models [NDM05]. Since it is often not possible to perfectly represent the measurement by a given model, the result may also strongly depend on the chosen error metric.

For spatially varying BRDFs, the fitting process becomes even more difficult. Apart from the quality of the individual fit, the spatial coherence between neighboring texels is also an important aspect in this case. It is possible to estimate the parameters for every texel independently [McA02, GTHD03, MK06]. However, depending on the number and distribution of available samples, this can result in a noisy SVBRDF, since each fit for an individual texel can have a different local minimum. The problem gets even worse, when the sampling varies from texel to texel. For example, when a SVBRDF is estimated for an object with a curved geometry, for every point on the surface a different sampling is available, introducing a different bias for each fit. This is especially a serious issue when estimating parameters such as specularity or the strength of the Fresnel effect, for which a reliable estimate is only possible when samples for the reflection direction or under grazing angles are available. One way to cope with this problem is to assume that certain parameters are constant on larger parts of the surface. This way, it is possible to estimate them for each of the parts instead of each texel. This can for example be performed either for the whole object [NZI01, Geo03, PCF05], for each surface patch [YDMH99] or for clusters of texels [PCDS12]. In [SWI97], the specularity parameters are only estimated at a sparse set of points, at which the conditions for the estimation of specular parameters are met, and then linearly interpolated in the spatial domain. An alternative approach is to assume that the surface consists of linear combinations of a small number of basis materials [LKG*03, HS05, GCHS05, HLZ10]. This way, the self-similarity of the material at different points on the surface can be exploited to estimate a more complex BRDF from a smaller number of per-texel samples. In [WDR11], a material is represented as a sparse combination of a large set of analytical basis materials. To further improve the quality of the approximation, an additional residual BTF is stored, that encodes the difference between the analytical model and the measured data. An alternative to these approaches, where individual point samples of the SVBRDF are measured using point-light sources, is the use of more complex illumination settings, such as polarized light and gradient illumination [MHP*07, GCP*09], a moving linear light source [GTHD03, WZT*08], or Gray code illumination [FCMB09].

A representation that is somewhere in between analytical BRDF models and data-driven techniques is proposed in [APS00]. Here, a tabulated micro-facet distribution is used. In [NDM05], this model is fitted to measured BRDFs. This technique can also be used to represent spatially varying BRDFs. In [WZT*08], a texture synthesis based approach is used to fill holes in the micro-facet distributions, whereas in [MG09] clustering is applied and for each cluster a common distribution is estimated.

Since analytical BRDF models mostly fulfill the reciprocity, and energy conservation constraints of a BRDF, it is not possible to represent materials with meso-structure using spatially varying BRDFs alone. Instead an additional geometry representation is necessary to encode these details. Many approaches utilize either a triangle mesh [SWI97, YDMH99, NZI01, LKG*03, HLZ10, PCDS12] or a heightfield [GTHD03, Geo03, PCF05, HS05, MG09] to store an explicit geometry. Some techniques [GTHD03, LKG*03, MHP*07, GCP*09] utilize normal and possibly tangent maps (sometimes in conjunction with rough geometry). In [WDR11], one normal is stored for each BRDF in the mixtures at a surface point. This way, a more complex underlying meso-structure can be approximated. In [MK06], a volumetric model is used. The material is represented as several slices, storing at each point an attenuation coefficient and a local normal.

2.2.2 Data-driven Representations

An alternative to these model based approaches are data-driven techniques. These are in principle capable of representing nearly arbitrary materials. Several of these techniques develop a BRDF with respect to a given basis. When using *Spherical Harmonics* [WAT92] or *Zernike Polynomials* [KDS96, LKK98] a large number of coefficients is needed to represent materials containing high-frequency features such as a strong narrow specular lobe, which makes the evaluation very expensive. This can be avoided by using wavelets [SS95, LF97] which reduce the number of evaluations by using a tree-structured encoding. A basis especially well-suited to representing BRDFs can be computed via *Principle Component Analysis* (PCA) from a database of materials [MPBM03a]. However, storing these basis functions itself is still very expensive if no further compression is used. In [WWHL07], a SVBRDF is represented via a linear combination of a small number of basis materials, which themselves are combinations of either *Radial Basis Functions* (RBF) or materials from the MERL BRDF database. This way, the SVBRDF can be efficiently estimated from a rather small number of measured samples. However, since the space of materials that can be represented is restricted to the chosen basis, all of these approaches have find a compromise between the generality of the representation with the size of the basis and thus the number of measured samples needed. In [DWT*10], it is therefore proposed to obtain the basis-BRDFs for a mixture based SVBRDF directly from the material sample by using an additional specialized measurement device, which captures high-resolution measurements at several sparsely sampled points on the surface.

A large number of representations are based on matrix decompositions of the reflectance data. For this, it is necessary to obtain reflectance samples on a dense and regular grid, either by directly measuring these samples or using resampling to obtain the data from an irregular sampling. Once these are available, the dataset can be represented as a matrix and then compressed by taking advantage of self-similarities in the dataset. Several types of factorizations have already been used to represent BRDFs in this way, including *Singular Value Decomposition* (SVD) [Fou95, KM99], *Homomorphic Factorization* [MAA01] and *Non-Negative Matrix Factorization* (NMF) [LRR04]. In [SBLD03], a *Chained Matrix Factorization* is used, where the data matrix is factorized repeatedly using a different parameterization each time.

Several compression techniques for BTFs are also based on Singular Value Decompositions or the closely related Principal Component Analysis. The main difference between these compression techniques lies in the question, how to arrange the reflectance data in one or several matrices which are then factorized. Sattler et al. [SSK03] grouped all images for each view direction in one matrix and then compressed these independent from each other. Müller et al. [MMK03] used *local PCA* for BTF compression by employing spatial clustering and then applying the PCA to each cluster. If instead the whole BTF data is represented as one matrix [KM03, LHZ*04], it is possible to exploit correlations throughout the full data set. Thus, the compression ratio of a *Full Matrix Factorization* (FMF) approach is superior to the other matrix factorization methods. The main problem here is the sheer size of the matrix and the resulting processing times needed to factorize this matrix. In [Mül08, GMSK09], the BTF is first decorrelated, e.g. by using a suitable color space such as YUV, and then the color channels are compressed independently from each other, each with a different number of components. This way, one can take advantage of the fact that the intensity information is usually far more complex than the color information. In [GMSK09], a perceptual weighting is additionally applied prior to the PCA computation.

Since BRDFs are actually four-dimensional functions and SVBRDFs and BTFs are even six-dimensional, a two-dimensional matrix is not their canonical representation. Instead, some of their dimensions have to be unfolded to represented these datasets in this form. Correlations along these dimensions cannot be exploited by matrix factorizations any more. In contrast to this, tensor decompositions can take advantage of these additional correlations. Several different approaches for the decomposition of a tensor exist (for an overview we refer to [KB09]). For BRDFs, both the *Tucker Decomposition* (also called *Higher-Order SVD* or *N-mode SVD*) [SZC*07] and the *Parallel Factor Analysis* (PARAFAC) (also called *Canonical Decomposition (CANDECOMP)*) [SKB10] have been applied. In [BÖK11], several chained Tucker Decompositions, each with a 1×1 core-tensor, are used, which results in a decomposition very similar to a PARAFAC.

Several BTF compression techniques based on tensor factorization have been proposed by now. In [FKIS02], a PARAFAC was used, whereas in [VT04] and [WWS*05] a Tucker Decomposition was used. However, compression techniques which are based on Tucker Decompositions have several drawbacks when compared to PCA-based representations of equal quality, as was reported by Müller in [Mül08]. On the one hand, the compression times are quite long with only small or no increase in compression ratio compared to matrix factorization based techniques. On the other hand, the reconstruction speed is very slow if only one element of the tensor is to be reconstructed, which is the standard case in BTF rendering. Wu et al. [WXC*08] use a hierarchical tensor decomposition, where first the whole dataset is approximated by one Tucker Decomposition and then the residual is subdivided into smaller blocks, which are again approximated by additional decompositions. This is then continued recursively. This way, the compression ratio can be further improved. In [Tsa09], a Tucker factorization is combined with a clustering step, which is applied along one of the tensors modes, to reduce the size of the individual tensors and enable faster decompression. In [Tsa09, TS12], this approach is extended by representing each entry along the clustered mode as a combination of k factorized tensors. This approach achieves compression ratios similar to the Tucker factorization while still providing an improvement in rendering performance. To find this representation, the authors suggest a generalization of the K-SVD [AEB06] to tensors. In [LBAD*06], a technique for the representation of SVBRDFs closely related to the tensor-based approaches is suggested. It is based on several repeated non-negative matrix factorizations, each applied after unfolding the data along a different mode. This way, the SVBRDF is decomposed into a *Shade Tree*.

However, all of these techniques require a dataset sampled on a regular grid prior to compression. If only an irregular and sparse sampling is available, it is possible to first resample these samples to regular grid [MBK05, SWRK11] and then utilize one of the afore mentioned compression techniques. However, if this approach is used, the required amount of storage during compression is no longer determined by the number of input samples but the resolution of the resampled dataset. Thus, high angular and spatial resolutions cannot be achieved, even if the compressed dataset would be small, due to the required memory during compression. In [AZK08], this problem is circumvented by directly fitting a factorization into bivariate basis BRDFs and maps describing their distribution to the irregular samples. In [ZERB05], the scattered data are used directly for rendering via an RBF interpolation. While this avoids the problems introduced by a representation on a regular grid, it requires storing the input samples (though only a subset is used) and performing the interpolation during rendering, which is rather expensive.

2.3 Notation

Here, we will shortly introduce the notations we will use throughout this thesis. We use bold face to represent vectors, matrices, and multi-indices. Lower-case letters are used for vectors (e.g. $\mathbf{v}$) and multi-indices (e.g. $\mathbf{i}$), whereas upper-case letters are used for matrices (e.g. $\mathbf{M}$). For tensors, we use script, upper-case letters (e.g. $\mathcal{T}$). In all cases, we use lower indices to denote individual elements, using the respective convention for the sub-element (e.g. $v_i, m_{i,j}, t_{\mathbf{i}} = t_{i_1,\ldots,i_D}$). When we have a collection of objects which require indices themselves, we use upper indices in brackets to denote individual elements (e.g. $\mathbf{M}^{(i)}$).

We regard a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of order N as an N-way array of tabulated data. We assume that tensors are represented in a fixed basis and do not distinguish between contravariant and covariant indices by using upper and lower indices. For tensor operations, we mostly follow the notation from [DL97]. To describe fibers and slices of tensors, we replace some of the indices with a colon to denote all elements of the corresponding mode (e.g. $\mathcal{T}_{i,:,:,:,k}$ would be the corresponding mode-3 tensor). We also use this notation to denote rows and columns of matrices (e.g. $\mathbf{m}_{i,:}$, $\mathbf{m}_{:,j}$ for a matrix $\mathbf{M}$). We use $\mathbf{v}^{(1)} \circ \cdots \circ \mathbf{v}^{(D)}$ for the outer product of vectors $\mathbf{v}^{(d)}$, $\mathcal{T} \times_n \mathbf{M}$ for the mode n product, $||\mathcal{T}||$ for the Frobenius norm and $[\mathcal{T}]_S$ to denote the sum over all entries of a tensor. For element-wise multiplication, division, square, and inverse, we use $\mathcal{A} \otimes \mathcal{B}$, $\mathcal{A} \oslash \mathcal{B}$, $\mathcal{T}^2$, and $\mathcal{T}^{-1}$. Furthermore, we use the notation $\mathcal{T}_{(I_1 I_2 \ldots I_M \times I_{M+1} \ldots I_N)}$ to describe the unfolding of a tensor into a matrix. During this operation, the first M modes are mapped onto the rows of the matrix and the remaining modes are mapped onto the columns of the matrix. This results in a $I_1 I_2 \cdots I_M \times I_{M+1} \cdots I_N$ matrix.

To write tensor decompositions, we also use the *Einstein summation convention*, which states that repeated indices are implicitly summed over (e.g. $a_{ij}b_{jk} =$

$\sum_j a_{ij} \cdot b_{jk}$). Using this convention, $\mathcal{C} = \mathcal{A}_{ij}\mathcal{B}_{jk}$ is used to denote that $\mathcal{C}$ is a tensor whose elements are given by the following expression:

$$\mathcal{C}_{i,k} = (\mathcal{A}_{ij}\mathcal{B}_{jk})_{i,k} = \sum_j a_{i,j} \cdot b_{j,k}.$$

Part I

Material Representations

CHAPTER 3

PARALLELIZED MATRIX FACTORIZATION

Several techniques for the compression of reflectance datasets, discussed in Section 2.2.2, rely on dimensionality reduction methods like Principal Component Analysis and the closely related Singular Value Decomposition. These provide very good compression ratios, and the resulting datasets are well-suited for real-time rendering. However, for large datasets like BTFs, a serious drawback of these methods is that their computation is very expensive because it requires the factorization of large matrices. Another general issue is the huge data size which makes out-of-core algorithms necessary. These are severely encumbered by IO bottlenecks and thus can get nearly unusable for large BTF datasets even if their in-core equivalents would suffice the needs.

Because of the comparatively long measurement times and the low-resolution of older BTF acquisition setups, such as the sequential gonioreflectometer at the University of Bonn (see Section 2.1.1), the compression times of these techniques were a less important question so far. Still often techniques which sacrifice compression ratio for the ability to use faster in-core algorithms were applied. These subdivide the whole BTF matrix into smaller blocks, which can be factorized independently. However, the availability of high-resolution and lower cost digital cameras has made the development of highly parallel BTF acquisition devices possible. Modern devices, such as the ones described in Section 2.1.3 and Section 2.1.4, are able to capture a material sample with high dynamic range, an angular resolution of more than 100 view and light directions and a spatial resolution of several megapixels in a few hours. This corresponds to data sizes of several hundreds of gigabytes. Current techniques do not scale well to these large datasets. For example, with non-parallelized methods the compression of a $512 \times 512 \times 95 \times 95$ BTF using an out-of-core PCA on the full BTF matrix takes about 13 hours on a Intel Q6600 with 8GB of RAM, which is in marked contrast to a few hours of measurement, severely hampering the practical operation of a BTF acquisition setup. Thus, the compression of high-resolution BTFs is still a challenging problem of high practical relevance.

In this chapter, we propose a method for the efficient and parallelizable factorization of large matrices and show its application to BTF compression. The core operations of the algorithm are performed on graphics hardware exploiting the massive parallel computing power of modern GPUs. The algorithm is designed to use only existing libraries for matrix operations and is thus very easy to implement. We achieve speed gains of up to a factor of 35 compared to implementations on a single CPU core.

Our basic idea is to subdivide the large BTF data matrix into several smaller blocks that can be processed in-core and then to use eigenspace merging to obtain the factorization of the complete matrix. We use the EM-PCA algorithm of Roweis [Row97] for the factorization of the small blocks. The runtime of this iterative algorithm is primarily dominated by matrix operations in its inner loop. By performing most of these operations on the GPU, we are able to gain a massive speed increase for the factorization of the individual matrix blocks.

This chapter corresponds to the paper "Parallelized Matrix Factorization for fast BTF Compression" by Roland Ruiters, Martin Rump, and Reinhard Klein, published in *Eurographics Symposium on Parallel Graphics and Visualization*, pages 25–32, March 2009.

3.1 Theory

Given a BTF $\rho\left(x, \omega_i, \omega_o\right)$ as a six-dimensional table with an RGB-triple in every entry, we can define a BTF data matrix $\mathbf{M}_{BTF}$ by unfolding the color channels c and the directions in one dimension as well as the spatial position x in the other one by defining indexing operators i and j. We end up with the $m \times n$ matrix $\mathbf{M}_{BTF}\left(i(\omega_i, \omega_o, c), j(x)\right) = \rho\left(x, \omega_i, \omega_o\right)[c]$ with m as the number of light and view direction combinations times the number of color channels and n as the number of texels.

Given such a $m \times n$ BTF matrix $\mathbf{M}_{BTF}$, its PCA can be calculated by first determining the column mean $\mathbf{m}$ of $\mathbf{M}_{BTF}$ and then performing a Singular Value Decomposition (SVD) of the matrix $\mathbf{M}$ obtained by subtracting this mean from $\mathbf{M}_{BTF}$. The full SVD of $\mathbf{M}$ is a decomposition $\mathbf{M} = \mathbf{U}_f \mathbf{S}_f \mathbf{V}_f^T$, with orthogonal matrices $\mathbf{U}_f$ and $\mathbf{V}_f$ and a diagonal matrix $\mathbf{S}_f$ containing the singular values sorted in descending order. Here, $\mathbf{U}_f$ is a $m \times m$ and $\mathbf{V}_f$ is a $n \times n$ matrix, but for BTF compression this representation is truncated, by only keeping the first k columns of $\mathbf{U}_f$ and $\mathbf{V}_f$ corresponding to the first k largest singular values of $\mathbf{S}_f$. In the following, we will thus only consider the $m \times k$ matrix $\mathbf{U}$, the $k \times k$ matrix $\mathbf{S}$ and the $n \times k$ matrix $\mathbf{V}$ obtained after this truncation. For most BTF materials, keeping about 100 columns is sufficient for a very faithful reproduction.

It is possible to compute the SVD of $\mathbf{M}$ by calculating the eigenvectors and eigenvalues of the matrix $\mathbf{M}\mathbf{M}^T$. However, this is not the best approach with regard to numerical precision and there exist algorithms that directly compute the SVD from $\mathbf{M}$ for all singular vectors and values at the same time (see e.g. [GL96]). Unfortunately, they require $O\left(mn^2 + m^2n + n^3\right)$ time and are furthermore not well-suited for out-of-core implementations.

Since for compression purposes only the first k eigenvalues and eigenvectors are needed, performing a full factorization is not very efficient. To overcome this problem, several techniques have been proposed by now, which allow to calculate only the k largest eigenvalues and corresponding eigenvectors in a considerably smaller amount of time and which are also better suited to out-of-core implementation. In the incremental SVD algorithm from Brand [Bra02] the data is processed column-wise by updating the eigenspace as new columns are added. Its time complexity is $O\left(mnk\right)$. Roweis [Row97] proposed an iterative expectation-maximization (EM) algorithm for PCA which also has a time complexity of $O\left(mnk\right)$. A different approach has been taken by Liu et al. [LWWT07]. They subdivided the data matrix into several blocks, performed a traditional PCA on each block and then merged the eigenspaces of the single blocks with the method of Skarbek [Ska03] to obtain the eigenspace of the whole matrix.

Blockwise processing is also the basic idea behind our approach, as it allows to parallelize the computation of the single blocks. Additionally, the blocks can be chosen in such a way that they fit into the memory of a GPU and can therefore be processed in-core. We decided to use the EM-PCA algorithm from Roweis for these subproblems because it can process one block of data at once with only a few matrix operations. In contrast to the incremental SVD method, where each column must be added successively, this allows for easy GPU acceleration.

For this, the matrix $\mathbf{M}$ is first subdivided into N blocks $\mathbf{M} = [\mathbf{M}_1 \ \cdots \ \mathbf{M}_N]$ of the respective sizes $m \times s_i$. On each of these blocks, a SVD is performed independently resulting in the matrices $\mathbf{U}_i, \mathbf{S}_i, \mathbf{V}_i$. This step thus can be easily performed in parallel. The SVDs for these blocks are then merged to finally obtain the decomposition of the complete matrix $\mathbf{M}$. See Figure 3.1 for an illustration of this process. In our implementation, we merge the matrices successively. Instead, the merging could also be performed in a binary tree, as suggested by Liu et al. [LWWT07]. As the results in their paper show, tree-structured merging does reduce the error, but only by a small amount (below 1% in all examples given there). On the other hand, when using tree-structured merging, it is necessary to store more intermediate results, increasing the memory requirements.

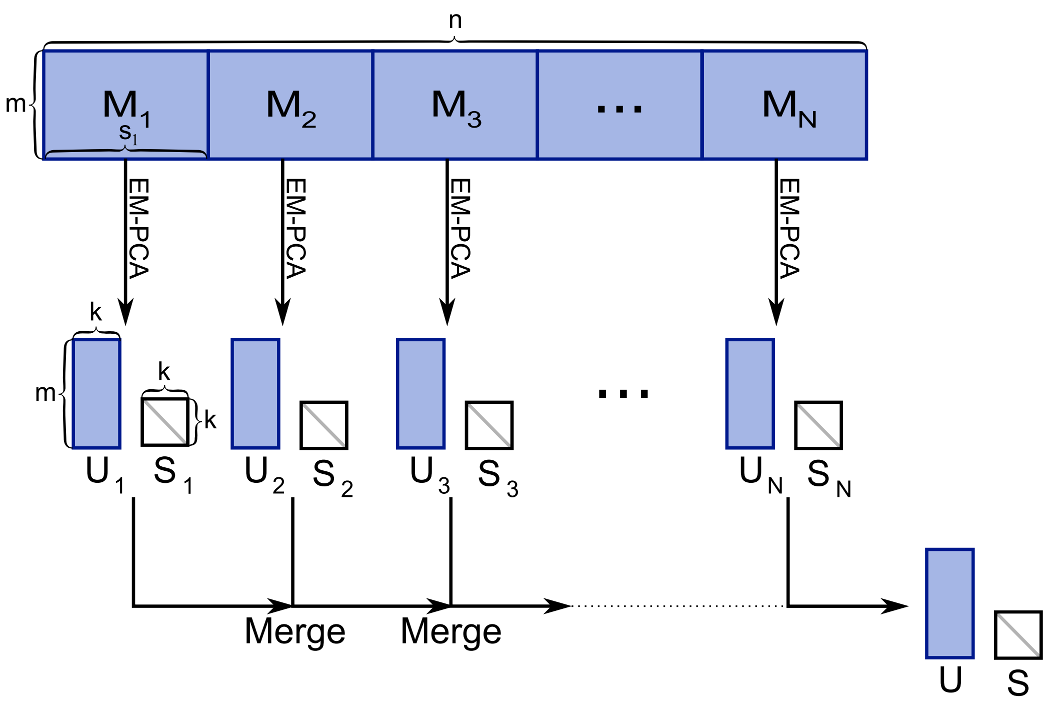


Figure 3.1: *Illustration of our parallelized matrix factorization algorithm . The matrix* $\mathbf{M}$ *is first divided into blocks* $\mathbf{M}_1, \cdots, \mathbf{M}_N$*. For each of the blocks, an independent SVD is calculated via the EM-PCA algorithm to obtain* $\mathbf{U}_i$ *and* $\mathbf{S}_i$*. Then, the individual decompositions are merged to obtain the final result* $\mathbf{U}$ *and* $\mathbf{S}$*.*

Since the matrix $\mathbf{V}$ contains the projection of $\mathbf{M}$ into the $\mathbf{U}$-space, parts of the data vectors orthogonal to this space are not represented. In each merge step, however, the subspace spanned by the matrix $\mathbf{U}$ changes. Thus, if the vectors in $\mathbf{V}$ are reprojected into this new $\mathbf{U}$-space, only the part in the intersection of the old and the new space can be represented and the orthogonal part is lost. This would lead to an accumulation of errors during the merge steps. To avoid this accumulation, we first compute only $\mathbf{U}$. Instead of calculating and merging the individual matrices $\mathbf{V}_i$, we calculate $\mathbf{V}$ in an additional step by projecting the columns of $\mathbf{M}$ on the subspace spanned by $\mathbf{U}$. For the same reason, we also recalculate the singular values in the final projection step, even though we have to update $\mathbf{S}$ during the calculation of $\mathbf{U}$ since it is needed to perform the merge steps. In addition to the improved accuracy, this reduces the complexity of the implementation as well as the memory requirements. The drawback of this approach is that we must spend additional IO time for this final step since we have to load the full matrix again. Thus, for applications where speed is more important than precision, it might be advantageous to instead update $\mathbf{V}$ together with $\mathbf{U}$ during the merge steps.

In the following sections, we will give a short overview of the EM-PCA algorithm we used for the factorization of the sub-problems and the technique we used to merge the individual factorizations to obtain the full SVD.

3.1.1 EM-PCA

Instead of calculating an SVD of $\mathbf{M}_i$ directly, we approximate it by first using the EM-PCA algorithm introduced in [Row97] to find the subspace spanned by the first k principal components of $\mathbf{M}_i$ and then performing the SVD on the projection of $\mathbf{M}_i$ into this subspace. This way, we only have to compute the SVD for a $(k+1) \times s_i$ matrix, containing the data vectors projected into the subspace spanned by the first k principal components and the mean direction of $\mathbf{M}$. This factorization can be done very fast for small k.

The EM-PCA algorithm is an expectation-maximization algorithm which allows to find the subspace spanned by the first k principal components, without explicitly calculating all principal components. After initializing the $m \times k$ output matrix $\mathbf{C}$ with random values, it iterates between the following two steps:

E-step:

$$\mathbf{X} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\overline{\mathbf{M}}_i$$

M-step:

$$\mathbf{C}^{new} = \overline{\mathbf{M}}_i\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$$

Here, $\overline{\mathbf{M}}_i$ is a $m \times s_i$ input matrix with zero mean and $\mathbf{X}$ is a $k \times s_i$ matrix of unknown states. After the iteration has completed, the columns of $\mathbf{C}$ span the principal subspace. As analyzed in [Row97], this EM algorithm always converges and the number n_i of necessary iterations is rather small and independent from the size of $\overline{\mathbf{M}}_i$. In our experience, 15 iterations were sufficient for good compression results.

Thus, only matrix multiplications and inversions are needed for the calculation of the PCA. In practice, the runtime is dominated by the two multiplications with the $m \times s_i$ matrix $\overline{\mathbf{M}}_i$, since these require $O(kms_i)$ operations. The algorithm is therefore practically linear in the size of the input data. This approach is thus well-suited for GPU implementations, as the matrix multiplications are easily parallelizable, especially for very large matrices like $\overline{\mathbf{M}}_i$. Matrix inversions have a runtime which is cubic in the matrix dimension and are furthermore not easily parallelizable. However, the contribution of the two inversions in this algorithm to the total runtime is negligible for small k, because the matrices $\mathbf{C}^T\mathbf{C}$ and $\mathbf{X}\mathbf{X}^T$ are both only of size $k \times k$.

When using EM-PCA to calculate the principle subspace for the matrices $\mathbf{M}_i$, it is important to keep in mind that, even though the mean has been subtracted from the full matrix $\mathbf{M}$, the individual block matrices can have non-zero mean and are therefore not directly suited as input data for the EM-PCA. We thus calculate the mean $\mathbf{m}_i$ for each block matrix independently and subtract it from $\mathbf{M}_i$ to obtain the matrix $\overline{\mathbf{M}}_i$ which is then used for the calculation of the subspace. Afterwards, we add the mean vector as an additional column to the matrix $\mathbf{C}$, obtaining the matrix $\mathbf{C}_m$. This is necessary, since the component of the mean vector, which is orthogonal to the determined subspace would otherwise be lost when projecting the data points into the space spanned by $\mathbf{C}$ and thus neglected during the following SVD.

The actual SVD calculation is then performed on a projection of $\mathbf{M}_i$ into the subspace spanned by $\mathbf{C}_m$. For this, $\mathbf{C}_m$ is first orthogonalized, obtaining the matrix $\mathbf{C}_o$. The projection can then be calculated as $\mathbf{P} = \mathbf{C}_o^T\mathbf{M_i}$. Since the columns of $\mathbf{P}$ contain only $k+1$ entries, the SVD $\mathbf{U}_P\mathbf{S}_P\mathbf{V}_P^T = \mathbf{P}$ can be calculated efficiently. To obtain the final result, we project the matrix $\mathbf{U}_P$ back into the original space by setting $U_i = \mathbf{C}_o\mathbf{U}_P$ and $\mathbf{S}_i = \mathbf{S}_P$.

3.1.2 SVD Merging

Let $\mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T$ and $\mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T$ be two Singular Value Decompositions which have been truncated after c_1 and c_2 singular values respectively and let $\widetilde{\mathbf{M}}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T$ and $\widetilde{\mathbf{M}}_2 = \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T$ be the matrices reconstructed from these decompositions. We have to find the Singular Value Decomposition $\mathbf{USV}^T$ of the composed matrix $\widetilde{\mathbf{M}} = [\widetilde{\mathbf{M}}_1\ \widetilde{\mathbf{M}}_2]$. For this, we generalized and adapted the update step of the incremental SVD [Bra02] to the merging of the two SVDs. Similar eigenspace merging techniques like the one in [HMM00] could however be used instead.

The merging of the two SVDs is performed by first constructing an orthogonal space for the subspace spanned by both $\mathbf{U}_1$ and $\mathbf{U}_2$ and then performing the SVD within this subspace.

For this, the Singular Value Decomposition of $\widetilde{\mathbf{M}}_2$ is split into the part which lies within the subspace spanned by $\mathbf{U}_1$ and the part orthogonal to this subspace. First, $\mathbf{U}_2$ is projected into this space, resulting in $\mathbf{L} = \mathbf{U}_1^T\mathbf{U}_2$. Then, the orthogonal part is computed as $\mathbf{H} = \mathbf{U}_2 - \mathbf{U}_1\mathbf{L}$. In the next step, an orthogonal basis $\mathbf{Q}$ for the space spanned by $\mathbf{H}$ is determined. Now, $\mathbf{H}$ is projected into this space by setting $\mathbf{R} = \mathbf{Q}^T\mathbf{H}$. $\mathbf{U}' = [\mathbf{U}_1\ \mathbf{Q}]$ is thus an orthogonal basis for the subspace spanned by both $\mathbf{U}_1$ and $\mathbf{U}_2$.

We can now consider the following identity:

$$
\begin{aligned}
\widetilde{\mathbf{M}} &= \mathbf{U}'\mathbf{U}'^T\widetilde{\mathbf{M}} && (3.1)\\
&= \begin{bmatrix}\mathbf{U_1} & \mathbf{Q}\end{bmatrix} \begin{bmatrix}\mathbf{U_1}^T \\ \mathbf{Q}^T\end{bmatrix} \begin{bmatrix}\mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T & \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T\end{bmatrix} && (3.2)\\
&= \begin{bmatrix}\mathbf{U_1} & \mathbf{Q}\end{bmatrix} \begin{bmatrix}\mathbf{U}_1^T\mathbf{U}_1\mathbf{S}_1 & \mathbf{U}_1^T\mathbf{U}_2\mathbf{S}_2 \\ \mathbf{Q}^T\mathbf{U}_1\mathbf{S}_1 & \mathbf{Q}^T\mathbf{U}_2\mathbf{S}_2^T\end{bmatrix} \begin{bmatrix}\mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2\end{bmatrix}^T && (3.3)\\
&= \underbrace{\begin{bmatrix}\mathbf{U_1} & \mathbf{Q}\end{bmatrix}}_{\mathbf{U}'} \underbrace{\begin{bmatrix}\mathbf{S}_1 & \mathbf{LS}_2 \\ 0 & \mathbf{RS}_2\end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix}\mathbf{V}_1 & 0 \\ 0 & \mathbf{V}_2\end{bmatrix}^T}_{\mathbf{V}'^T} && (3.4)
\end{aligned}
$$

In this identity, (3.4) is already of similar structure as an SVD of $\widetilde{\mathbf{M}}$ because $\mathbf{U}'$ and $\mathbf{V}'$ are orthogonal matrices. However, $\mathbf{C}$ is not a diagonal matrix. Therefore, we have to perform a Singular Value Decomposition $\mathbf{U}''\mathbf{S}''\mathbf{V}''^T = \mathbf{C}$ which is computationally not very expensive in our case since $\mathbf{C}$ is a $(c_1 + c_2) \times (c_1 + c_2)$ matrix and $c_1, c_2 \ll m, n$.

Since $\mathbf{U}', \mathbf{U}'', \mathbf{V}', \mathbf{V}''$ are orthogonal matrices and $\mathbf{S}'$ is a diagonal matrix, setting

$$
\mathbf{U} = \mathbf{U}'\mathbf{U}'' \qquad \mathbf{S} = \mathbf{S}' \qquad \mathbf{V} = \mathbf{V}'\mathbf{V}'' \tag{3.5}
$$

results in the Singular Value Decomposition of $\widetilde{\mathbf{M}}$:

$$
\begin{aligned}
\widetilde{\mathbf{M}} &= \mathbf{U}'\mathbf{C}\mathbf{V}'^T = \mathbf{U}'\mathbf{U}''\mathbf{S}\mathbf{V}''^T\mathbf{V}'^T && (3.6)\\
&= \mathbf{U}\mathbf{S}\mathbf{V}^T && (3.7)
\end{aligned}
$$

The calculation of the matrix $\mathbf{U}$ is thus possible from only $\mathbf{U}_1, \mathbf{S}_1$ and $\mathbf{U}_2, \mathbf{S}_2$. Therefore, we do not need to calculate and update $\mathbf{V}$ during the calculation of $\mathbf{U}$, but can neglect it first and then obtain $\mathbf{V}$ afterwards by projecting the data on the basis $\mathbf{U}$.

After the merge step, the new SVD has $c_1 + c_2$ singular values and vectors. Since we always merge a matrix with $k + 1$ columns to the already computed result this would grow by $k + 1$ in each merge step. Therefore, it is necessary to truncate after each merge step. To reduce the error introduced by this truncation, we simply keep $2k$ singular values and vectors instead of only k during the calculation. In our experiments, this was sufficient for good BTF compression results. However, an approach to further reduce the error would be to instead use a threshold on the singular values to decide where to truncate the decomposition, as done in [Bra02]. We avoid this, because the decomposition would continue to grow during the merge operations, though to a lesser degree.

```
Function: BlockSVD($\mathbf{M}$, $\mathbf{m}$, $n_i$, $k$)
    // Subtract mean
    $\mathbf{m}_l$ := mean ($\mathbf{M}$)
    $\mathbf{M}$ := add-to-columns ($\mathbf{M}$, $-\mathbf{m}_l$)
    // EM-PCA
    $\mathbf{C}$ := [$k$ random unit column vectors]
    foreach $i \in \{1, \ldots, n_i\}$ do
        $\mathbf{X} := (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{M}$
        $\mathbf{C} := \mathbf{M}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$
    end
    // Perform SVD in subspace
    $\mathbf{m}_d := \mathbf{m}_l - \mathbf{m}$
    $\mathbf{C}_o$ := orthogonalize ([$\mathbf{C}$ $\mathbf{m}_d$])
    $\mathbf{M}$ := add-to-columns ($\mathbf{M}$, $\mathbf{m}_d$)
    $\mathbf{U}, \mathbf{S}, \mathbf{V}$ = svd ($\mathbf{C}_o^T\mathbf{M}$)
    return $\mathbf{C}_o\mathbf{U}, \mathbf{S}$

Function: MergeBlocks($\mathbf{U}_1, \mathbf{S}_1, \mathbf{U}_2, \mathbf{S}_2, k$)
    // Find orthogonal subspace for $\mathbf{U}_2$
    $\mathbf{L} := \mathbf{U}_1^T\mathbf{U}_2$
    $\mathbf{H} := \mathbf{U}_2 - \mathbf{U}_1\mathbf{L}$
    $\mathbf{Q}$ := orthogonalize ($\mathbf{H}$)
    $\mathbf{R} := \mathbf{Q}^T\mathbf{H}$
    // Merge the SVDs
    $\mathbf{C} := \begin{bmatrix} \mathbf{S}_1 & \mathbf{L}\mathbf{S}_2 \\ 0 & \mathbf{R}\mathbf{S}_2 \end{bmatrix}$
    $\mathbf{U}'', \mathbf{S}'', \mathbf{V}''$ := svd ( $\mathbf{C}$ )
    $\mathbf{U} := [\mathbf{U}_1\ \mathbf{Q}]\mathbf{U}''$
    return $\mathbf{U}_{1:m,1:2k}$, $\mathbf{S}''_{1:2k,1:2k}$
```

```
Function: BlockwisePCA($\mathbf{M}, k, n_i$)
    $\mathbf{m}$ := mean ($\mathbf{M}$);
    // Calculate $\mathbf{U}$
    foreach $i \in \{1, \ldots, N\}$ do
        $\mathbf{M}_i$ = LoadBlock ($i$);
        $\mathbf{U}', \mathbf{S}'$ = BlockSVD ($\mathbf{M}_i$, $\mathbf{m}$, $n_i$, $k$);
        if $i = 1$ then
            $\mathbf{U}$ := $\mathbf{U}'$;
            $\mathbf{S}$ := $\mathbf{S}'$;
        else
            $\mathbf{U}, \mathbf{S}$ = MergeBlocks ($\mathbf{U}, \mathbf{S}, \mathbf{U}', \mathbf{S}', k$);
        end
    end
    // Project $\mathbf{M}$ into subspace
    // to get $\mathbf{S}$ and $\mathbf{V}$
    $\mathbf{V}$ := $(\mathbf{U}^T\mathbf{M})^T$;
    $\mathbf{S}$ := Diag(ColumnNorms($\mathbf{V}$));
    $\mathbf{V}$ := $\mathbf{V}\mathbf{S}^{-1}$;
    return $\mathbf{U}, \mathbf{S}, \mathbf{V}, \mathbf{m}$;
```

Pseudocode 3.1: *Our factorization method*

3.2 Implementation

As can be seen in Pseudocode 3.1, our algorithm is based on just a few basic matrix operations, most of which can be easily parallelized on the GPU. For this, we use the NVIDIA CUBLAS™library (for more information see [NVI08]), which allows to perform many basic linear algebra operations efficiently on the GPU. For our algorithm, the most important of these operations are the matrix multiplications, for which we use the `cublasSgemm` function. Similarly, we also calculate the column means, using a matrix-vector product, and the mean subtraction, using the rank-1 matrix update function `cublasSger`, with the CUBLAS™library, though none of these operations has a very high contribution to the total runtime. We also accelerated the matrix orthogonalization on the GPU.

Thus, the only parts of the algorithm not accelerated on the GPU are operations on the tiny matrices of size $k \times k$ and $(3k + 1) \times (3k + 1)$ respectively. For BTF compression, k is chosen quite small and therefore these operations are mostly irrelevant for the total runtime. Thus, a GPU implementation of these parts is not necessary, reducing the implementation complexity considerably, since CPU implementations of these algorithms are readily available, for example in the LAPACK library [ABD*90].

The size of the individual matrices $\mathbf{M}_i$ should be chosen as large as the available GPU memory allows, because each merge step introduces a certain error and we should strive to minimize the number of merge steps. Since the matrix is processed blockwise, the runtime can easily be further improved by performing the IO asynchronously to the actual calculation. For this, we use an additional thread which preloads the next block of the matrix while the current one is processed.

Using this technique, we can on the one hand directly perform a factorization of the full BTF data matrices. However, we also applied our algorithm to the LocalPCA BTF compression algorithm of Müller et al. [MMK03]. This algorithm first performs a clustering step in the spatial dimension and then applies the PCA to each cluster independently. The advantage of this method is, that a very low number of components is sufficient to faithfully reproduce the data in the individual clusters. Therefore, the decompression speed is considerably higher than for techniques based on a Full Matrix Factorization. We use our method to accelerate both the clustering and the final projection steps of this algorithm. Furthermore, we perform the clustering within the projection of $\mathbf{M}$ into the $\mathbf{U}$-space by first performing a factorization of the full BTF matrix, as this further increases the performance by reducing the time needed for the error calculations.

3.3 Results

To show the advantages of our parallelized factorization method in the context of BTF compression, we applied it to full BTF matrices of several materials. For the reconstruction, we used the first 120 principal components. We compare our runtimes and reconstruction errors to an out-of-core implementation of the EM-PCA algorithm performed on a single CPU core. For this, we simply used the average ABRDF *Root Mean Square Error* (RMSE):

$$E = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{\left\|\mathbf{m}_{:,i} - \tilde{\mathbf{m}}_{:,i}\right\|^2}{m}} \tag{3.8}$$

Here, $\mathbf{m}_{:,i}$ is the i-th column of the BTF matrix and $\tilde{\mathbf{m}}_{:,i}$ is the i-th column of the reconstructed matrix. Table 3.1 shows timings and the achieved reconstruction errors. All timings were measured on a computer with a Q6600 CPU, 8GB of main memory and a GeForce 8800 GTX GPU with 768MB GPU memory. Additionally we compared our extension of the LocalPCA compression method

Material	Resolution	Size	#Blocks	Block-PCA		EM-PCA		Speedup	Rel. error
		[GB]		Time[s]	$\overline{\text{RMSE}}$	Time[s]	$\overline{\text{RMSE}}$		increase
Leather1	256^2x95^2	6.61	12	317	0.0236689	11659	0.0236529	36.78	0.068%
Leather1	512^2x95^2	26.4	48	2398	0.078524	47019	0.0785006	19.61	0.030%
Leather2	128^2x151^2	4.17	11	280	0.0113223	7711	0.0113162	27.54	0.054%
Leather3	256^2x81^2	4.81	9	261	0.0143584	8109	0.0143554	31.07	0.021%
Pulli	256^2x81^2	4.81	9	266	0.0282213	8129	0.0282085	30.56	0.045%
Fabric	256^2x81^2	4.81	9	223	0.0060206	8146	0.0060141	36.53	0.108%
				LPCA with Block-PCA		LPCA with EM-PCA			
				Time[s]	$\overline{\text{RMSE}}$	Time[s]	$\overline{\text{RMSE}}$		
Leather1	256^2x95^2	6.61	12	858	0.0368971	19104	0.0370494	22.27	-0.41%
Pulli	256^2x81^2	4.81	9	546	0.0374273	13573	0.0373398	24.86	0.23%

Table 3.1: *Upper part: Comparison of runtime and reconstruction error between our method and a non-parallel EM-PCA with* $k = 120$*. Lower part: Comparison between our modified LocalPCA method and LocalPCA based on EM-PCA.*

of Müller et al. [MMK03] to a CPU implementation of the LocalPCA algorithm using the full data matrix and the EM-PCA method.

Our method achieves roughly a speed-up by a factor between 20 and 35. At the same time the increase in reconstruction error does not exceed 0.11% for the Full Matrix Factorization. The relative error is more unstable for the LocalPCA, but this is mainly due to the jitter of the clustering step. In Figure 3.4 we compare renderings of the materials compressed with both full matrix techniques and in Figure 3.5 we make the same comparison for the two LocalPCA implementations. There is no visible difference between the version compressed with our techniques and the serial CPU implementations.

It should be noted, that the runtimes in Table 3.1 include the necessary IO times, which dominate the runtime of our algorithm for large datasets since caching by the operating system is no longer possible for them. For example, for the large dataset `Leather1` with 26 GB matrix size one complete IO pass required about 700 seconds. Thus, already more than half of the 2398 seconds runtime is spent on read operations during the mean calculation and the final computation to determine $\mathbf{V}$. The block factorizations have small additional IO cost, because we perform the IO asynchronously. Only the IO for the first block is performed synchronously, on the 26 GB matrix it takes about 40 seconds. Except for the second block, where still 10 additional IO seconds are needed, the IO for all further blocks is completely asynchronous and only one second is required after the calculation step to fetch the data for the next block.

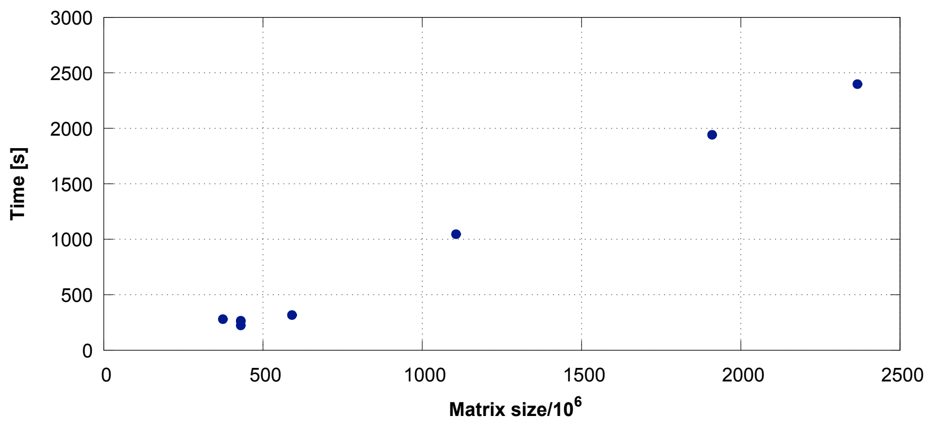


Figure 3.2: *Runtime of our algorithm for different matrix sizes and* $k = 120$ *components. Runtime for the smaller matrices is heavily influenced by the caching behavior of the operating system.*

We investigated the runtime of our algorithm with increasing matrix size $m \times n$ and increasing number of components k. As it can be seen in Figure 3.2, the runtime is linear in $m \times n$ as expected. Figure 3.3 shows the runtime in dependence on k. We performed cubic regression to determine the contribution of the $O\left(k^3\right)$ operations to the total runtime. The coefficients for the quadratic and cubic part are very low compared to the linear part. This shows that the matrix multiplications with the large matrices dominate the total runtime of the algorithm as it was stated in Section 3.1.

3.4 Conclusion

We presented a method which accelerates the factorization of large data matrices, as they can be found in BTF compression, by exploiting the massive parallel computing power offered by modern GPUs.

This is achieved by first subdividing the input matrix into blocks, which are factorized independently using the EM-PCA algorithm, and then merging the resulting eigenspaces to obtain the final result. This technique allows to process matrices of nearly arbitrary size. We evaluated our technique by applying it to the compression of full BTF matrices. Here, it achieves speed-ups between 20-35, without increasing the reconstruction error by more than 0.11%, when compared to an out-of-core CPU implementation of the EM-PCA algorithm. This considerable

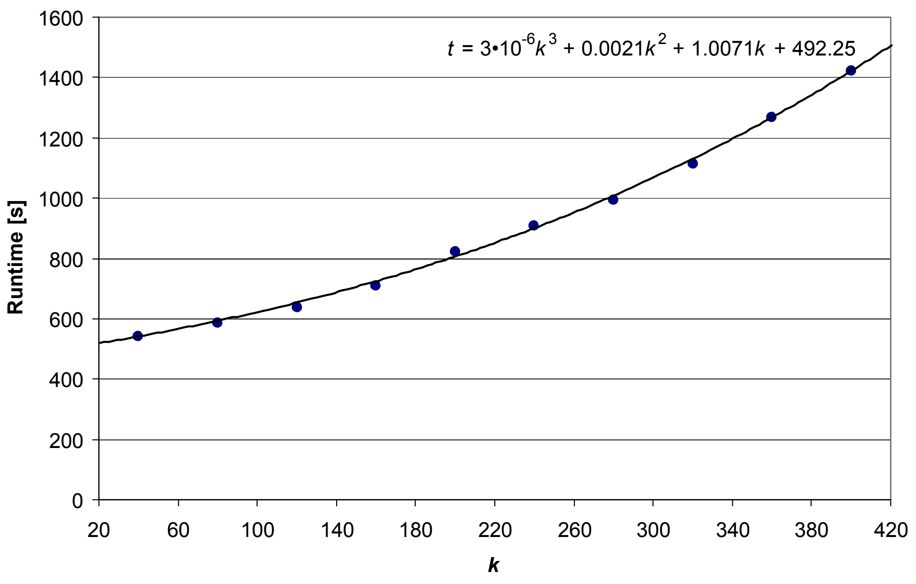


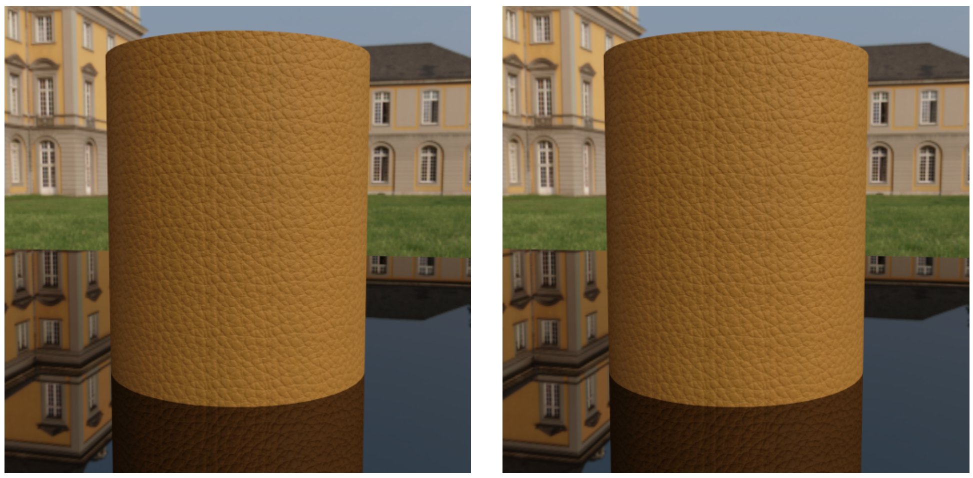
Figure 3.3: *Runtime of our algorithm for increasing number of components k.*

acceleration enables the practical processing of BTF datasets with high angular and spatial resolution.

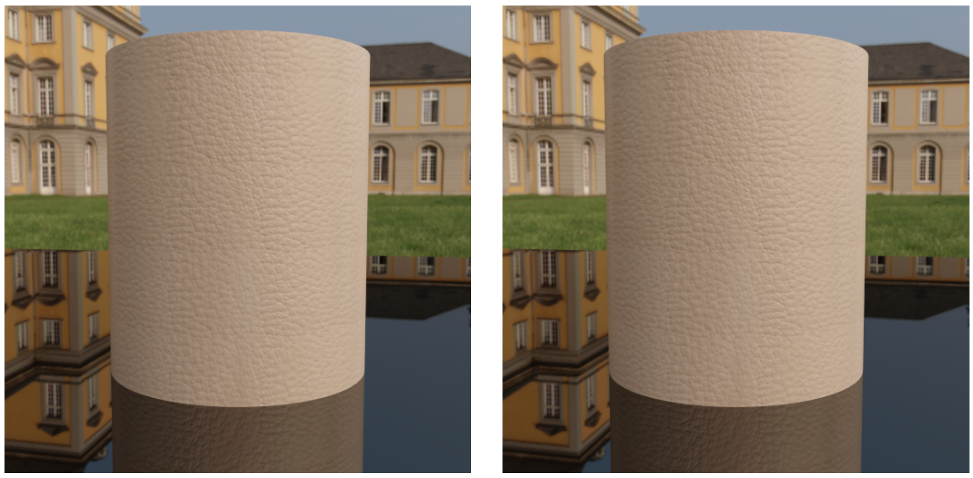
The computation time for each block is not dependent on the contained data and the individual blocks can be processed independently and in arbitrary order. Therefore, we think it will be quite easy to parallelize the algorithm to multiple GPUs, because the load balancing between the execution threads should not be too complex.

The algorithm presented in this chapter has been used in our working group for the compression of most BTFs we had to process since 2009 and has worked reliably for even larger BTFs up to $151 \times 151 \times 2048 \times 2048$. Though the technique has originally been developed for the compression of BTFs, the factorization of big matrices is a problem that arises in a large number of applications and thus the algorithm can also be applied in these cases. For example, we used it to perform a dimensionality reduction prior to the K-SVD computation described in Chapter 4 and prior to the nearest neighbor search for the texture synthesis algorithm used in Chapter 9 and Chapter 10. Therefore, we made the source code available for other researchers at `http://cg.cs.uni-bonn.de/en/publications/additional-material/blockpca-source-code/`.

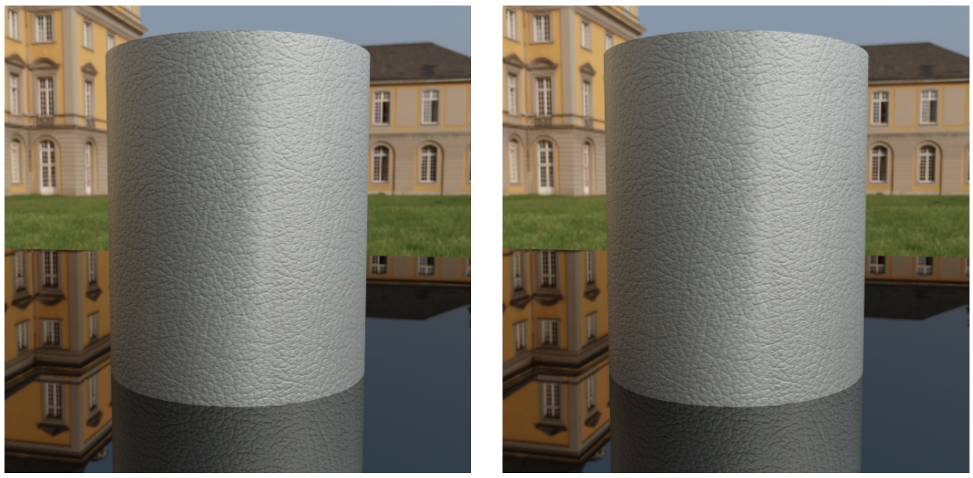
Leather1



Leather2



Leather3



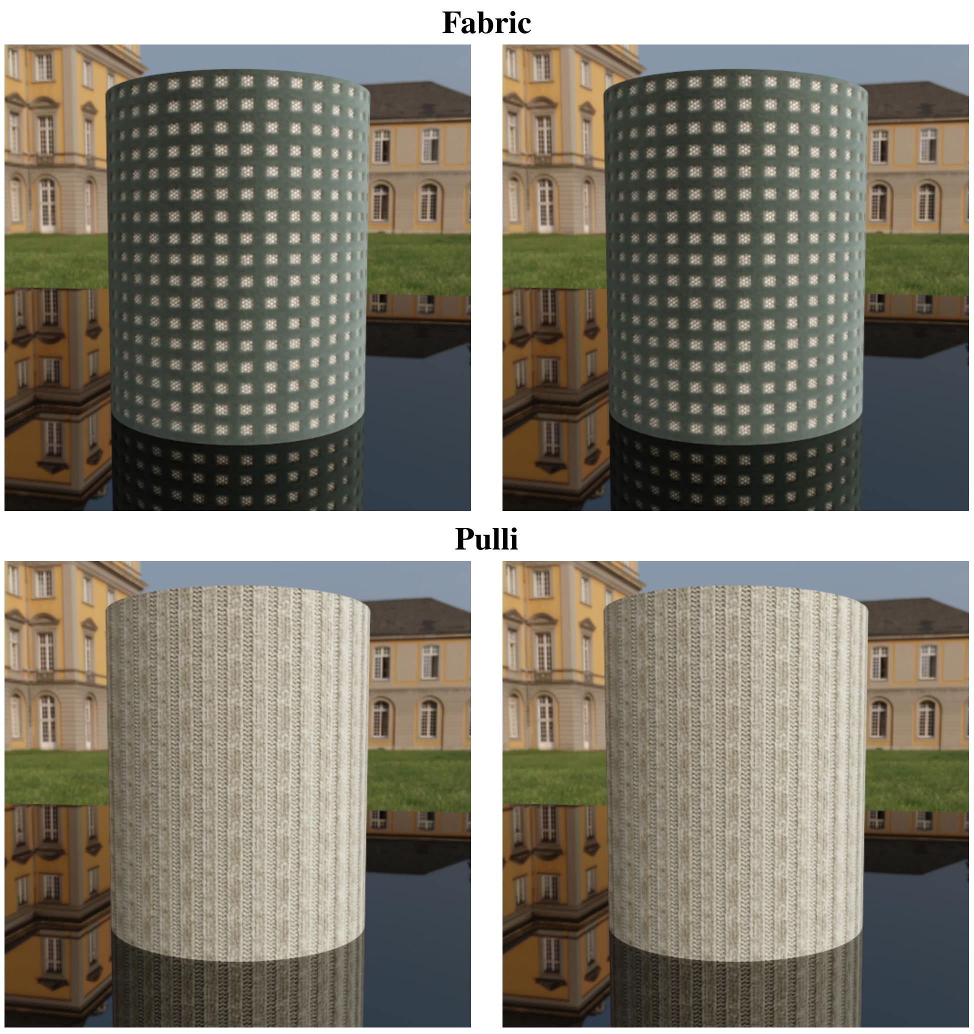


Figure 3.4: *Visual comparison between our factorization method (left row) and out-of-core EM-PCA (right row)*

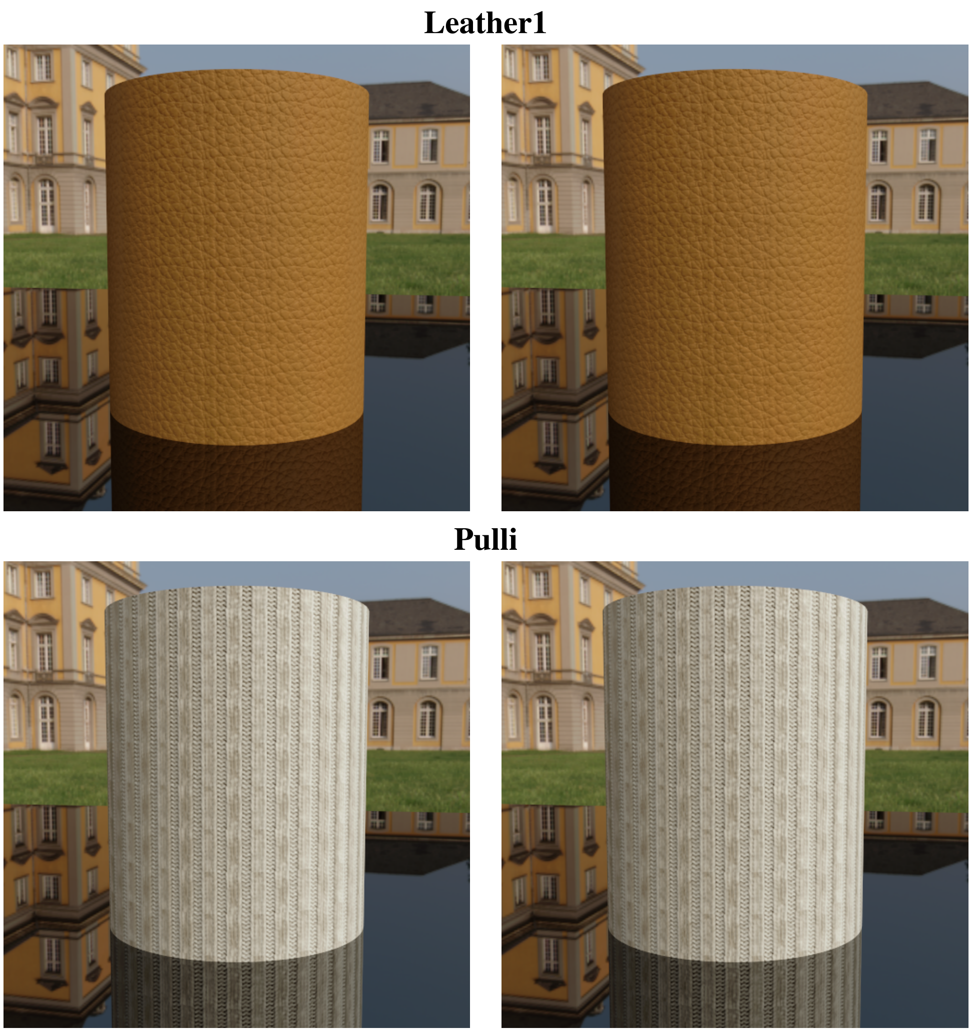


Figure 3.5: *Visual comparison between our modification of the LocalPCA method (left) and the original algorithm (right).*

CHAPTER 4

BTF COMPRESSION VIA SPARSE TENSOR DECOMPOSITION

To be of practical use, BTF compression techniques (see Section 2.2.2 for an overview) have to fulfill two important requirements. On the one hand, they must provide high compression ratios at an acceptable degradation in rendering quality. On the other hand, since during rendering usually every rendered pixel corresponds to a different texture position, and a different light and view direction, efficient random access to the data is also necessary.

Many BTF compression techniques are based on matrix factorizations (e.g. [KM03, LHZ*04, SSK03]). However, these techniques can only exploit correlations between the columns of the matrix and thus do not take advantage of the higher-dimensional structure of BTF datasets. To overcome this limitation, several approaches based on tensor decompositions have been proposed (e.g. [FKIS02, VT04, WWS*05, WXC*08]). However, as these techniques are based either on Tucker or PARAFAC decompositions, random access into the tensor is quite expensive because the reconstruction of individual entries requires evaluating long sums with many terms.

In [MMK03], the use of local PCA was proposed. This approach improves the decompression performance by clustering the ABRDFs and then performing a PCA on each cluster independently. Since the samples in each cluster are represented in a different basis, only a smaller number of coefficients is needed. However, for each cluster a set of basis vectors has to be stored which is not further compressed, reducing the total compression ratio.

In this chapter, we propose a sparse BTF tensor decomposition which combines the ability of tensor based techniques to exploit correlations in several dimensions with a sparse representation that, similar to local PCA, reduces the number of terms that have to be evaluated during decompression. This way, we achieve at the same time very high compression ratios and a decompression performance which is at high compression ratios superior to the factorization of a matrix containing the whole BTF.

For this, we use the K-SVD algorithm from Aharon et al. [AEB06] to split the BTF tensor into a dictionary and two sparse tensors. By splitting the tensor along two different modes, we can utilize correlations both in the spatial dimension and between different view directions. Since the dictionary itself is very compact and the two sparse tensors can also be stored efficiently, this approach achieves very high compression ratios. In our experiments, we achieved compression ratios that were better by a factor of three to four than those provided by current state-of-the art methods.

This chapter corresponds to the paper "BTF Compression via Sparse Tensor Decomposition" by Roland Ruiters and Reinhard Klein, published in *Computer Graphics Forum (Proc. of EGSR)*, 28(4):1181–1188, July 2009.

4.1 Theory

The use of sparse representations of signals has received considerable interest in recent years and has been used in a wide range of applications like image denoising, restoration, classification and compression. These techniques represent a set of signals, given as the columns of the matrix $\mathbf{Y}$, as a sparse combination of the columns of a *dictionary* $\mathbf{D}$: $\mathbf{Y} \approx \mathbf{DX}$. For a fixed dictionary $\mathbf{D}$, a sparse representation $\mathbf{X}$ with at most k entries in each column can be found by solving the following problem:

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{DX}\|^2 \quad \text{subject to} \quad \forall i : \|\mathbf{x}_{:,i}\|_0 \leq k. \tag{4.1}$$

Here, $\|\mathbf{x}_{:,\mathbf{i}}\|_0$ is the number of non-zero entries in column i of $\mathbf{X}$. The exact solution to this problem is NP-hard, but several *pursuit algorithms* exist which efficiently approximate the solution, among them are *Orthogonal Matching Pursuit* (OMP) [PRK93], *Basis Pursuit* (BP) [CDS98] and the *Focal Undetermined System Solver* (FOCUSS) [GR97]. Though for many applications predefined dictionaries, like wavelets or an overcomplete DCT dictionary, are used, it is also possible to learn a dictionary from the samples. Here, the user only specifies the number D of columns in the dictionary matrix and the sparsity k. Then, $\mathbf{D}$ and $\mathbf{X}$ in Equation 4.1 are minimized together. Several algorithms to find approximate solutions to this problem exist (e.g. [GR97], [LS00], [EAH00], [KDMR*03]). Aharon et al. [AEB06] introduced the *K-SVD* and reported superior results compared to the aforementioned

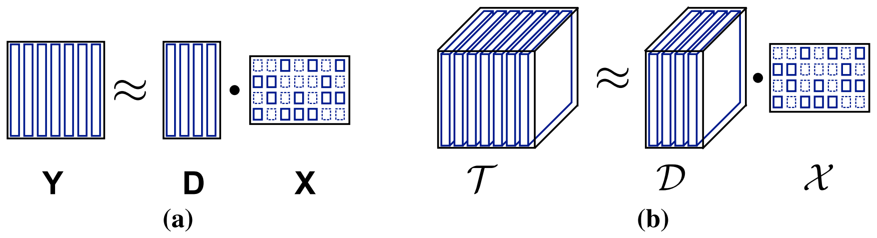


Figure 4.1: *(a) The Matrix* $\mathbf{Y}$ *is approximated as a sparse combination of the columns of* $\mathbf{D}$. *(b) The mode-3 tensor* $\mathcal{T}$ *is approximated as a sparse combination of mode-2 subtensors of* $\mathcal{D}$.

algorithms. It is a generalization of k-means clustering which iterates between a sparse coding step, in which $\mathbf{X}$ is optimized using one of the pursuit algorithms, and a codebook update step, in which $\mathbf{D}$ is optimized. This is done for each codebook entry independently by performing an SVD on a residual matrix computed without the entry itself and for only those samples which are represented by this codebook entry.

4.1.1 Sparse Tensor Decomposition

A decomposition $\mathbf{Y} \approx \mathbf{DX}$ of a matrix $\mathbf{Y}$ into a dictionary $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ (see Figure 4.1a), can be regarded as approximating each of the columns of $\mathbf{Y}$ as a linear combination of at most k *atoms* from a dictionary $\mathbf{D}$. Here, $\mathbf{Y}$ is considered as a set of vectors and $\mathbf{D}$ is a dictionary in which each column represents one atom. $\mathbf{X}$ then consists of sparse vectors, each of which contains at most k non-zero entries describing how one column of $\mathbf{Y}$ can be approximated as a combination of the dictionary atoms.

This can easily be generalized to tensors. $\mathbf{Y}$ is actually a mode-2 tensor which we regard as consisting of mode-1 subtensors, arranged in a mode-1 tensor. Similarly, a mode-N tensor $\mathcal{T}$ can be regarded as a collection of mode-M subtensors, which are then arranged in a mode-$(N - M)$ tensor. Each of these mode-M subtensors can then be approximated as a linear combination of at most k dictionary atoms. Since each of these dictionary atoms is itself a mode-M tensor, the whole dictionary $\mathcal{D}$ is a mode-$(M + 1)$ tensor.

For each of the subtensors from $\mathcal{T}$, a sparse vector is needed to describe which of the dictionary atoms are used to approximate it. This can be represented as a mode-$(N - M + 1)$ tensor $\mathcal{X}$, for which in one of the modes each fiber has at most k non-zero entries. Figure 4.1b shows an illustration of this kind of tensor decomposition for the case of a mode-3 tensor.

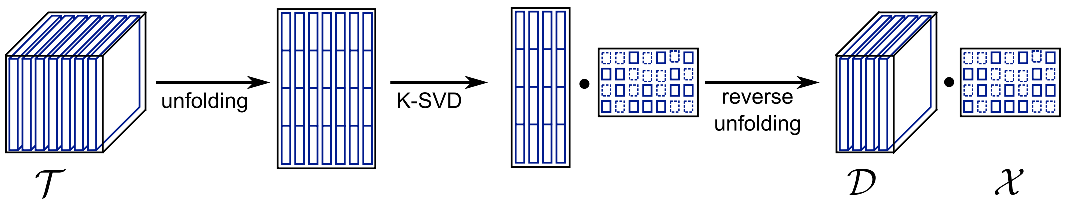


Figure 4.2: *Illustration of the computation of the sparse tensor decomposition for a mode-3 input tensor.*

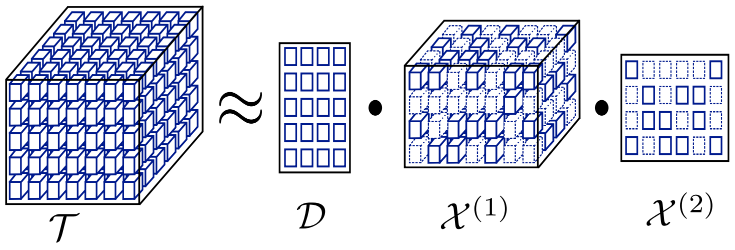


Figure 4.3: *Final Sparse Tensor Decomposition for a mode-3 input tensor.*

We thus approximate a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as

$$\mathcal{T} \approx \mathcal{D}_{i_1 \cdots i_M j} \mathcal{X}_{j i_{M+1} \cdots i_N}.$$

Here, $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M \times D}$ is the dictionary tensor containing D atoms and $\mathcal{X} \in \mathbb{R}^{D \times I_{M+1} \times \cdots \times I_N}$ is the tensor describing how the atoms from $\mathcal{D}$ are combined. $\mathcal{X}$ is a sparse tensor in which each mode-1 fiber contains at most k non-zero entries.

To obtain such a decomposition for a given tensor $\mathcal{T}$, we first unfold the tensor in such a way, that the M modes corresponding to the dictionary entries are represented in one column of the resulting matrix. Then, we use the K-SVD [AEB06] algorithm to find the decomposition

$$\mathcal{T}_{(I_1 I_2 \ldots I_M \times I_{M+1} \ldots I_N)} \approx \mathbf{DX}$$

of this unfolded tensor. By assigning $\mathcal{D}_{(I_1 I_2 \ldots I_M \times D)} = \mathbf{D}$ and $\mathcal{X}_{(D \times I_{M+1} \ldots I_N)} = \mathbf{X}$, the unfolding is then reversed to obtain the actual tensor decomposition. See Figure 4.2 for an illustration.

This decomposition so far only utilizes correlations between the individual mode-M subtensors but not correlations along other modes within each of the subtensors. In contrast to matrix decompositions via SVD, the dictionary $\mathbf{D}$ is not orthogonal. The atoms can therefore still exhibit correlations to each other. This can be used to further improve the compression by decomposing the dictionary $\mathcal{D}$ again, this time using a different partitioning of the tensor modes. See Figure 4.3 for an illustration. When repeated for all modes, this finally results in one dense mode-2 dictionary tensor $\mathcal{D}$, a set of sparse mode-3 tensors $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(N-1)}$ and one sparse mode-2 tensor $\mathcal{X}^{(N)}$, decomposing $\mathcal{T}$ in the following way:

$$\mathcal{T} \approx \mathcal{D}_{i_1 j_1} \mathcal{X}^{(1)}_{j_1 i_2 j_2} \mathcal{X}^{(2)}_{j_2 i_2 j_3} \cdots \mathcal{X}^{(N)}_{j_N i_N}.$$

4.1.2 BTF Compression

Prior to the BTF compression, we subtract the mean ABRDF from the dataset and store it together with the decomposition. This is often done for BTF compression because many materials have a characteristic mean ABRDF which contributes strongly to nearly all samples. This way, we avoid storing coefficients at each position for this ABRDF.

We then represent the seven-dimensional BTF dataset as a tensor $\mathcal{B} \in \mathbb{R}^{C \times L \times V \times P}$, in which the $C = 3$ color channels, the L light directions, the V view directions and the P different spatial positions are each represented in one mode. On this tensor, we then perform a decomposition

$$\mathcal{B} \approx \mathcal{L}_{clj} \mathcal{V}_{jvk} \mathcal{P}_{kp}$$

to obtain a dictionary $\mathcal{L} \in \mathbb{R}^{C \times L \times D_1}$ whose atoms represent the color in dependence on the light direction, a sparse tensor $\mathcal{V} \in \mathbb{R}^{D_1 \times V \times D_2}$ which gives for each view direction a combination of dictionary atoms and finally a sparse tensor $\mathcal{P} \in \mathbb{R}^{D_2 \times P}$ which describes the spatial distribution of the ABRDFs.

We decided to perform no further subdivisions in more modes. Splitting the dictionary $\mathcal{L}$ into its two modes is not necessary because $\mathcal{L}$ does not require much storage and thus further compression would be of little use. The light or view directions cannot be represented in more modes because they are usually sampled in an irregular pattern which cannot easily be mapped onto two modes without resampling the input data. It is possible to represent the position on the surface in two modes, as it was done e.g. in [WWS*05]. However, this would increase the reconstruction costs further and strong correlations are only to be expected for regular patterns.

4.2 Implementation

4.2.1 Compression

Performing the K-SVD calculations directly on the full BTF dataset would be prohibitively expensive. Furthermore, large datasets would require an out-of-core implementation of the K-SVD, which would be even slower. Therefore, we calculate the K-SVD not on the original dataset but on a projection into a lower-dimensional subspace of the full BTF. For this, we first calculate a truncated SVD $\mathcal{B}_{(C,L,V\times P)} \approx \mathbf{USV}^T$. We keep a rather high number (e.g. 300) of singular values, which on the one hand reduces the size of the dataset sufficiently to allow for further processing but still retains most of the details in the input dataset. The SVD can be calculated out-of-core on the GPU using the technique from Chapter 3, which reduces the time for this preprocessing to a few minutes for small datasets (e.g. 128×128 spatial and 81×81 angular resolution) and allows to preprocess even very large datasets (e.g. $512 \times 512 \times 95 \times 95 \approx 26$GB) within less than an hour. Once this preprocessing has been completed, even large datasets require only a few hundred MB and thus all further processing can easily be performed in-core.

To get the first decomposition $\mathcal{B} \approx \mathcal{D}_{lvk}\mathcal{P}_{kp}$, we apply the K-SVD to the matrix $\mathbf{SV^T}$, resulting in the sparse tensor $\mathcal{P}$ and the dictionary matrix $\mathbf{D}$, which is then projected back into the higher-dimensional space. The corresponding dictionary tensor is thus obtained by assigning $\mathcal{D}_{(C,L,V\times D_2)} = \mathbf{UD}$. The second K-SVD can then directly be performed on the unfolded tensor $\mathcal{D}_{(C,L\times V,D_2)}$ because $\mathcal{D}$ is already sufficiently small to allow for in-core processing. During this second calculation, we perform an additional reweighting of $\mathcal{D}$ in which each atom of the dictionary is multiplied with the norm of the corresponding row of $\mathcal{P}$. This step is performed to take the relative importance of the individual atoms during the second K-SVD calculation into account. Atoms which contribute much to the final result are this way given a higher importance than rarely used ones during the second compression. However, in our experiments the K-SVD usually found very balanced dictionaries, in which most atoms had a similar contribution to the result, and thus we did not observe a big difference by this additional step.

Finding the best possible decomposition of datasets of this size is probably not feasible as even the sparse coding step is already NP-hard. Furthermore, optimizations like the use of a projection into a lower-dimensional subspace and performing the two K-SVD calculations independent from each other, instead of simultaneously optimizing the dictionary and both sparse tensors, are necessary to process these large datasets. However, our results show (see Section 4.3), that the approximations we can find with this approach are sufficient to achieve very good compression ratios.

The compression times range from about four hours for a BTF dataset with 128×128 spatial and 151×151 angular resolution ($k_1 = 7, k_2 = 10$) to about 18 hours for a 14.77 GB dataset ($335 \times 346 \times 151 \times 152$) compressed at a quite high quality ($k_1 = 13, k_2 = 13$). These timings were performed on a computer with a Q6600 CPU and 4GB RAM and include the parameter selection described in the next section. For the K-SVD, we use the MATLAB implementation of the algorithm from [AEB06], which the authors kindly made available on their website `http://www.cs.technion.ac.il/~elad/software/`, using OMP as pursuit algorithm.

4.2.2 Parameter Selection

For our compression technique, in total four parameters have to be chosen. The dictionary sizes D_1 and D_2 for $\mathcal{L}$ and $\mathcal{V}$ and k_1 and k_2, the number of non-zero entries in the mode-1 fibers of $\mathcal{V}$ and $\mathcal{P}$. We currently use a fixed value of 256 for both D_1 and D_2 because this is the largest dictionary size we can use when the indices for the sparse tensors are represented as bytes.

k_1 and k_2 are then chosen in dependence on the desired compression ratio. Here, k_2 primarily influences the quality of the spatial distribution, whereas the quality of the angular approximation primarily depends on k_1. However, because of parallax and shadowing effects, there is no total decoupling between those two parameters. For certain applications, a strategic dimensionality reduction, as suggested in [VT04], can be desirable. However, we will focus on finding a parameter combination, which achieves the best *Root Means Square* (RMS) error between the original tensor $\mathcal{T}$ and the reconstructed one $\mathcal{T}'$, calculated as

$$\sqrt{\frac{1}{CLVP}\sum_{c,l,v,p}(t_{c,l,v,p} - t'_{c,l,v,p})^2}.$$

For a user specified maximum file size, there are several possible combinations of k_1 and k_2 which could be used. To select a combination which results in a small RMS error, we would have to first compute the error for each. However, performing the full K-SVD calculations for each combination is very expensive. Therefore, we apply a heuristic instead. We observed that the final squared error can be approximated very well by adding the squared errors that were obtained during the first K-SVD and the weighted second one. We now assume that dictionaries for different values of k_1 and k_2 are still similar to each other and then avoid performing the K-SVDs for all combinations by only calculating one decomposition with both high k_1 and k_2. From this decomposition, we now estimate the errors that would result during the two K-SVDs independent from each other keeping the dictionaries fixed. With a fixed dictionary, it is still necessary to perform the pursuit calculations for each value of k again. Instead, we simply greedily choose for each column of the sparse matrices the k largest values from the result of the K-SVD and calculate the error for this selection. When matching pursuit is used, this is actually the exact result. For OMP, it is only an approximation, which could however be improved by solving a linear system of equations for each column.

Because of the simplifications we have made, the errors estimated this way are only very rough approximations. More precise estimates could be found by performing for each value of k a few K-SVD iterations with the already found dictionary as initialization. However, we found that the rough estimates are usually also sufficient to find a combination of k_1 and k_2 with a small final error. The decomposition is then finally repeated for these parameters, using the previously calculated dictionaries as initialization to improve the convergence speed.

4.2.3 Rendering

During rendering, we have to reconstruct random samples from the tensor corresponding to a given position on the surface and a view and light direction. For this, we have to evaluate the sum

$$\mathcal{T}_{c,l,v,p} = \sum_{j=1}^{D_2} \mathcal{P}_{j,p} \sum_{i=1}^{D_1} \mathcal{V}_{i,v,j} \mathcal{L}_{c,l,i}. \tag{4.2}$$

This would require $O(D_1 D_2)$ operations. However, since $\mathcal{P}$ is a sparse tensor, the term $\mathcal{P}_{j,p}$ is non-zero for only k_2 entries and only for these entries it is necessary to evaluate the second sum at all. Similarly, since $\mathcal{V}$ is sparse, for this sum only k_1 terms have to be evaluated, resulting in a total of $O(k_1 k_2)$ operations. Thus, by using a sparse decomposition, the reconstruction cost can be considerably reduced compared to classical tensor factorization techniques, which have to evaluate the full sums (though with possibly smaller values for D_1 and D_2).

For most practical applications, it is furthermore necessary to interpolate both in the angular and the spatial direction during reconstruction. We use bilinear filtering for the spatial interpolation, and the angular filtering is performed by calculating a Delaunay triangulation of the samples in the hemisphere and then interpolating

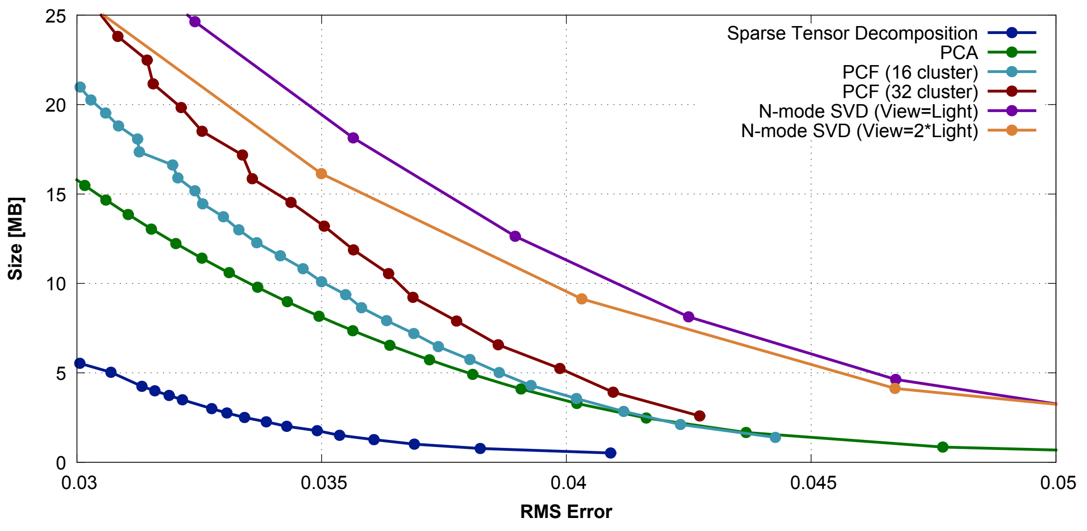


Figure 4.4: *Comparison of our approach (Sparse Tensor Decomposition) to current BTF compression techniques. At the same RMS error, we achieve a compression ratio which is by a factor of 3 to 4 better than the PCA based compression.*

within the resulting triangles. Thus, $4 \times 3 \times 3 = 36$ entries from the tensor have to be decoded to obtain one filtered sample.

For PCA based compression techniques, the interpolation can be performed independently in the angular domain (between 9 samples) and the spatial domain (between 4 samples), considerably improving the performance. However, this is not possible for our technique. For each spatial position at which Equation 4.2 is evaluated, $\mathcal{P}$ has the non-zero entries at different positions. Since the inner loop depends on the index of the non-zero element, it iterates for each position over a different set of dictionary entries, for each of which the angular interpolation has to be performed. Fortunately, adjacent points on the surface often have a similar appearance and therefore also many of the non-zero entries in common. When performing the bilinear interpolation, we first iterate over the four contributing positions p, and compile a list of all indices j for which $\mathcal{P}_{j,p}$ is non-zero. Since we store the indices for the sparse tensors as sorted lists, these can easily be merged to get a list of all contributing indices. Then, the inner loop has to be evaluated for each of these indices only once, eliminating repeated evaluations and considerably improving the reconstruction speed. We observed a speed-up of two in our experiments with this approach.

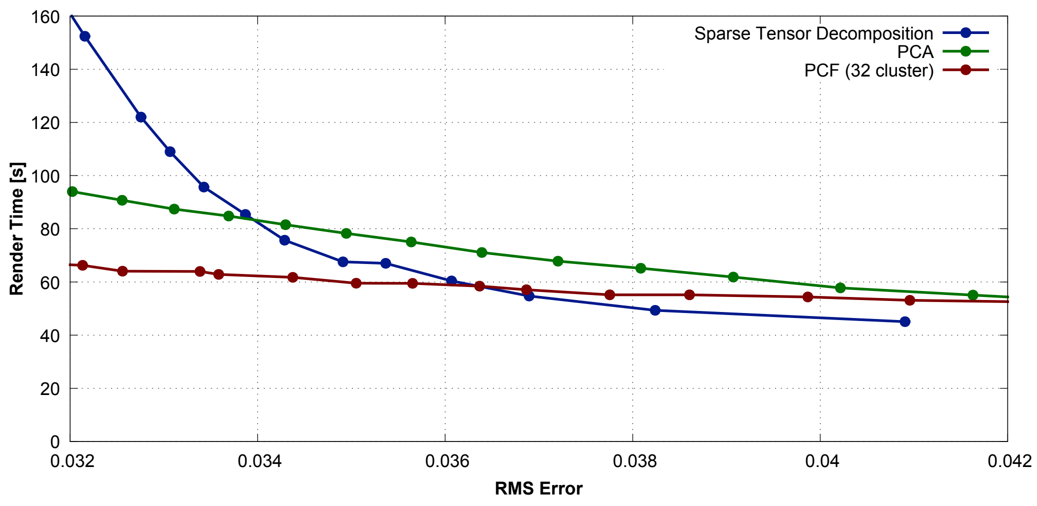


Figure 4.5: *Rendering time in dependence on the BTF quality (On a Q6600 CPU, one core used).*

4.3 Results

To evaluate our technique, we used a BTF of a knitted fabric because it has a distinctive meso-structure creating both shadowing and occlusion effects. It has a spatial resolution of 256×256 and contains all combinations of 81 view and 81 light directions. We store the data as 16 bit floating point values. Without compression the dataset requires about 2.4 GB. On this dataset, we investigate both the compression ratio and the rendering performance. We measure both in dependence on the RMS error.

We compared our approach to the PCA based Full Matrix Factorization [KM03, LHZ*04], the Per Cluster Factorization (PCF) [MMK03] and a tensor factorization similar to the one proposed in [WWS*05]. For all compared approaches, we unfolded the three color channels into the light directions to exploit correlations between the three color channels. Alternatively, it would be possible to first perform a color decoupling and then store the channels independently, using different numbers of coefficients for the different channels as was proposed in [Mül08].

For the Full Matrix Factorization results, we increased the number of components in steps of 5 from 5 to 100. For the PCF compression, the graph in Figure 4.4 shows results for two series, obtained by keeping the number of clusters fixed at 16 and 32 and then varying the number of components between 1 and 29 (for 16 cluster) and between 1 and 19 (for 32 cluster). To perform the tensor factorization, we used the `tucker_als` algorithm from the MATLAB Tensor Toolbox 2.2 [BK07].

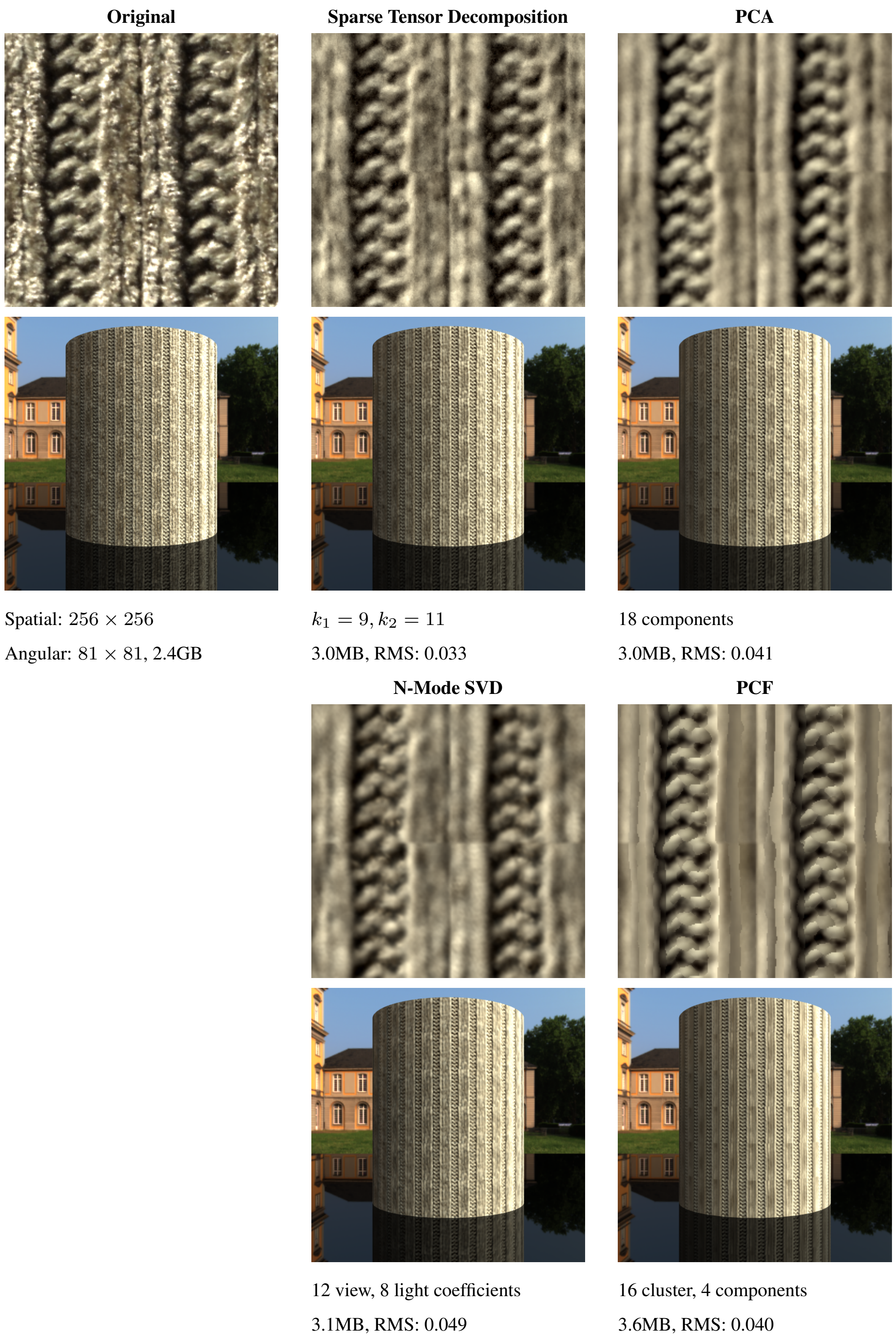
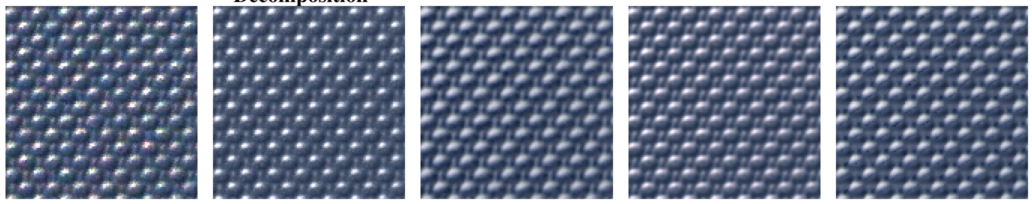


Figure 4.6: *Comparison of rendering quality at the same file size.*



Original	Sparse Tensor Decomposition	PCA	N-Mode SVD	PCF
Spatial:128×128 Angular:151×151	$k_1 = 7, k_2 = 10$	9 components	32 × 32 spatial 40 view, 20 light coefficients	2 cluster, 5 components
2.1GB	1.6MB	1.6MB	1.6MB	1.7MB
	RMS: 0.024	RMS: 0.034	RMS: 0.040	RMS: 0.036

Figure 4.7: *Comparison of rendering quality at the same file size for an example material with strong highlights. Our sparse tensor factorization conserves especially high-frequency components of the dataset considerably better than the other compression techniques.*

As was done in [WWS*05], we compressed the spatial resolution to half of the resolution of the input images. Then we calculated two series, one with the same number of entries in the modes representing light and view direction, and one for which we used twice as many entries to represent the view than for the light. For our approach, we used the parameter selection technique from Section 4.2.2.

In Figure 4.4, the file sizes necessary to achieve a given RMS error are plotted. Compared to the best of the other techniques, the PCA based Full Matrix Factorization, our sparse tensor decomposition achieves compression ratios which are by a factor of three to four higher at the same RMS error. In Figure 4.6, we show a direct comparison between the quality of BTF datasets compressed with the different techniques at about the same file size. Our approach preserves considerably more high-frequency details. Especially in the protruding parts of the knitted fabric at the center and the borders, which thus exhibit the strongest parallax effects, more details are visible. Similarly, in the blue plastic sample shown in Figure 4.7 our technique preserves the specular highlights considerably better than the other approaches. A result at a very high compression ratio for a large BTF dataset of 14.8 GB is show in Figure 4.8. Compared to the other techniques, our approach preserves the structure of the fabric, shadows and occlusion effects better.

To compare the rendering performance of different compression techniques under realistic conditions, we ray traced a small sample scene containing a sphere with the knitted fabric. In Figure 4.5, we compare the total required rendering times. We also performed measurements for BTF compression via N-mode SVD [WWS*05].

Here, the rendering time can be considerably accelerated by multiplying the tensor with the matrices in the spatial dimensions in a preprocessing step, which increases the required memory but eliminates the by far most expensive part during reconstruction. However, even in this case the technique is still much slower than the other approaches and thus the results are not included in the graph. A rendering with a RMS of 0.035 required more than 3000 seconds.

For high compression ratios, our technique is faster than PCA, but for higher qualities, it is necessary to increase both k_1 and k_2. While this increases the file size only linearly, the rendering time growth with $O(k_1k_2)$ and thus at a certain point PCA based techniques become faster. However, at this point the rendering quality is for many applications already sufficient. For example, the images in Figure 4.6 are rendered at a RMS of 0.033, where our technique is about 25% slower than the PCA based approach.

In parallel with the first publication of this work, a different approach which also combines clustering and a tensor-approximation has been published in [Tsa09] and in an extended version in [TS12]. The comparison in [TS12] shows that their technique provides a superior rendering performance. However, as the comparison in Figure 4.9 shows, the compression ratios achievable with this approach are inferior to both PCA and the technique presented in this chapter.

Our technique is thus especially well-suited for applications where high compression is necessary because here it offers a combination of better quality together with good reconstruction performance.

4.4 Conclusion

In this chapter, we presented a BTF compression technique based on a sparse tensor decomposition. We use the K-SVD algorithm to decompose a tensor into a small dictionary and two sparse tensors. This representation is very compact, allowing for a compression ratio which is at the same RMS by a factor of three to four better than a PCA based approach. At the same time, the approach is much faster than other tensor decomposition based approaches, achieving a rendering performance similar to matrix factorization based techniques.

In our current MATLAB implementation, the compression times are still quite high. However, in [RZE08] a faster K-SVD techniques has recently been proposed, which might considerably improve upon this. In the future, it would also be interesting to investigate the use of our technique for real-time rendering. This is still a challenging problem as the texture filtering hardware of current GPUs cannot be utilized as it is possible for PCA based techniques. On the other hand, the small memory footprint of our approach is especially for this type of application a very important advantage.

For a similar decomposition of a tensor into a chained product of mode-2 and mode-3 tensors, Oseledets and Tyrtyshnikov introduced the name *tensor train* [OT10] or also *TT-decomposition* [OT09]. In contrast to our work, they do not utilize a sparse decomposition. Our representation thus could be regarded as a *sparse tensor train*. However, since their work was published only slightly before our original publication (their first preprint was from January 2009), we were not aware of these terms and thus did not use it in the original paper but instead called it a Sparse Tensor Decomposition.

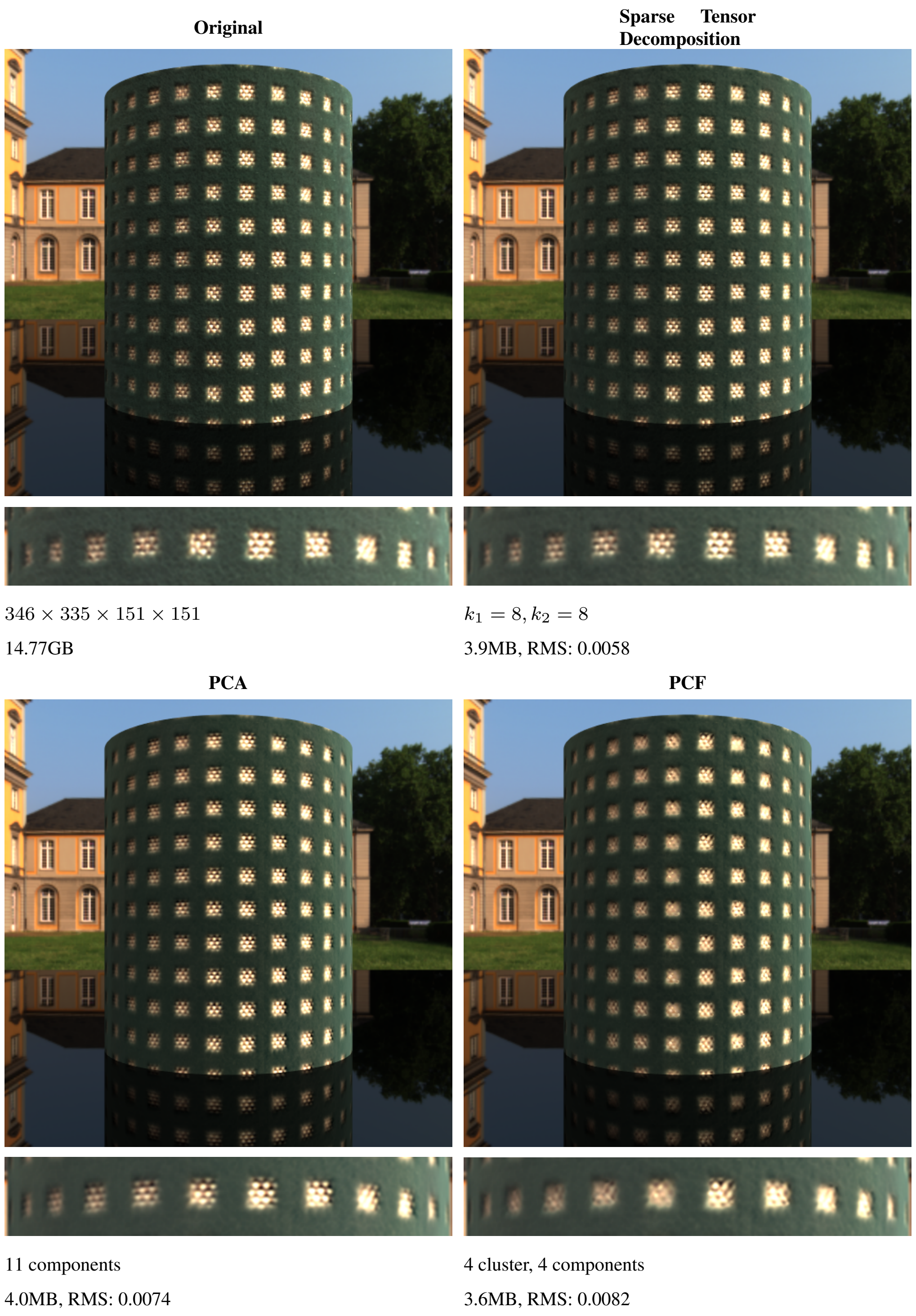


Figure 4.8: *Results at a very high compression ratio ($\approx 1 : 3900$) for a large BTF dataset.*

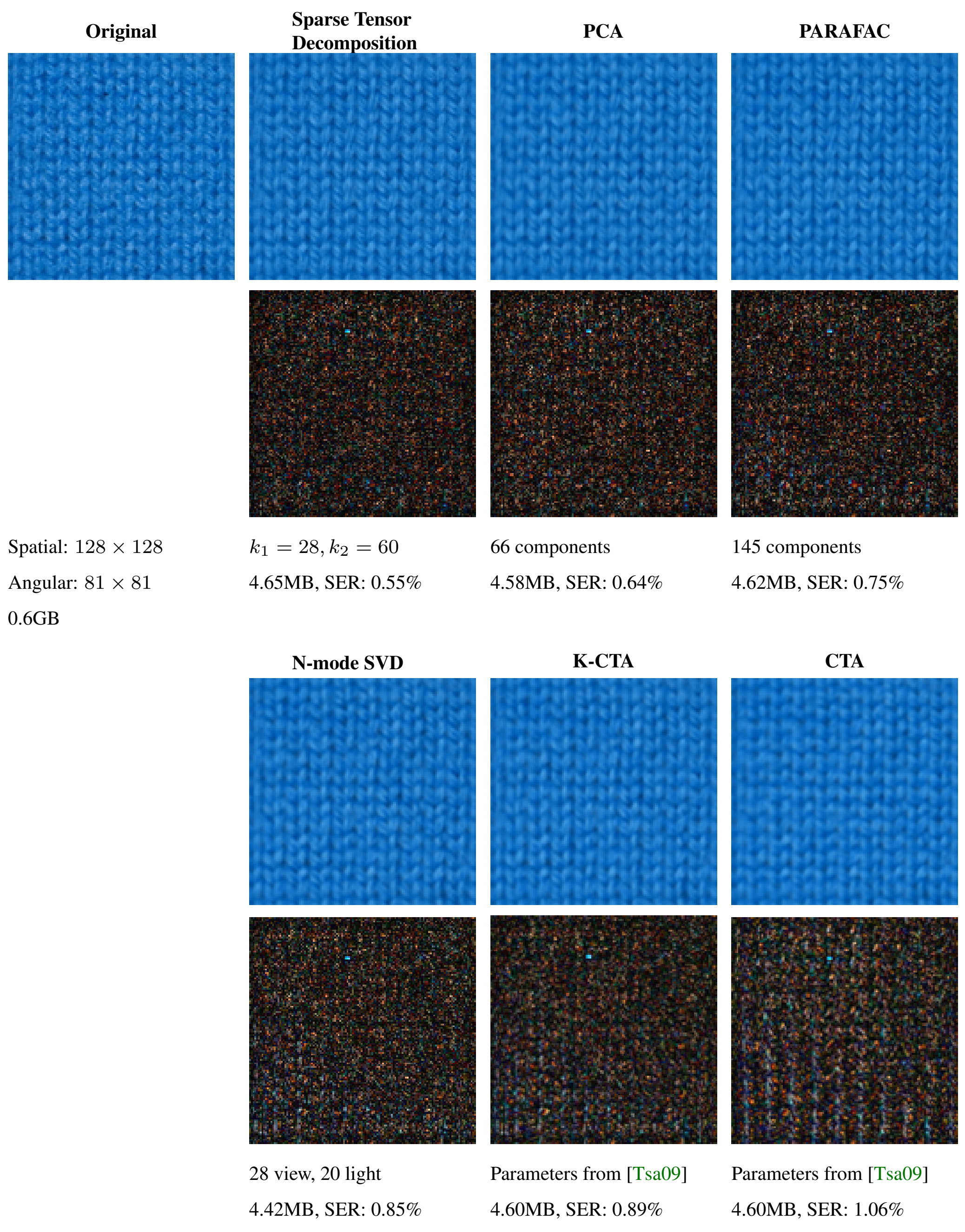


Figure 4.9: *Comparison of compression error at approximately the same file size with the results from [Tsa09]. The results for K-CTA and CTA are taken directly from [Tsa09], the N-mode SVD result was recomputed by us and gave the same error as the result reported in [Tsa09]. The squared error ratio (SER) is computed as the ratio of the mean of the squared errors to the mean of the squared values of the input dataset. The difference images are scaled by a factor of 6.*

CHAPTER 5

HEIGHTFIELD AND SVBRDF RECONSTRUCTION

In the previous chapters, we have focused on techniques to compress BTFs. This is a completely data-driven approach that works without any additional model assumptions, apart from the assumption that the BTF matrix has a low-rank and that the data is band-limited allowing for linear interpolation. However, for many applications, a representation which includes additional model assumptions has considerable advantages. In this chapter, we will therefore investigate the use of a heightfield combined with a spatially varying BRDF. In contrast to BTFs, this representation is far more compact and allows for much easier editing of the acquired materials. However, deriving this representation from a measurement of a material sample is a non-trivial problem.

A possible approach in this context would be to separate the geometry reconstruction, using techniques like laser scanners or structured light, from the BRDF acquisition. For example, in [WSRK11, WRO*12], we describe techniques to integrate structured light reconstructions into our BTF measurement setup and how to improve the quality of the resulting geometry by using super-resolution and additional Helmholtz normals. These approaches are capable to provide high-quality geometry and we will therefore apply our approach from [WSRK11] in Chapter 10. Still, the separated geometry acquisition increases the measurement time and complexity of the measurement device and results in additional calibration problems. All this contributes to higher measurement costs. Since digital images of the sample are needed in any case for the SVBRDF reconstruction, it would be desirable to reconstruct also the geometry from the same images and avoid the additional acquisition steps. Unfortunately, reconstructing geometry and SVBRDF together is a difficult problem, since both are mutually dependent.

In [GCHS05], an algorithm is proposed which alternates between SVBRDF and geometry estimation. However, the use of photometric stereo limits this approach to a single viewpoint. As a result, strongly view dependent BRDF effects, such as the Fresnel effect, cannot be reconstructed correctly. An approach combining multiple-views with photometric stereo reconstruction to overcome this restriction is proposed in [PCF05], but it is limited to a very restrictive BRDF model and requires the manual selection of corresponding points in the images to combine several view directions. A common problem of both techniques is that they estimate a normal field and then perform an integration step to obtain geometry. This is prone to low-frequency drifts as small errors in the estimated normals can accumulate. Furthermore, the reconstructed normals can be inconsistent and thus no corresponding geometry may exist.

To overcome these restrictions, we propose a new algorithm which combines multi-view stereo and photometric stereo. It extracts both a heightfield and a spatially varying BRDF from images taken under different view and light directions. In contrast to the previous approaches, our objective function does not depend on the normals, but instead directly on the 3D geometry of the reconstructed object. This has several important advantages. We can reproject points on the surface into the input images and thus optimize our objective function for all viewpoints simultaneously. We do not need any additional integration steps and therefore do not have to cope with inconsistent normals. Furthermore, shadowing and masking effects can be included in the objective function in a straightforward manner and it also becomes possible to take interreflections into account. Both the reconstructed geometry and the recovered SVBRDF are highly accurate, resulting in a faithful reproduction of the materials characteristic appearance, which is of paramount importance in the context of material rendering.

As in [GCHS05], we iterate between geometry and SVBRDF reconstruction. In contrast to photometric approaches, our new objective function can no longer be optimized for each surface position independently, but has to be minimized for the whole geometry at once. Since we are reconstructing materials, we can restrict ourselves to the assumption of nearly planar surfaces, which can be represented by a heightfield. In this case, the resulting optimization problem can be solved efficiently with a local optimization algorithm by taking advantage of the sparsity of the resulting Hessian matrix.

To cope with interreflections within the surface meso-structure, we perform an additional step during the iterative optimization, in which the light exchange within the surface geometry is approximated using the currently available heightfield and spatially varying BRDF. For this calculation, an efficient GPU implementation is used, as otherwise this step would be prohibitively expensive. With the estimate of the light transport, we are able to remove the contribution of indirect light from the input images. The resulting images can then be used to reconstruct better heightfields and SVBRDFs. This way, we are able to obtain spatially varying BRDFs which are no longer influenced by the indirect light, but instead faithfully represent the actual material. This is especially important for editing applications, as here the interreflections have to be recalculated to be consistent with changes to the geometry.

This chapter corresponds to the paper "Heightfield and spatially varying BRDF Reconstruction for Materials with Interreflections" by Roland Ruiters and Reinhard Klein, published in *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):513–522, April 2009.

5.1 Previous Work

Though only a few works [Geo03, PCF05, GCHS05] describe techniques for the simultaneous acquisition of spatially varying BRDFs and geometry from images only, a lot of work has been done on the two problems individually. In Section 2.2, an overview over a wide range of techniques to acquire SVBRDFs under the assumption of known geometry is given. We will therefore focus on the geometry reconstruction from images in this section. This reconstruction is one of the central topics in computer vision and a lot of work has been done in this area. We will thus only give a short overview here. Most techniques can be classified into *photometric stereo* techniques, which capture objects from a static viewpoint under varying light, and *binocular* and *multi-view stereo* techniques, which use multiple viewpoints.

Photometric stereo [Woo80] techniques observe a given object under different illumination directions to reconstruct a normal map, which then can be integrated to obtain the object's geometry. While early work in this area assumed the object to be lambertian and homogeneous, recent approaches overcome this limitation either by using sample objects of the same materials and with known geometry [HS03], by determining the direction of specular highlights to estimate the normals [CGS06], or by iterating between SVBRDF and shape estimation [Geo03, PCF05, GCHS05, HB08]. In [NRDR05], geometry obtained from a 3D scanner is combined with photometric normals to obtain high-quality models without low-frequency drift.

Binocular and multi-view stereo techniques determine the object shape either by determining correspondences between pixels in images taken from different viewpoints and then triangulating the position on the surface or by finding an object shape that maximizes a photo-consistency metric. Though a lot of work has been done in this area, establishing the correspondences between pixels remains a difficult problem, especially when the objects have very homogeneous or specular surfaces. Furthermore, these techniques are prone to noise. An evaluation of several multi-view stereo techniques can be found in [SCD*06].

While the photometric stereo techniques often suffer from a low-frequency drift, resulting from the accumulation of small errors in the estimated normals during the integration step, multi-view stereo techniques tend to exhibit high-frequency noise. For this reason, several combinations of the two techniques have been proposed. In [NZG05], images taken under several light directions are combined to obtain viewpoint robust features, which are then used for stereo reconstruction. In [CLL07], a technique for the integration of several normal maps, reconstructed from different points of view, is proposed. For this, a level set method is used to reconstruct a surface which minimizes the error between the estimated normal maps and the surface normals. Another approach is to alternate between the photometric reconstruction of the surface and a parallax correction step. This step uses the surface estimate to resolve the correspondences between the multiple views improving the next photometric stereo reconstruction. In [LHYK05], this technique is applied to lambertian objects and in [PCF05] a spatially varying BRDF is additionally estimated. However, both techniques require knowledge about corresponding pixels to combine images from different viewpoints, which are either selected manually [PCF05] or obtained by tracking features in the images [LHYK05]. In [JCYS04], [VHC06], and [EVC08] algorithms are proposed, which alternate between the estimation of surface normals and the evolution of a mesh which corresponds to these normals to resolve the parallaxes.

All of these techniques to combine photometric and multi-view stereo separate the geometry reconstruction from the normal estimation. To cope with the aforementioned ambiguous or inconsistent normals, a second optimization step has to be conducted, which reconstructs the geometry by minimizing the normal error. This increases the complexity of these algorithms considerably. To avoid these problems we directly reconstruct the geometry. A similar approach is taken in [YXA04], where a star-shaped polygonal mesh is directly optimized to correspond to the input images. However, their technique is limited to homogeneous materials.

A technique for the reconstruction of surfaces in the presence of interreflections is proposed in [NIK90]. As in our approach, the algorithm iteratively refines the reconstructed surface by subtracting an approximation of the indirect light from the measured intensity. This approximation is calculated from the currently available surface geometry under the assumption of a lambertian object. There are also a few methods to cope with the influence of indirect light during BRDF acquisition from objects. In [YDMH99] a technique is described, which iterates between light exchange simulation and BRDF estimation. In [WSL04] a set of homogeneous BRDF parameters is determined with a simulated annealing algorithm. The fitting error is calculated by rendering the object with a ray tracer and comparing the result to the input image. This way, interreflections are taken into account. However, both approaches require knowledge of the scene geometry. Direct and indirect

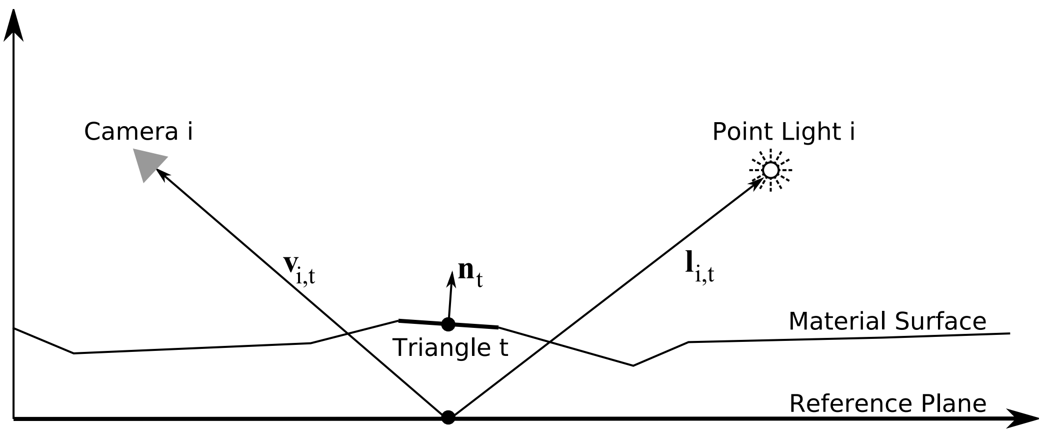


Figure 5.1: *Illustration of the capture setup for image I_i*

illumination can also be separated by using polarization filters or with structured light techniques, e.g. [NKGR06, CLFS07, CSL08]. However, these approaches require special measurement setups.

5.2 Objective Function

We assume that a measurement as it is provided by a typical BTF acquisition device, such as the ones described in Section 2.1, is available. We thus have a set of images $\{I_i\}$ as input each of which has been taken under illumination by a single point-light source at position L_i from camera position C_i. For this, we require the necessary calibration to determine both the position of the camera and light source as well as the intrinsic camera parameters. Additionally, we assume that the sample is nearly planar and that a reference plane is available, which is already quite close to the surface of the material. In the following, we will use a coordinate system which is aligned to this reference plane, with its origin in one corner of the material sample. We denote the bilinearly interpolated intensity at the position $\mathbf{x}$ in the image I_i as $I_i(\mathbf{x})$.

As in [GCHS05], the BRDF of the surface is modeled as a linear combination of a set of *basis BRDFs*. Our objective is thus to find a heightfield h, a weight map γ and a matrix of basis BRDF parameters α which describe the input images as faithfully as possible. For the reconstruction, we assume the heightfield h to be triangulated and the triangles Δ_t to have consecutive indices t. For simplicity of notation, henceforth we will identify the triangles by their indices. Therefore, the normal of a triangle is given by $\mathbf{n}_t$ and the local light and view directions from the center of this triangle for the image I_i are denoted $\mathbf{l}_{i,t}$ and $\mathbf{v}_{i,t}$. Since the distance

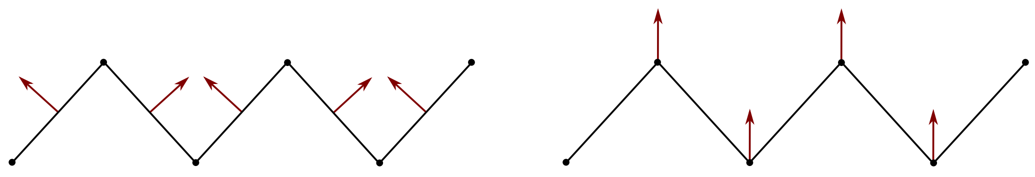

(a) Triangle normals (b) Vertex normals

Figure 5.2: *Normals calculated either for each triangle, or for the vertices, by averaging triangle normals*

to both camera and light source is large when compared to the size of structures on the surface, we approximated these directions using points on the reference plane neglecting the heightfield value at that position. See Figure 5.1 for an illustration. For each of these triangles, the reconstructed intensity when seen from C_i and illuminated from L_i is denoted by $I'(t, \mathbf{v}_{i,t}, \mathbf{l}_{i,t})$ and is modeled as a function of the parameters h,γ, and α. We define the objective function as

$$E(h, \gamma, \alpha) = \sum_{t,i} \chi(t, \mathbf{v}_{i,t}, \mathbf{l}_{i,t}) \| I_i(\Pi_i(t)) - I'(t, \mathbf{v}_{i,t}, \mathbf{l}_{i,t}) \|^2,$$

where χ takes care of ignoring samples which are masked and shadowed and the function $\Pi_i(t)$ maps the center of the triangle Δ_t into the input image I_i using the interpolated heightfield value to obtain its 3D position. The actual reconstruction is performed by minimizing this objective function. Each triangle in our heightfield is smaller than one pixel in the input images, and thus it is sufficient to evaluate one sample per triangle instead of integrating over the triangle surface. In the following, we will describe the different components of the objective function in more detail.

5.2.1 Geometry Model

We model the geometry of the surface as a triangulated heightfield, assuming a constant normal and BRDF for each triangle. Since we choose our heightfield resolution in such a way that it corresponds to the pixel size in the input images with the highest resolution, this is a sufficiently precise approximation of the actual appearance of the surface and allows us to evaluate the error using one sample for each triangle. Note that it is important to model the surface with per-triangle normals instead of interpolated vertex normals. If the intensity is evaluated at the vertices instead, the normals of adjacent triangles have to be averaged to obtain this normal. As illustrated in Figure 5.2b, the averaged normals can be smooth even for very strongly oscillating heightfields and thus photometric methods cannot distinguish both cases. This results in additional degrees of freedom in our objective function. For homogeneous materials, the multi-view reconstruction is also not sufficient to resolve this ambiguity, and thus the algorithm can fit oscillating heightfields to smooth surfaces, resulting in strong high-frequency artifacts.

5.2.2 Spatially Varying BRDF Model

We use the well-known Cook-Torrance reflectance model [CT82] with the Beckmann normal distribution function and the Schlick approximation for the Fresnel term [Sch94], which is important in a multi-view setting. We have chosen this model, since, as analyzed in [NDM05], it is able to represent most materials with a single lobe quite faithfully and thus requires the estimation of just a rather low number of parameters. However, other BRDF models could be used with our technique, too. The basis BRDFs are described by a matrix of parameter values, where $\alpha_{j,k}$ gives the k^{th} parameter of the j^{th} basis BRDF. We will denote the vector of parameters for the j^{th} basis BRDF with α_j. We use the function $\rho_c(\mathbf{n}, \mathbf{l}, \mathbf{v}, \alpha_j)$ to denote the Cook-Torrance BRDF in dependence on this parameter vector α_j and the normal $\mathbf{n}$, light direction $\mathbf{l}$ and view direction $\mathbf{v}$.

The spatially varying weights for the basis BRDFs are represented by a vector $\gamma_j(t)$ which gives the weight for the contribution of the j^{th} basis BRDF to the BRDF of the triangle t. We assume, that two adjacent triangles, forming a square in the height field, have the same BRDF and thus store only one weight for each of these, allowing us to represent the weights in a 2D-map. We restrict our BRDF weights to convex combinations, and thus all γ_j are non-negative and sum to one at every position on the surface. The modeled intensity of a triangle is then given by:

$$I'(t, \mathbf{l}, \mathbf{v}) = \sum_j \gamma_j(t)\rho_c(\mathbf{n}_t, \mathbf{l}, \mathbf{v}, \alpha_j))(\mathbf{l} \cdot \mathbf{n}_t).$$

5.2.3 Shadow and Masking Model

We have to ignore samples which are either occluded or shadowed. This is done by the term $\chi(t, \mathbf{v}, \mathbf{l})$. It is composed of a term $V(t, \mathbf{v})$ which is 1 if the triangle is visible and otherwise 0, and a corresponding term $S(t, \mathbf{l})$ for shadows. Furthermore, it renormalizes the error by dividing through the total number of samples that have been taken into account, as otherwise, increasing the number of masked or occluded samples would decrease the reconstruction error. It is thus defined as follows:

$$\chi(t, \mathbf{v}, \mathbf{l}) = \frac{V(t, \mathbf{v})S(t, \mathbf{l})}{\sum_{t',i'} V(t', \mathbf{v}_{i'})S(t', \mathbf{l}_{i'})}.$$

5.3 Optimization

The minimization of the objective function $E(h, \gamma, \alpha)$ requires solving a non-linear optimization problem with a large number of unknowns. Therefore, a direct optimization of all unknowns at the same time with a local optimization algorithm easily gets stuck in a local minimum. However, due to the structure of this problem, the use of global optimization algorithms is impractical. Therefore, as in [GCHS05], we solve the problem by alternating between the optimization of the basis BRDF parameters, the weight map parameters and the heightfield until the error no longer decreases sufficiently. This way, we can use optimization algorithms which are specially tailored to each of the subproblems.

5.3.1 Initialization

As all three optimization problems depend on each other, it is not clear which initial values should be used. Our approach, which worked well in practice, is to first estimate an averaged homogeneous BRDF by fitting a single basis BRDF assuming a planar surface and a constant weight map. For the BRDF estimation, we are using a genetic algorithm and thus do not need to specify a start value. In the next step, this homogeneous BRDF is used to reconstruct a first estimate of the heightfield. For this, the reference plane is used as initial geometry, which means that we set all heightfield values to zero. In order to get a reasonable first heightfield, we are only using view directions up to an angle of $30°$ relative to the reference plane normal for the heightfield reconstruction. For these views, parallax effects are rather small, and in our experiments this initialization resolves the pixel correspondences to a sufficient degree for the stable convergence against an initial heightfield. To further improve the stability of the optimization, during both of these initial steps we average the colors of the input images and use only the intensity during optimization.

Once we have found this initial estimate of the heightfield, we have to initialize the weight maps γ_j. This is done by successively adding further basis BRDFs up to a user specified total number. We thus start with two basis BRDFs. Since for many materials a strong correlation between surface geometry and BRDF exists, for these we use the heightfield to initialize the weight maps γ_0 and γ_1. This is done by scaling and shifting the heightfield values in the range between 0 and 1 to obtain the height $h'(t)$ at the position of triangle t and then setting $\gamma_0(t) = h'(t)$ and $\gamma_1(t) = 1 - h(t)$. Using these weight maps, the BRDF estimation is now repeated for two basis BRDFs and then weight map and basis BRDF estimation are alternated until the error no longer decreases sufficiently between two iterations.

For materials which do not exhibit a strong correlation, the use of a clustering step, like the one described in [GCHS05] may be better, though.

When more than these two BRDFs are to be used or for materials with no correlation between meso-structure and BRDF, further basis BRDFs are added successively. Based on the reasonable assumption that the error is especially high at places where a further BRDF has a high contribution, the new weight map γ_j is initialized from the error obtained during the previous optimization step at the position of the entry γ_j. This error is scaled to 0 and 1, assigned to the new channel and then all basis BRDF weights are rescaled to enforce the convex combination constraint on γ. Now, BRDF and weight map estimation are again iterated and then this procedure is repeated for the next BRDF.

5.3.2 Basis BRDF Optimization

As already noted in [NDM05], because of their non-linearity, the fitting of a single BRDF model with local optimization algorithms is already very prone to local minima. The simultaneous fitting of several basis BRDFs against samples consisting of linear combinations of these BRDFs is even more difficult, as a higher number of parameters has to be optimized at the same time, and a good initial guess of the BRDF parameters is crucial. A manual selection as suggested in [NDM05] is impractical for an iterative algorithm. In each iteration, the BRDF optimization is once performed with the last values as initialization and once with the initialization obtained from a genetic algorithm and then the better result is kept. This way, a bad initialization from the genetic algorithm does not lead to an increase in the error, but we are also able to leave a local minimum. We used the GADemeGA algorithm from the GALib library [Wal96] with 30 populations each having 30 individuals and a mutation probability of 10% and the Quasi-Newton algorithm from [MOHW07] using analytically computed first derivatives of the Cook-Torrance BRDFs.

Even with this initialization, the BRDF estimation was still prone to local minima in our experiments. A further improvement can be made by reducing the dimensionality of the optimization problems. To this end, we iterate between the solution of different subsets of the parameter matrix α, keeping the remaining parameters fixed. We thus first optimize the diffuse components of all BRDFs together. Then, each of the basis BRDFs α_j is optimized individually and finally the full matrix α is optimized. To further reduce the number of parameters that have to be optimized, this procedure is first performed for averaged colors and then repeated for color BRDFs, initializing the diffuse and specular color with the gray-scale intensity obtained in the previous step.

5.3.3 Weight Map Optimization

Given a set of basis BRDFs and a heightfield, determining the weight map can be performed independently for each entry of the weight map. The color of the corresponding triangles under the different light and view directions is extracted from the input images in a preprocessing step. Since we are considering linear combinations of the basis BRDFs, determining the γ_j is a least squares problem. However, it is necessary to enforce the non-negativity and the convex combination of the γ_j during the optimization. Therefore, we perform this optimization also with a Quasi-Newton optimizer.

5.3.4 Heightfield Optimization

The heightfield optimization is a non-linear problem. This results from the fact, that changes in the heightfield alter the derived normals, which in turn result in non-linear changes in the color because of the non-linearity of the BRDF model. Furthermore, the problem cannot be solved for each heightfield entry independently, as each entry depends on the neighboring entries through the derived normals. Therefore, all entries in the heightfield are coupled to each other and the problem has to be solved for the whole heightfield at once. Thus, e.g. for a 129×129 sized heightfield, an optimization problem with 16,641 unknowns has to be solved. The Quasi-Newton optimization algorithm we used for the basis BRDFs and weight maps cannot be used directly for problems of this size. However, the Newton algorithm can be adapted to take advantage of the special structure of this problem.

The gradients for the heightfield optimization can be calculated using finite differences. Note that each entry in the heightfield influences only six triangles. Thus, it is only necessary to recalculate the error for these six triangles when calculating the finite-difference derivative of one of the unknowns.

Standard Quasi-Newton methods cannot be used, because the size of the inverse of the Hessian matrix, which is approximated by these algorithms, increases with the square of the number of unknowns. However, for this special problem, the Hessian matrix has a sparse structure. As only six terms in our objective function depend on one heightfield entry, the first derivative of the sum only contains these terms, which together depend on only seven heightfield entries. Each row of the Hessian matrix of this problem thus contains just seven non-zero entries. See Figure 5.3 for an illustration. Therefore, it can be approximated with finite-differences in linear time in the number of unknowns and it can be inverted efficiently using the conjugated gradients method. We used the TN [Nas84] algorithm for this

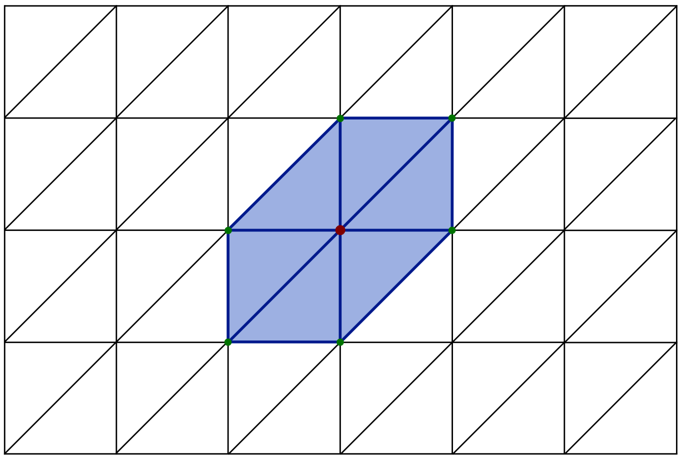


Figure 5.3: *One entry in the heightfield (red) influences six adjacent triangles (blue), which in turn depend on six additional entries of the heightfield (green).*

minimization, which we modified to take advantage of the sparsity of the Hessian matrix.

In the first optimization steps, the correspondences are only resolved correctly for views close to the top-view and thus the algorithm can, like photometric techniques, reconstruct a first estimate of the surface from these views, but it is more difficult to determine the position of this surface relative to the reference plane. To find this position we perform an additional optimization step, in which the whole heightfield is shifted perpendicular to the reference plane and scaled uniformly. Afterwards, most correspondences are already resolved correctly. Additionally, it is possible to better compensate for large-scale drift by performing an optimization on a low resolution grid (we use a resolution of 12×12) and then shift each pixel by the bilinearly interpolated values from this grid.

We sometimes observed individual pixels or small groups of pixels (up to about 5 pixels) which got stuck in a local minimum. However, these minima could be resolved by performing an additional step, which eliminates these outliers by searching for entries with very high errors and setting these to the average of their surrounding entries. If this step reduces the total error, the new values are kept, otherwise the old ones are restored.

For most results shown in this chapter, this algorithm converged to reasonable heightfields without obvious artifacts. However, in experiments on datasets with large parallaxes the algorithm got stuck in a local minimum, which was visible as discontinuities in the reconstruction. To resolve these, we utilize a multi-resolution initialization scheme, where the optimization is first performed on subsampled versions of the input images, where the parallaxes are smaller, and the results are

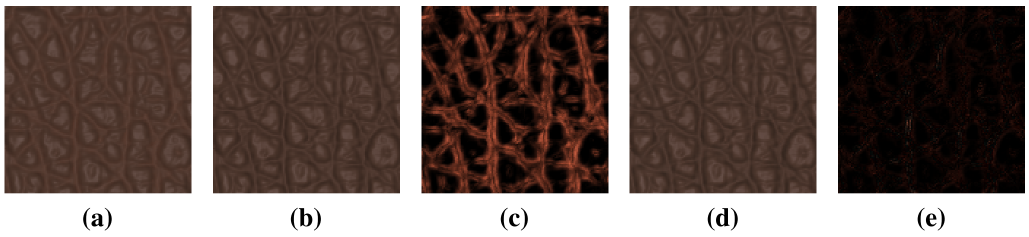


Figure 5.4: *Two images from a synthetic sample dataset, rendered (a) without and (b) with interreflections. (c) shows the contribution of indirect light simulated with the first reconstruction and (d) is the image then obtained by subtracting (c) from (a). The remaining difference to the image without interreflections (b) is shown in (e). (Both (c) and (e) are scaled by a factor of 10.)*

then used to initialize the reconstruction on the next higher resolution. Furthermore, we add a regularization to the objective function, which penalizes the square of the first derivative of the heightfield. The strength of this regularization is successively decreased during the optimization. This way, it helps to prevent local minima during the initialization, but does not smooth out fine details in the final reconstruction. These additional improvements were only used for the results shown in Figure 5.9.

The full optimization of all heightfield entries, the optimization of the offset and scale, optimization on a low-resolution grid, and the elimination of outliers are iterated until the error no longer decreases sufficiently between two iterations.

5.3.5 Acceleration

When a large number of input images has to be processed, an efficient implementation of the reconstruction algorithm is necessary. Since especially the evaluation of the objective function is very expensive, we used several optimizations reducing the runtime considerably.

The two slowest parts of the reconstruction algorithm are the genetic algorithm for BRDF estimation and the heightfield optimization, as both require a very high number of evaluations of the objective function. However, the evaluation of the objective function is easily parallelizable and can thus be accelerated by performing this calculation on a GPU. A straight-forward CUDA implementation of the objective function evaluation accelerated the BRDF optimization by a factor of about 5 and the heightfield optimization by a factor of about 20 when running on a GeForce 8800 GTX compared to a moderately optimized parallelized version running on a Q6600 quad-core CPU. Furthermore, we found that using rectangular facets with a constant normal instead of two triangles, neglecting the normal of one of the triangles, to approximate the area in between four heightfield entries is usually sufficient to reconstruct good heightfields and reduces the number of samples that have to be evaluated and the storage needed on the GPU.

Instead of optimizing the BRDFs over the whole dataset, we instead just optimize the BRDF for a subset of the full dataset. Typically, about $100{,}000$ samples are drawn, using a simple binning scheme on the weight map to ensure that all basis BRDFs are represented sufficiently by the samples.

Even though during each optimization step the objective function always decreases, it is not guaranteed that the algorithm converges when these optimizations are used. This results from the fact, that we use different subsets of the whole dataset and we thus do not optimize exactly the same objective function during each step. We therefore simply iterate the algorithm until the weight map reconstruction error no longer decreases.

5.4 Interreflections

Interreflections can result in wrong estimates of the heightfield and can disturb the SVBRDF reconstruction. Since grooves in the surface are brightened by the interreflections, the BRDFs estimated for these parts of the surface are also too bright, when this effect is not considered during the reconstruction. In Figures 5.4a and 5.4b an example for this effect is shown.

To remove this brightening of the SVBRDFs and to improve the quality of the heightfields further, we first calculate the contribution of the indirect light to the measured brightness of each pixel and then subtract it from the input images. To get an estimate of this contribution, we simulate the light exchange within the meso-structure, using our current estimates of the heightfield and the spatially varying BRDF, for all combinations of light and view directions present in the input dataset. After these simulations, we render views of the heightfield from the camera viewpoints, in which only the contribution of the indirect light is present. See Figure 5.4c for an example of such an image. These renderings are then subtracted from the input images. By rendering the heightfield from the viewpoints present in the input dataset we take care of parallax effects and occlusions.

Since we use our first erroneous BRDF estimate for the light exchange simulation, we overestimate the contribution of the indirect light in the first step. When we subtract these overestimated values, this results in the next step in BRDF estimates which are too dark. However, as long as the contribution of indirect light is smaller than the contribution of direct light, which can be assumed to be the case on nearly all surface which can be described by heightfields, this second estimate is still better than the first one and thus this algorithm converges against the real BRDF. In our experiments, even after the first iteration the estimates were very close to the correct result. An example for this is shown in Figure 5.4d, which is obtained by subtracting the interreflections estimated using the first reconstruction of the heightfield and SVBRDF from the original image. As the difference image in Figure 5.4e shows, this first step already removed nearly all interreflections.

5.4.1 Simulation of the Interreflections

The calculation of the interreflections is repeated several times during the iterative reconstruction process and has to be performed for all input images. Therefore, an efficient method to perform this simulation is needed. If a common path tracer is used, the simulation would take several minutes for each image. We thus implemented a path tracer on the GPU, using a technique similar to the one proposed in [HDKS00]. For each point on the surface, we first precompute for a fixed set of directions in which distance a ray originating from this point will hit the heightfield again. This information can then be used to perform the whole path-tracing algorithm in a pixel shader, as intersection tests can now be calculated with a single texture look-up. We use a non-uniform sampling of the hemisphere, as in the heightfields usually only rays reflected in very shallow angles hit the surface again. Therefore, we scale the uniform sampling in such a way, that primarily these rays are considered and adjust the probability distribution function accordingly to compensate for the resulting bias. We performed our simulation with 128 rays for each pixel and simulated up to three bounces. On a GeForce 8800, the algorithm then simulated the light exchange within a 129x129 sized heightfield in about 3.5 seconds.

This technique can also be used to synthesize a new BTF with interreflections from edited versions of the reconstructed representation, which can then again be used during rendering. Furthermore, it is suitable for interactive preview during the editing.

5.5 Evaluation

We evaluated our technique both on a synthetic dataset, which provides a ground truth and thus allows for a direct comparison of the reconstructed heightfields, weight maps and BRDFs, and on several real-world BTF datasets. To create the synthetic dataset, we simulated the image acquisition process used for the real

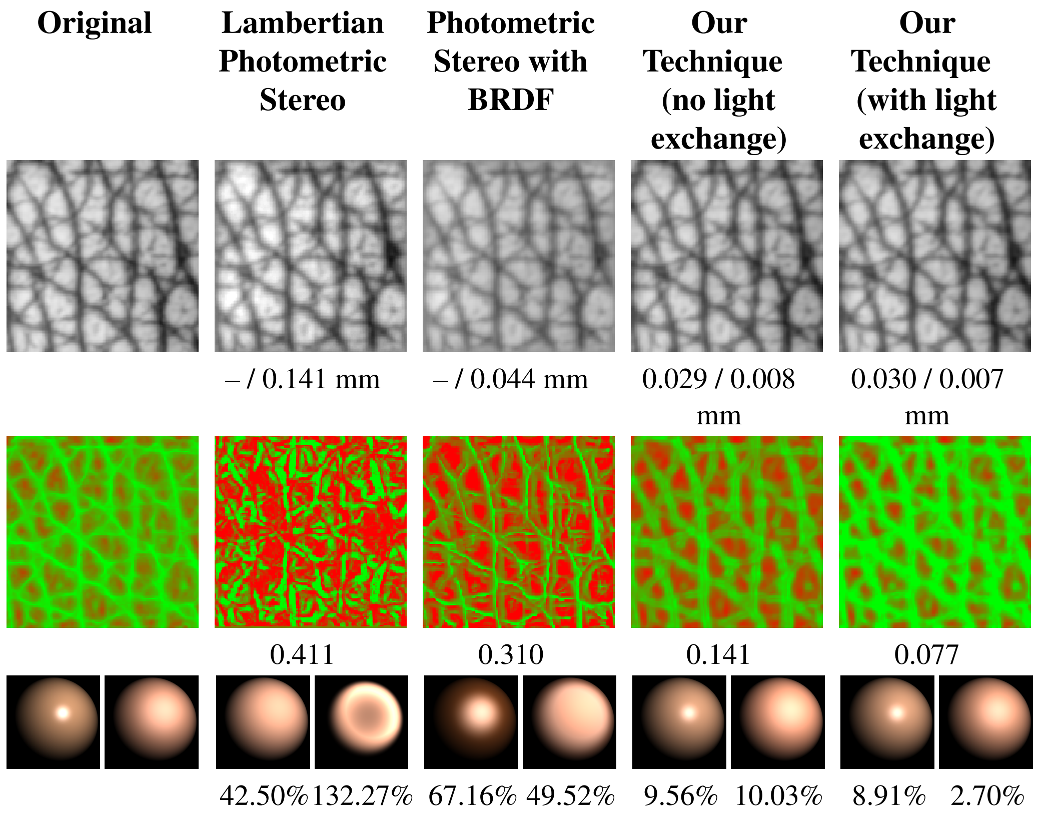


Figure 5.5: *Evaluation on a synthetic dataset. In the first row, the reconstructed heightfields are compared. The average errors compared to the ground truth are given below the images, first the absolute error and then the error obtained by aligning the heightfield to the original one by shifting it upwards/downwards to the position minimizing this error. The next row shows the reconstructed weight maps and gives their average difference to the original. The red channel describes the contribution of the left BRDF and the green channel the contribution of the right one. In the last row, the reconstructed basis BRDFs are shown. Here, the errors are obtained by averaging the relative error over the whole BRDF.*

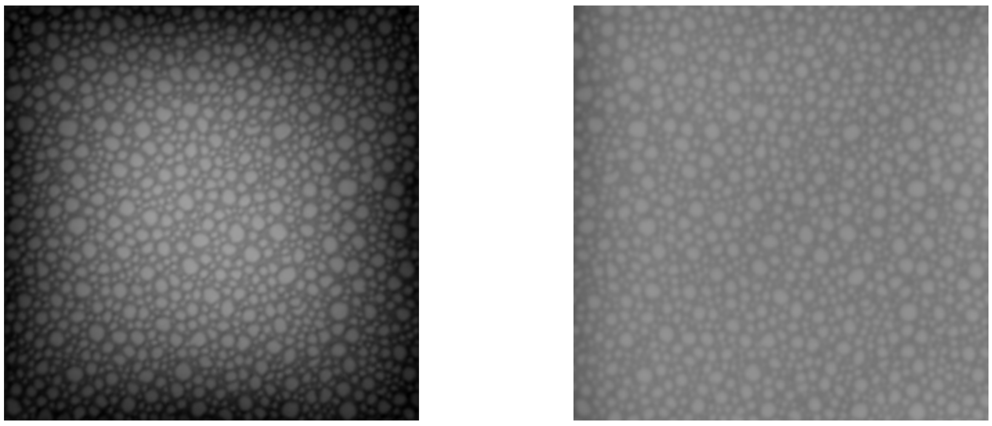

(a) Photometric Stereo **(b)** Our Technique

Figure 5.6: *Comparison of resulting low-frequency drift*

datasets with the Multi-View Dome setup (Section 2.1.3) as exactly as possible using a path tracer. The dataset contained all combinations of 26 viewpoints and 26 light sources. Furthermore, the images were rendered in the same resolution as the photographs taken by the cameras (3 Megapixel). To render the images, a Monte-Carlo path tracer was used, which simulated the interreflections using $4{,}096$ rays for each pixel. The heightfield was placed about 0.3 mm below the reference plane to simulate the imprecise calibration of the reference plane usually obtained in practice. In Figure 5.5, the reconstruction results obtained by either using lambertian photometric stereo [BP03] to reconstruct the heightfield, by iterating between SVBRDF estimation, normal reconstruction and integration, similar to [GCHS05], and by using our algorithm are compared to the ground truth. For all techniques, the same code was used to reconstruct the BRDFs and weight maps. The errors for the heightfield are given in millimeters, using the size of the real measurement setup as reference. The cameras were placed in about 60 cm distance to the sample, which had a size of about 1.5×1.5 cm and a resolution of about 150 pixels in the top-view. Using the photometric techniques, the position of the heightfield with respect to the reference plane cannot be determined, as the reconstructed heightfield can be shifted along the z-axis without any change to the normals. In order to compare to the original heightfield, we shifted the reconstructed one along the z-axis in such a way that the mean error between the two becomes minimal. For our technique, both the error for the absolute reconstructed heightfield values and the error obtained by aligning the height fields are shown. The algorithm reconstructed the position of the heightfield with respect to the reference plane up to a precision of about 0.03 mm, which corresponds to a parallax of about 0.15 pixel in the 30° views we used for the reconstruction. When the heightfields are aligned as for the photometric techniques, the actual surface geometry is reconstructed up to a precision of 0.0069 mm, which is about 1.3% of the total heightfield depth of about 0.55 mm.

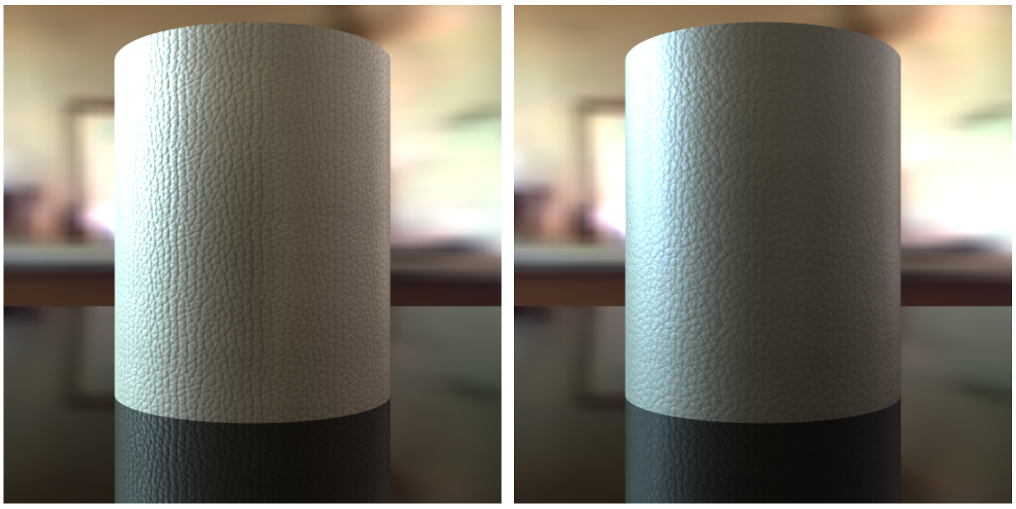
The purely photometric techniques had considerably higher errors in the estimated heightfield. This and the fact that only one view direction can be used result in considerably worse BRDF estimates. Especially the lambertian photometric stereo technique reconstructed a heightfield, which was too deep and thus in consequence the BRDFs estimations based on this heightfield are very bad. Since the heightfield estimated without considering interreflections was already quite good, the correction for interreflections only improved the reconstruction of the actual surface geometry by a small amount from 0.0080 mm to 0.0069 mm. On the other hand, the reconstructed weight map and the BRDF in the cracks improved considerably when the correction for interreflections was performed.

We found, that the use of several view directions eliminates most of the problems with low-frequency drift that purely photometric techniques are prone to. In Figure 5.6, a heightfield reconstructed from a real-world dataset with normal estimation and integration and one obtained with our technique are compared. The dataset contained all combinations 26 view and light directions, however, for the photometric reconstruction only one view direction can be used. We assumed for both reconstructions the same SVBRDF. Our technique removed nearly all low-frequency drift, only in the boundary regions, where not enough correspondences could be established, a small drift remained.

In Figure 5.7 and 5.8, we compare renderings of BTFs with our reconstructions to show that we are able to reproduce the overall appearance faithfully. The BTFs were captured with the gonioreflectometer setup (Section 2.1.1) or the Multi-View Dome setup (Section 2.1.3). The reconstruction was performed with a subset of the BTF datasets containing all combinations of 31 light and view directions.

To evaluate the precision of our heightfield estimation, we also compare in Figure 5.9 two of our reconstructions to heightfields reconstructed from structured light measurements. For this, we have employed the technique from [WSRK11], where first a point cloud is computed and then a surface is extracted via the *Poisson Surface Reconstruction* [KBH06]. Our results clearly show even fine surface details which are not present in the structured light reconstructions. This is the case even though we accepted a rather high noise level for the structured light results to avoid smoothing out the details in the Poisson reconstruction.

To evaluate the absolute error of our reconstruction, we show plots of the Haussdorff distance between our reconstructions and the point clouds obtained via structured light. Most parts of the two heightfields show a small error, however there is a systematic drift of at most $0.25\,\mathrm{mm}$ towards the border of the heightfields. $0.25\,\mathrm{mm}$


(a) BTF

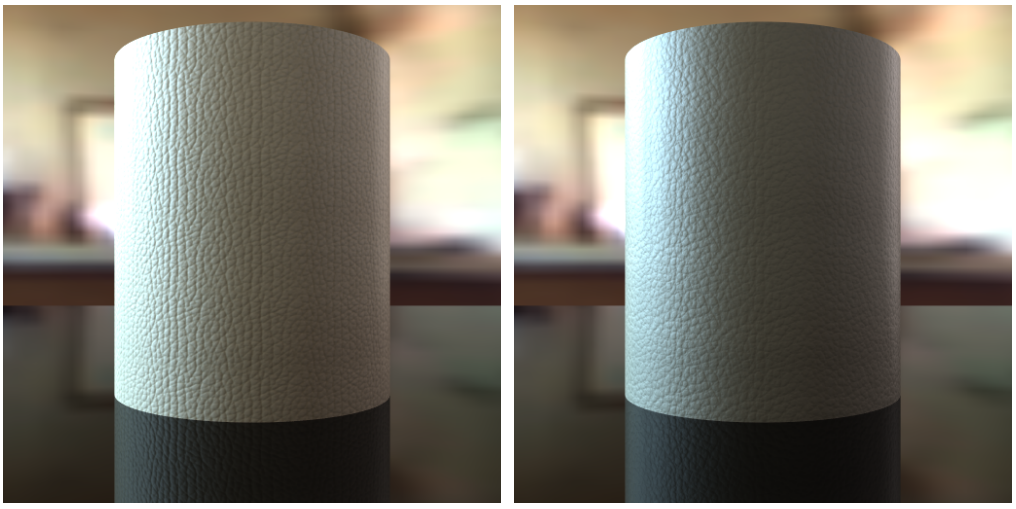
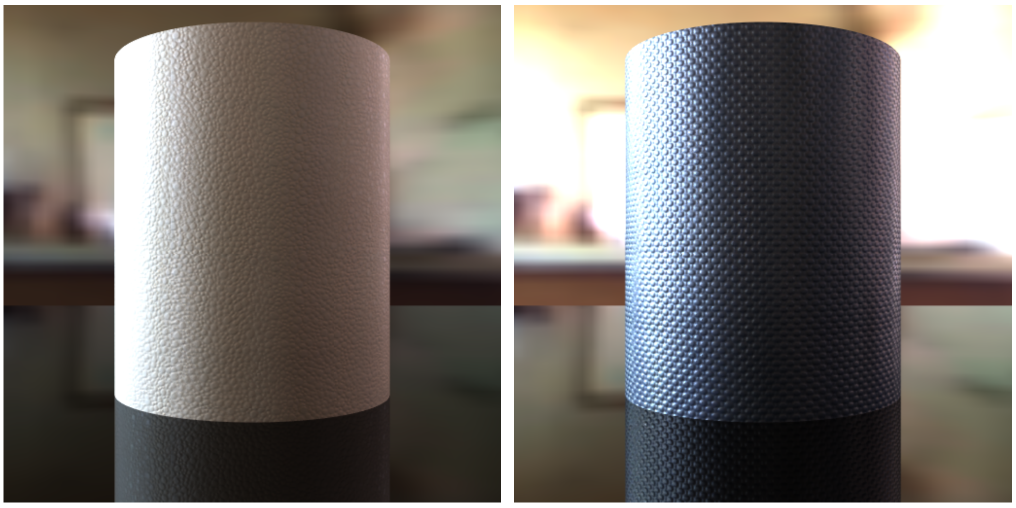

(b) Reconstruction

Figure 5.7: *Comparison between renderings of the BTF and renderings of our reconstructed heightfields and SVBRDFs for two leathers. Environment Map courtesy of Paul Debevec (`http://www.debevec.org/probes/`).*


(a) BTF

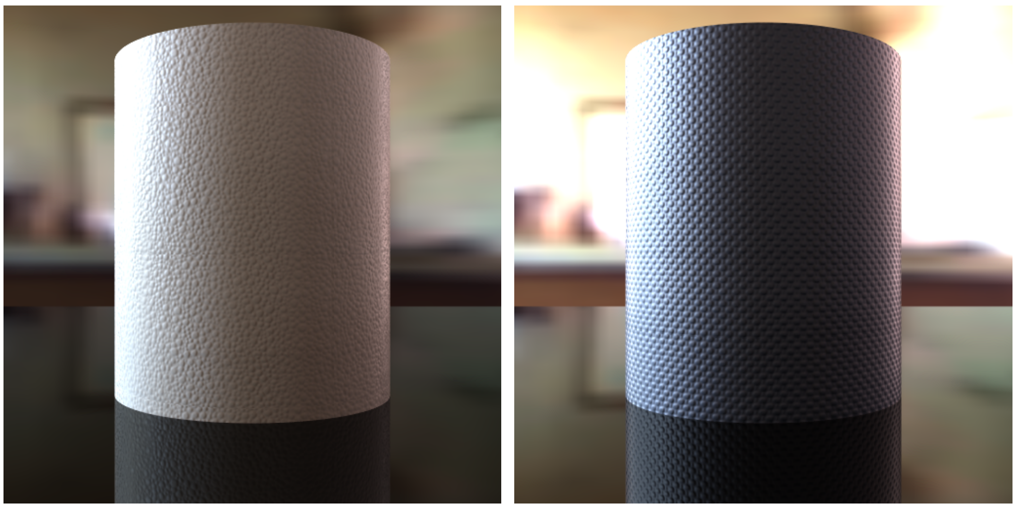
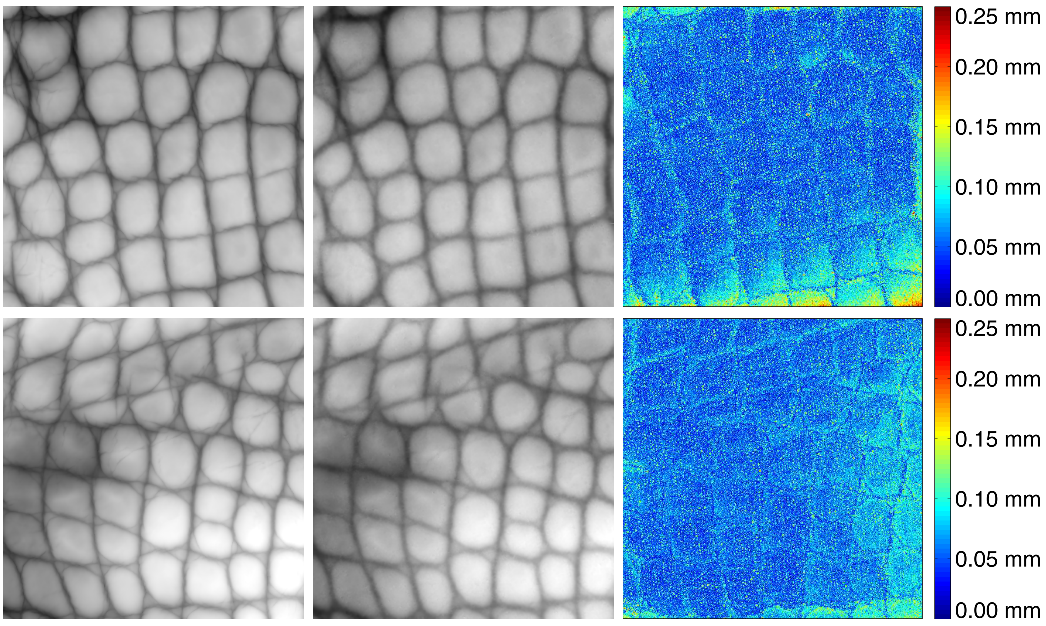

(b) Reconstruction

Figure 5.8: *Comparisons for two plastics with embossed patterns. Environment Map courtesy of Paul Debevec (`http://www.debevec.org/probes/`).*



(a) Our Technique **(b)** Structured Light **(c)** Hausdorff Distance to point cloud

Figure 5.9: *Comparison of our reconstruction technique with a heightmap reconstructed from a structured light measurement [WSRK11].*

height difference correspond to a parallax of approximately 2 px in the lowest views used for the reconstruction ($30°$ angle towards the normal direction). The error at the border is higher, as the heightfields are only constrained towards one side there. Furthermore, only a reduced number of view and light directions have been available at the borders for reconstruction as many were occluded by an adjacent sample during the capture (for this dataset always four material samples were acquired simultaneously).

5.5.1 Timing

The full number of view directions was only used for the BRDF reconstruction, as here shallow view angles are necessary to obtain a good estimate of the Fresnel effect. In contrast, for the spatially varying heightfield and weight map estimation, we limit ourselves to view angles up to 30° from the reference plane normal.

For the BRDF and SVBRDF estimation, the calculation of shadowing and masking was performed before the actual optimization process, which is not possible for the heightfield reconstruction. However, since we limit ourselves to view angles of 30°, the influence of these effects is small during this step. We performed experiments with a leather with considerable meso-structure and even there shadowing and masking only occurred in the grazing angles. Thus, we did not include these terms as it would slow down the heightfield optimization considerably.

For our tests with a synthetic dataset, carried out on a Q6600 quad-core with a Geforce 8800 GTX, the BRDF reconstruction needs about 20 min, the estimation of the weight map takes 5-10 min and about 1:15h is required for the first heightfield optimization. In later iterations, the heightfield estimation is faster, requiring about 10-20 min as better initial values are already available. About 5 iterations of these steps are performed to get the first estimated heightfield and SVBRDF, requiring thus about 6 hours. The calculation of interreflections requires about 40 min and is iterated 3 times with the reconstruction. As the reconstruction is faster for these later iterations, the whole process requires about 13 hours.

5.6 Conclusion

We presented a new algorithm, which combines multi-view and photometric stereo to directly recover heightfields without any intermediate normal integration steps and takes interreflections into account. Our evaluation on synthetic datasets showed that our algorithm is able to reconstruct the surface geometry considerably better than purely photometric techniques and that the removal of indirect light results in much better estimates of the spatially varying BRDF. Our results show, that for materials like the tested leathers and plastics the reconstruction of a accurate geometry and SVBRDF is possible, resulting in a quite faithful reproduction of the materials characteristic appearance, which is of paramount importance in the context of material rendering. However, for materials with more complex reflectance behavior, which would require elaborate models with more parameters, the resulting optimization problem would become even more challenging. For this reason, in Chapter 10, we still utilize a heightfield but instead of a BRDF model employ a BTF projected onto this heightfield for our work on material interpolation.

CHAPTER 6

PARAFAC BASED BRDF REPRESENTATION

In the previous chapter, we have used analytical BRDFs to describe the reflectance behavior of measured material samples. The resulting optimization problem was very challenging and prone to local minima, requiring the use of global optimization algorithms such as the employed genetic algorithm. Especially, when a very accurate representation even of fine details is desired, the use of analytical models becomes very challenging. For example in Figure 6.1, a measured material and a fitted Cook-Torrance model are compared. Though the model is able to reproduce the overall shape of the BRDF, several differences remain. The width and intensity of the lobe are chosen correctly, but the actual shape of the lobe, which depends on the microscopic roughness, does not fit to the original material, being too wide at the top and not wide enough at the bottom. This results in reflections which look too sharp. The fall-off of the diffuse part for very low light directions is not modeled correctly, resulting in a flat looking sphere. Furthermore, the Fresnel effect is not reproduced exactly and thus the highlights at grazing angles are not bright enough. Even though these effects seem to be only small deviations, each one is still perceptually relevant and all have to be considered to obtain a faithful rendering. To reproduce all these effects correctly, a very complex model with a high number of parameters would be required. These, however, soon become very difficult to handle because fitting becomes difficult for such complex models.

In this section, we will therefore focus on a data-driven representation instead. By directly using a measured dataset for rendering, it is possible to represent a very large space of materials of nearly arbitrary complexity very faithfully. However, in this case, it is necessary to find a compact, yet accurate representation which also allows for fast rendering. Several compression techniques have been proposed (e.g. [Fou95, KM99, MAA01, SBLD03, LRR04, SZC*07, SKB10, BÖK11]). For good compression ratios, it is important to use a technique which is able to exploit redundancies in the data as well as possible. However, it is equally important to use a similarity measure which takes into account that all parts of the BRDF are perceptually relevant. When, for example, an L^2 distance is used, often the lobe is

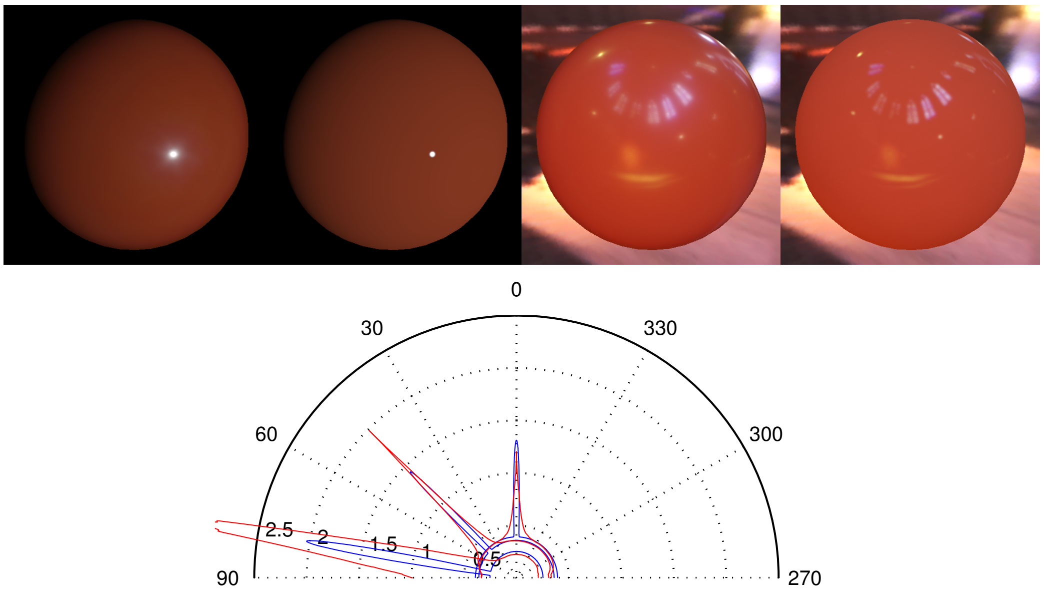
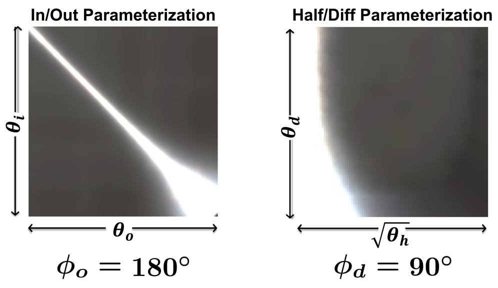


Figure 6.1: *Result of fitting a Cook-Torrance model (right, blue) to the material* `red phenolic` *from the isotropic MERL database (left, red). (The fourth root was applied to the plot, which shows the BRDFs for* $\theta_i = 0°, 45°, 80°$*.)*

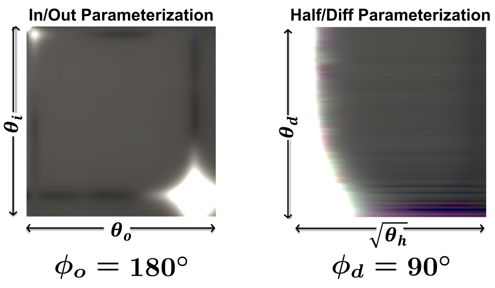
reproduced well but the much smaller diffuse parts are not represented correctly. Representations based on other distance measures have been proposed, such as the L^2 error in the logarithmic space [MAA01] or an error function based on the Kullback–Leibler divergence [LRR04].

In this chapter, we will show that perceptually superior results can be obtained by instead using a PARAFAC decomposition with a relative squared error as distance measure. The PARAFAC decomposition of the BRDF tensor achieves very good compression ratios as it can exploit correlations along all tensor modes. The relative squared error allows for good approximations of all parts of the BRDF at the same time, while it still can be minimized easily by choosing appropriate weights during the computation of the PARAFAC decomposition.

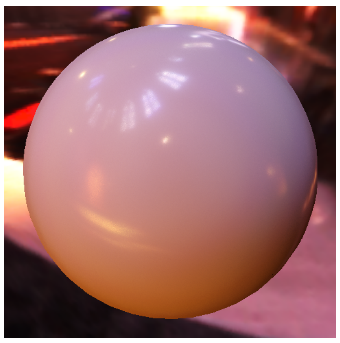
The work presented in this chapter has already been published as part of the technical report "A compact and editable representation for measured BRDFs", by Roland Ruiters and Reinhard Klein (Technical Report CG-2010-1, University of Bonn, December 2010) and was also a part of the EG tutorial "Tensor Approximation in Visualization and Computer Graphics" by Renato Pajarola, Susanne K. Suter, and Roland Ruiters (Eurographics Tutorials, number t6, May 2013).



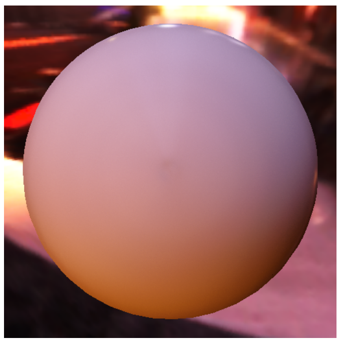
(a) Input data



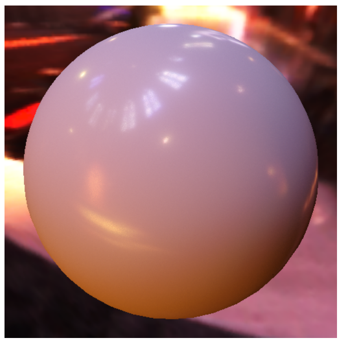
(b) PARAFAC approximation



(c) Original



(d) In/Out



(e) Half/Diff

Figure 6.2: *Comparison of the effect of different parameterizations. We show a slice through a BRDF dataset and the resulting approximation via a 6-component PARAFAC (with relative L^2 error). In the lower row, the resulting renderings are depicted.*

6.1 Representation

The canonical parameterization of a BRDF $\rho(\theta_i, \phi_i, \theta_o, \phi_o, c)$, via the incoming angle (θ_i, ϕ_i) and the outgoing angle (θ_o, ϕ_o) is useful for rendering, but not well-suited for factorization based approaches. Instead, we use a parameterization via the halfway vector (ϕ_h, θ_h) and a difference vector (ϕ_d, θ_d) as proposed in [Rus98], which aligns features of typical BRDFs along the axes of the coordinate system. In Figure 6.2, we show an example for the difference between the two parameterizations. When utilizing the In/Out parameterization, the highlight results in a diagonal structure in the parameter space. This is a disadvantageous case for factorization based compression techniques. These techniques represent the data as a sum of outer products and diagonal structures can only be created by summing up a large number of terms. In contrast, when using a half/diff parameterization, the highlight is aligned with the coordinate axis and thus the factorization results in a considerably better approximation. Since we only consider isotropic BRDFs, this representation has the additional advantage that these BRDFs can be represented via $(\theta_h, \theta_d, \phi_d)$ only. Furthermore, reciprocity can be enforced by storing only samples for $\phi_d \in [0, \pi)$ and then setting $\rho(\theta_h, \theta_d, \phi_d + \pi, c) = \rho(\theta_d, \theta_h, \phi_d, c)$. As suggested in [MPBM03a], we increase the resolution of θ_h near the lobe by parameterizing the BRDF over $\sqrt{\theta_h}$ instead of θ_h.

Since the function ρ depends on four parameters, a tabulated representation can be stored in a mode-4 tensor $\mathcal{B} \in \mathbb{R}^{n_{\theta_h} \times n_{\theta_d} \times n_{\phi_d} \times n_c}$. By using a tensor decomposition, it is then possible to take advantage of the fact that the parameterization via $(\theta_h, \theta_d, \phi_d)$ leads to a very regular structure along all of the tensor's modes.

As discussed in Chapter 4, a problem of these techniques is the slow decompression. This is especially true for the Tucker decomposition. Here, the dense core tensor has to be multiplied with all factor matrices to reconstruct a single entry of the tensor. This requires evaluating a sum with many terms. Therefore, this representation is not well-suited for applications which need fast random access into the tensor, such as real-time rendering of BRDFs. We thus decided to use the PARAFAC decomposition of the tensor $\mathcal{B}$ instead. The PARAFAC decomposition represents a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_m}$ as a sum of outer products of vectors $\mathbf{v}_i^{(j)}$: $\mathcal{T} \approx \sum_{i=1}^{k} \mathbf{v}_i^{(1)} \circ \cdots \circ \mathbf{v}_i^{(m)}$. This representation can thus also be considered as a Tucker decomposition with a diagonal core tensor and non-orthogonal factor matrices. Since the core tensor only has entries on its diagonal, calculating an entry of the tensor using this representation requires only $O(k)$ operations. As we will show in the next section, very good approximations of BRDFs can already be achieved with a small value such as $k = 8$, which allows both for fast rendering and a compact representation. Therefore, more complex techniques, such as the sparse decompositions we used in Chapter 4, are not needed in this case and would introduce additional problems such as a more complicated interpolation.

In contrast to the tabulated representation we used during the compression, we need a continuous representation during rendering. Here, the use of a PARAFAC decomposition has the additional advantage, that multi-dimensional interpolation within the tabulated dataset can be performed by interpolating within the one-dimensional vectors $\mathbf{v}_i^{(j)}$. This representation is the following sum of functions

$$\rho(\theta_h, \theta_d, \phi_d, c) \approx \sum_{i=1}^{k} f_i^{(1)}(\sqrt{\theta_h}) f_i^{(2)}(\theta_d) f_i^{(3)}(\phi_d), f_i^{(4)}(c) \tag{6.1}$$

in which $f_i^{(j)}$ is the piece-wise linear function resulting from linear interpolation between entries of the vector $\mathbf{v}_i^{(j)}$. This sum of separable functions is described in more detail in [BGM09], where other representations for the individual functions are considered. We will only use piece-wise linear functions in this work, as this can be regarded as the continuous analogue to the PARAFAC. However, other representations of the functions, e.g. polynomials, and interpolation techniques are in principle possible. k is also called *separation rank* in this case.

Usually, a PARAFAC decomposition of a tensor is obtained via an *alternating least squares (ALS)* optimization. However, these minimize the L^2 difference between the tensor and its approximation, which is not well-suited for the compression of BRDFs as BRDFs often have a very high dynamic range, with the lobe being several orders of magnitude brighter than the diffuse parts of the BRDF.

Two different approaches can be taken to achieve good fits for both the diffuse parts and the lobe of a material. On the one hand, a non-linear operation can be applied to reduce the dynamic range of the data. The factorization is then performed on the transformed data. For example, in [MPBM03a] the data was compressed by calculating the SVD in log-space. Similarly in [BÖK11] a log transform was utilized prior to a tensor factorization. However, when using this approach, not the original data are compressed but the representation in this different space and thus during rendering the mapping has to be reversed after the decompression, increasing rendering costs. The resulting decomposition is also no longer linear in the input data, which is important if linear transformations are to be applied. For example, in [SZC*07], the linearity of the a tensor decomposition of database of BRDFs was utilized for PRT computations. Similarly, an importance sampling approach such as the one suggested in [LRR04] might also be possible with the tensor decomposition but would require a linear decomposition of the data.

The other approach is to use a different distance measure during the optimization. This, however, requires solving a non-linear optimization problem to find the decomposition. Considering the high number of unknowns, finding a good solution efficiently is often a difficult problem. In contrast, a sum of squares objective function has the advantage that the optimization problems that have to be solved during the tensor decomposition are least squares problems for which very efficient algorithms exist.

We therefore measure the difference D between the original tensor $\mathcal{B}$ and its approximation $\mathcal{B}'$ via the following function:

$$D(\mathcal{B},\mathcal{B}') = \sum_{i_1,i_2,i_3,i_4} w_{i_1,i_2,i_3,i_4} \frac{(b_{i_1,i_2,i_3,i_4} - b'_{i_1,i_2,i_3,i_4})^2}{\max(b_{i_1,i_2,i_3,i_4}, \varepsilon)}.$$

We use a tensor $\mathcal{W}$ containing additional per element weights w_{i_1,i_2,i_3,i_4}. ε is a small constant which is necessary to avoid divisions by zero if the original dataset contains very small values. It should be chosen at about the size of the measurement noise for very dark materials.

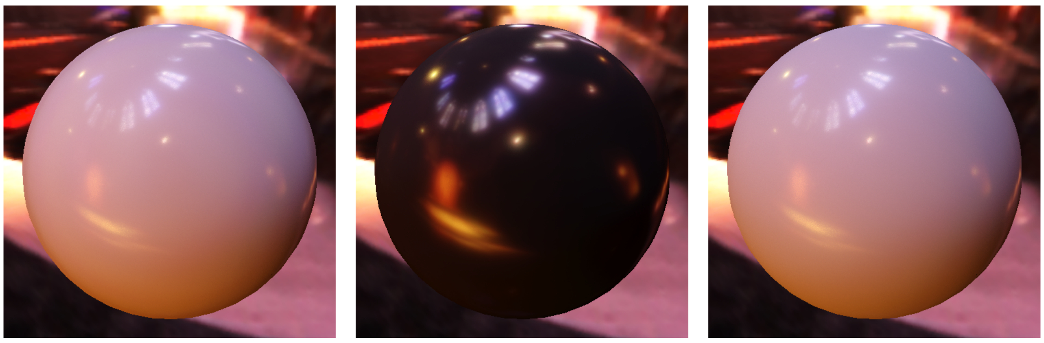

(a) Original **(b)** Absolute L^2 error **(c)** Relative L^2 error

Figure 6.3: *Comparison of absolute and relative L^2 error. The dataset was compressed with 6-components. No additional weights apart from the reciprocal sample value were applied.*

This distance measure thus consists of the squared distance in relation to the value of the original dataset. Therefore, the dynamic range does no longer influence the fitting results and each entry of the tensor has a similar importance in the final fitting result. As Figure 6.3 shows, this metric provides perceptually much better approximations for BRDFs with a high dynamic range. This choice is motivated by the fact that the human perception depends on relative brightness. Weber's Law states that the just noticeable difference between a stimulus and a background signal depends on the ratio of the difference to the intensity of the background and not the absolute difference (see for example [Boo02]).

This choice of distance function has the important advantage that by combining the tensor $\mathcal{W}$ and the term in the denominator into a new weight tensor $\mathcal{W}'$ the problem can be formulated as a weighted least squares tensor approximation for which existing and reasonably efficient algorithms can be used. The weight tensor $\mathcal{W}$ serves two purposes. On the one hand, we use it to compensate for the irregular sampling resulting from the use of a square root parameterization for θ_h. On the other hand, we choose $\mathcal{W}$ proportional to $\cos\theta_i \cos\theta_o$. In our opinion, these additional weights did improve the perceptual quality of the resulting BRDFs. This is probably the case because the datasets contain the most noise in regions where both θ_i and θ_o are small. However, a more throughout evaluation in a perceptual experiment might be an interesting avenue of further research.

One other important issue that has to be taken into account during the optimization is the fact that sampling the parameters $\theta_h, \theta_d, \phi_d$ on a regular grid, as necessary to store the resulting values in a tensor, results in many samples in which either θ_i or θ_o is larger than $90°$. During rendering, samples with $\theta_o > 90°$ should never be visible and for samples with $\theta_i > 90°$ the light source should alway be occluded. Selecting $\rho(\theta_h, \theta_d, \phi_d, c) = 0$ for these cases would be a straightforward choice, since at least for $\theta_i > 90°$ the corresponding pixels would appear black during rendering. However, during compression this causes considerable problems, since the boundary between valid and invalid samples runs diagonally through the parameter space and thus cannot be represented well as a sum of outer products. Instead, we treat these cases as missing values and allow the optimization algorithm to impute values which allow for the best possible compression ratio.

We used the PARAFAC implementation from the N-way Toolbox for MATLAB [AB00], which directly supports solving the weighted least squares problem with missing values. We furthermore constrained our decomposition to be non-negative, as negative BRDFs could cause problems during rendering.

6.2 Evaluation

To evaluate our decomposition technique, we used 24 samples from the isotropic MERL BRDF database (see Section 2.1.2), which were selected to be representative for a quite large range of materials including very specular, glossy and diffuse materials. These were already stored in a $\sqrt{\theta_h}, \theta_d, \phi_d$ parameterization with a $1°$ sampling resolution. Since 3 color channels are used, one BRDF thus contains $90 \times 90 \times 180 \times 3 = 4,374,000$ values. In the following, we will give all sizes for BRDFs under the assumption that the values are stored as double precision floats since we performed most of our calculations in MATLAB using double values. During rendering usually float or even half precision floats would certainly be sufficient, reducing the sizes correspondingly. Stored as doubles, one BRDF thus requires uncompressed about 33 MB.

To evaluate the performance, we compressed the same materials with several other techniques. Since these techniques use several different distance measures to compress the BRDFs, we consider directly calculating the differences between the compressed and original BRDF not a fair comparison. Instead, we decided to use the approach suggested in [NDM06] and rendered the materials under environment illumination (Grace Cathedral, courtesy by Paul Debevec). To compare the resulting images, we then used a perceptual image difference, the *Structural Similarity Index* [WBS*04].

We computed our PARFAC decompositions with $k = 8$ terms, which was sufficient to reproduce all materials with a good quality. For each material, $(90 + 90 + 180 + 3) * 8 = 2904$ doubles and thus about 23 KB are needed. On a computer with a Q6600 CPU and 4 GB RAM, the factorization requires about 15 minutes per

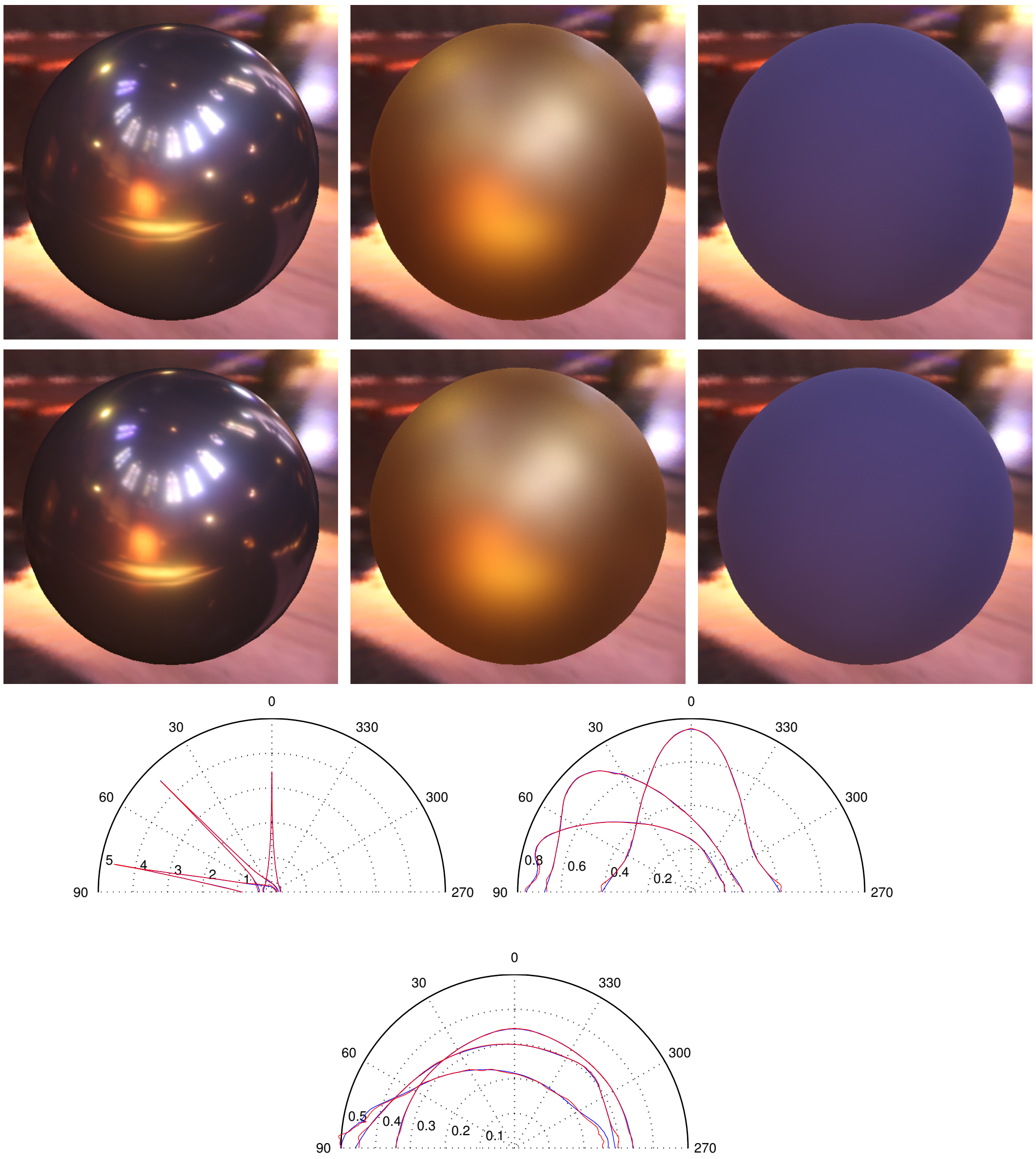


Figure 6.4: *Comparison of renderings of the uncompressed (top,red) and PARAFAC compressed BRDFs (middle,blue) and the corresponding plots (bottom) for the materials* `aluminium`*,* `gold metallic paint` *and* `blue fabric`*. (The fourth root was applied to the plots, which show the BRDFs for $\theta_i = 0°, 45°, 80°$.)*

material when using 100 ALS iterations. Apart from the results obtained with the relative squared error, we also include results obtained with exactly the same decomposition but with a simple L^2 difference.

For the comparison, we used the Cook-Torrance model [CT82] with the Beckmann normal distribution function, the Schlick approximation for the Fresnel term [Sch94] and one lobe, since this model achieved good results in [NDM05]. We performed the fitting similar to the approach described in [NDM05], but we used the same relative squared error as for the data-driven compression since the use of similar distance functions allows for a more direct comparison of the results.

We also compared against both the homomorphic factorization from [MAA01] and the non-negative matrix factorization [LRR04]. As suggested in [MAA01] we took advantage of the isotropy of our BRDFs by storing for each color channel the two rotationally symmetric results as 1D functions with a resolution of 256 entries each, which results in a size of 12 KB. Though, higher resolutions for these functions would be possible, this would not significantly improve the result. For many materials, the approach achieved good results. However, effects which depend on both view and light direction, like the Fresnel effect, cannot be reproduced because of the selected parameterization. Adding further terms to the factorization might improve upon this.

We decided, in contrast to [LRR04], to use the same parameterization for the NMF based factorization as for our PARAFAC decomposition, to not include the cosine term and to consider invalid view/light directions as missing values. These decisions in the original paper were primarily motivated by the need for importance sampling which we do not consider in this chapter and we obtained better results this way. Still, we found that performing two independent factorizations causes certain problems. During the second factorization, a different error is minimized as in the first one. Especially when using missing values, this can cause problems as during the second factorization all entries are considered to be of equal importance. A weighted NMF algorithm might help with this. We used the same resolution as for our factorization. During the first factorization, along the ϕ_d dimension, we kept two components and during the second factorization we kept three components, resulting in nearly exactly the same data size. For three representative materials, we show renderings of the measured original data and of our compressed representations in Figure 6.4. Additionally, polar plots of the BRDFs in the plane of specular reflection are shown for three representative incoming light directions.

In Figure 6.5, we compare the resulting image differences. As can be seen in the plot, our PARAFAC based representation achieves the best quality for all materials. Furthermore, the quality is very consistent without any large outliers. The graph

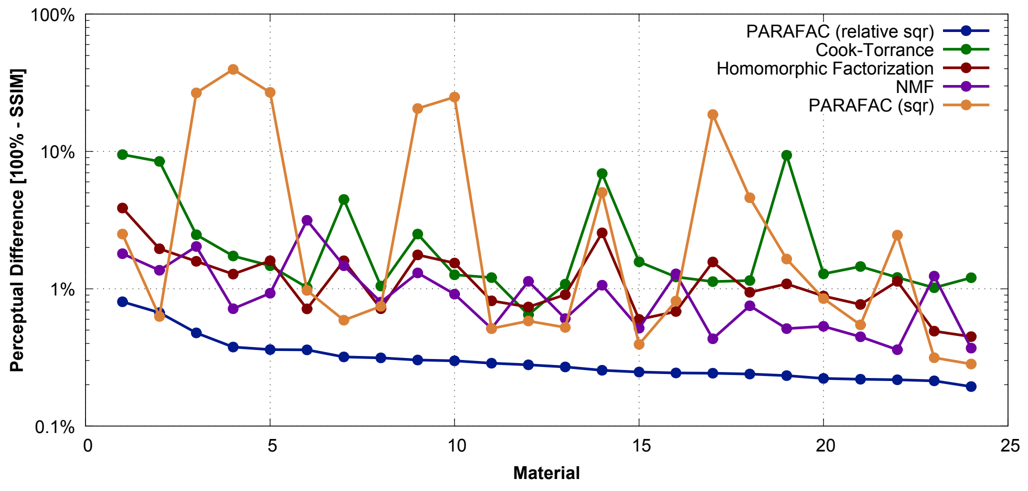


Figure 6.5: *Comparison of several BRDF representations (materials sorted by error of our PARAFAC based representation)*

also shows the importance of using a relative error function, since the PARAFAC decomposition with L^2 distance often gives quite bad results.

6.3 Conclusion

In this chapter, we have shown that by minimizing a relative squared error during a PARAFAC decomposition it is possible to store an isotropic BRDF in a half-angle/difference-angle parameterization very compactly and at the same time accurately. This representation is well-suited for efficient real-time rendering. So far, we only considered homogeneous BRDFs and assumed that a densely sampled measurement on a regular grid and a high angular resolution is available.

After the development of the representation described in this chapter, a similar approach was published in [BÖK11]. In contrast to the PARAFAC factorization employed by us, they use repeated Tucker factorizations. However, since they always use a core tensor of size 1, this results in effect in a greedy algorithm to compute a PARAFAC factorization. Furthermore, in contrast to the relative error metric employed by us, they use a logarithmic transformation prior to the compression to reduce the dynamic range. This additional transformation might be a disadvantage in cases where the linearity of the resulting representation is required.

It would be an interesting question, whether our PARAFAC representation could be used for importance sampling in a similar manner to the technique described in [LRR04]. However, this would require the use of a different parameterization which might reduce the achievable compression ratio.

CHAPTER 7

Surface Reflectance from Sparse and Irregular Samples

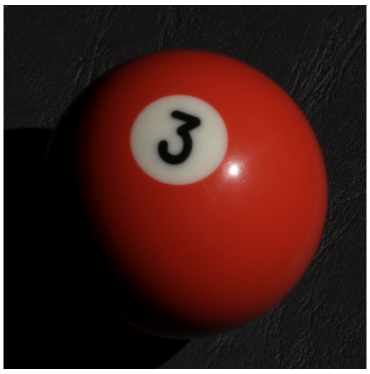
In the previous chapter, we have seen that it is possible to store isotropic BRDFs very compactly and accurately using a PARAFAC based representation. In this chapter, we will extend this approach to the acquisition of spatially varying BRDFs on complicated geometries. In contrast to the densely and regularly sampled datasets from the previous chapter, in this case the input data is considerably more challenging. Again we will assume, that the reflectance of the objects has been acquired with a device such as our Multi-View Dome (Section 2.1.3) or our Mobile Dome (Section 2.1.4). These provide a large number of pictures from several viewpoints and with varying light directions and additionally the geometry of the object. A fundamental problem is that the number of view and light directions is limited, either by the capture setup or by the available measuring time and storage space. Therefore, only a coarse sampling of the angular domain is possible. Furthermore, the sampling is irregular for curved geometries since the view and light directions are different at each point as the local coordinate systems vary over the object. Additionally, due to occlusions and shadowing, it might for some surface points happen that there are holes in the angular sampling which have to be filled.

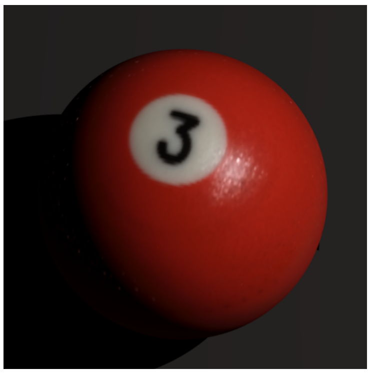
Therefore, a scattered data interpolation problem has to be solved to find a seven-dimensional function $\rho(x, y, \theta_i, \phi_i, \theta_o, \phi_o, c)$ (two spatial dimensions (x, y), four angular dimensions (θ_i, ϕ_i) and (θ_o, ϕ_o), and wavelength/color c) from a set of irregularly sampled function values. In this general form, the problem is obviously under-determined. To make it tractable, additional assumptions have to be made, which impose further constraints on ρ. A common assumption is isotropy, which specifies a symmetry in the angular dimensions, eliminating one of them completely. As we have discussed in Chapter 5, one possibility to tackle this problem is the use of analytical BRDF models. Since a model specifies a large amount of prior knowledge about the structure of BRDFs, these are able to provide reasonable results even for very small numbers of samples. Furthermore, analytical models also provide a very compact representation. However, this approach resulted in a very difficult non-linear optimization problem, especially when an exact reproduction of a given BRDF is desired. In contrast, as we have shown in Chapter 6, a data-driven representation is in principle capable of very accurately and compactly representing BRDFs. Several data-driven techniques (e.g. [FKIS02, MBK05, SWRK11]) to capture the reflectance of objects have already been proposed. Here, the reflectance is represented either as a tabulated *spatially varying BRDF* (SVBRDF) or as a *Bidirectional Texture Function* (BTF). This way, nearly arbitrary materials can be represented without the restrictions imposed by analytical models. To obtain such a tabulated representation, the available scattered data points are usually resampled to a regular grid and an interpolation has to be performed to fill holes in the dataset. Both the use of a tabulated representation itself and the employed interpolation and compression techniques are based on the assumption that the reflectance is a locally smooth function.

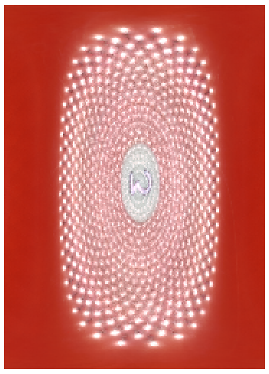
Usually, it is additionally assumed that the material exhibits a strong self-similarity in the spatial domain, which means that the reflectance behavior is similar at different positions. More formally, this can be expressed as a constraint on the separation rank C [BGM09] of ρ. That is, the reflectance can be approximated as a sum of C separable functions

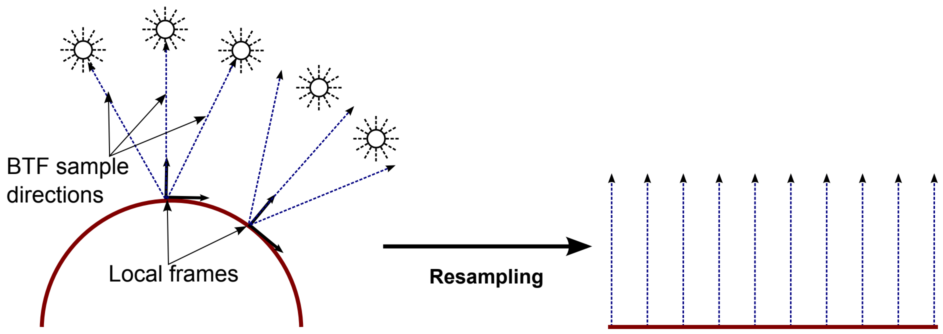
$$\rho(x, y, \theta_i, \phi_i, \theta_o, \phi_o, c) \approx \sum_{j=1}^{C} \rho_1^{(j)}(x, y)\rho_2^{(j)}(\theta_i, \phi_i, \theta_o, \phi_o, c).$$

Techniques based on analytical models often exploit this property by assuming that the material is composed of several basis BRDFs, as we did in Chapter 5. In contrast, data-driven techniques can exploit this fact during interpolation to perform hole-filling (e.g. [SWRK11] and for the final compression step. This step is usually based on a matrix or tensor factorization which is applied to the dense, interpolated dataset. Unfortunately, for specular materials this process does not scale well. Here, a dense measurement and a high angular resolution in the regular sampling would be required to faithfully represent the highlights. However, working with such a dataset is no longer feasible due to the data size. For example, for the measurements from the MERL BRDF database (see Section 2.1.2) a 1° angular sampling, resulting in $90 \times 90 \times 180$ values, is used to store a single, homogeneous, tabulated, isotropic, high-quality BRDF. A dataset using this angular resolution and a spatial resolution of 256×256 would require about 1TB. In Figure 7.1c, the problems both due to insufficient angular sampling during the measurement and coarse angular resolution of the resampled dataset are illustrated for a BTF measurement.


(a) Photograph


(b) BTF


(c) Top-View



(d) Illustration of the BTF resampling

Figure 7.1: *Problems due to insufficient sampling and angular resolution. These images show a photograph (a) and a BTF rendering (b) of a specular billiard ball. The drawing (d) illustrates the necessary resampling process. The hemisphere of view and light directions of the BTF is parameterized within the local coordinate frames. Therefore, at different positions on the surface, samples for different light and view directions are stored (blue arrows). However, the measured directions (which are determined by the positions of the light sources/cameras) do not correspond exactly to the stored sample directions, and thus it is necessary to interpolate the measurements during the resampling to create the idealized BTF with parallel view and light directions (illustrated on the right). Unfortunately, for some points on the surface, the directions corresponding to a highlight are not observed and thus the interpolation cannot recreate these highlights. This problem is shown in image (c) which depicts one slice from the BTF, with both light and view direction exactly perpendicular to the surface. This slice should be uniformly white as the material is mostly homogeneous, but instead for many points on the surface the highlight has not been captured. Furthermore, even at points where the highlight was observed, it is blurred as the coarse angular sampling of the BTF does not allow for a faithful representation of the sharp highlight.*

We will therefore generalize the BRDF representation from the previous chapter (Equation 6.1) to the case of spatially varying BRDF with irregular input samples. We avoid to store the whole dataset on a regular grid, eliminating the consequential interpolation and resampling step and memory requirements. For this, we directly fit a representation of a SVBRDF as a sum of separable functions to irregular reflectance samples. Since our representation is very compact and can also be used directly for rendering, no additional compression steps are necessary. Using a sum of separable functions inherently exploits the fact that typical reflectance datasets exhibit a very low separation rank, both, due to spatial self-similarity and due to the fact that BRDFs have a very regular structure when represented in a suitably parameterized space [Rus98]. Furthermore, we integrate regularization which enforces local smoothness and spatial self-similarity both for filling holes in the dataset and for allowing a reconstruction at a high angular resolution. Our approach is a data-driven technique which replaces the hard constraints imposed by an analytical model with a more flexible regularization. This way, we avoid the selection of a suitable model and allow for good reconstructions from rather sparse samplings, but, as the number of available samples increases, the amount of regularization can be continuously reduced to obtain accurate approximations.

This chapter corresponds to the paper "Data Driven Surface Reflectance from Sparse and Irregular Samples" by Roland Ruiters, Christopher Schwartz, and Reinhard Klein published in *Computer Graphics Forum (Proc. of Eurographics)*, 31(2):315–324, May 2012. Since the publication of the original paper, we have added an additional regularization, described in Section 7.4. This increased the robustness of the fitting and helped to prevent problems due to regularization bias in the original datasets. Figure 7.8c and 7.8h therefore show novel results, computed with the improved regularization.

7.1 Overview

We consider the problem of finding a compact approximation of an unknown function $F(\mathbf{x}) = F(x_1, \ldots, x_D)$ of D variables. Here, the function is described by a set of N samples $s^{(n)} = F(\mathbf{p}^{(n)})$, drawn at the irregular positions $\mathbf{p}^{(n)} = (p_1^{(n)}, \ldots, p_D^{(n)})$ for $n \in \{1, \ldots, N\}$.

For this approximation, we use a sum of C separable functions, where C is called the separation rank. This way, it is possible to avoid the curse of dimensionality and represent even high-dimensional functions very compactly if they have a very regular structure in the chosen parameterization and thus a low separation rank. The function F is then approximated in the following way:

$$F(\mathbf{x}) \approx \tilde{F}(\mathbf{x}) = \sum_{c=1}^{C} f^{(c)}(\mathbf{x}) = \sum_{c=1}^{C} \prod_{d=1}^{D} f^{(c,d)}(x_d). \tag{7.1}$$

The use of this representation for multivariate regression has recently been investigated in the context of machine learning. In [BGM09], a general framework is introduced. An alternating least squares optimization algorithm for functions representable within a linear function space is discussed. Although a regularization approach is described, it is performed on each of the separated functions independently instead of the final sum of products. In contrast, in [Gar10] the authors discuss the use of global regularization. However, due to their choice of error function, they have to solve a non-linear optimization problem for each of the iterative updates. As was discussed in the previous chapter, this representation is closely related to the PARAFAC decomposition of a tensor into a sum of outer products.

We again use the parameterization via a half-angle and a difference-angle [Rus98] discussed in the previous chapter to align typical features of BRDFs with the axes of the coordinate system to get a representation with a low separation rank. In addition, we limit ourselves to isotropic BRDFs, which can be represented in this parameterization via $(\theta_h, \theta_d, \phi_d)$ only, reducing the dimensionality of the problem by one. In contrast to the homogenous BRDF we approximated in Equation (6.1), we now additionally have to consider the position p on the surface. We thus have to find an approximation for the $D = 5$ dimensional function $\tilde{F}(\mathbf{x}) \approx \rho(p, \theta_h, \phi_h, \phi_d, c)$ from the set of measured samples.

Finding such a representation for a given set of samples $s^{(n)}$ can be achieved by finding one-dimensional functions $f^{(c,d)}$, which minimize an energy functional describing the distance between the model and the samples. However, for arbitrary functions $f^{(c,d)}$ the problem is under-determined. Therefore, we need an additional regularization term. We use for each of the D parameters a linear operator $R^{(d)}$ describing the regularization within this dimension. For the spatial dimension, we use an AppProp-inspired regularization described in Section 7.2.3. For the remaining dimensions, we use a weighted second derivative $R^{(d)} = \partial_d^2$ as regularization. Minimizing this operator enforces local smoothness, a property commonly assumed for BRDFs. However, other operators are applicable as well.

Both for the choice of the distance measure used to describe the quality of a given approximation and for the regularization it is very important to recognize that BRDFs exhibit a very high dynamic range. The intensity of the darkest parts of a BRDF is easily several thousand times lower than in the highlight.

Nonetheless, correctly approximating these parts of a material is perceptually as important as representing the highlight correctly. In the final rendering, the highlight often occupies just a few pixels, whereas the darker diffuse parts of the material are visible everywhere else. Additionally, we need to reduce the amount of regularization in the highlight, compared to diffuse parts, since the function exhibits a much larger curvature here than in diffuse parts. Thus, we measure the approximation error relative to the sample value and the regularization relative to the function value at each position. This results in the following energy functional describing the quality of a given approximation:

$$E(f^{(1,1)},\dots,f^{(C,D)}) = \sum_{n=1}^{N}\left(v^{(n)}\frac{s^{(n)}-\tilde{F}(\mathbf{p}^{(n)})}{s^{(n)}}\right)^2 + \sum_{d=1}^{D}\lambda_d\int_{\mathcal{V}}\left(w(\mathbf{x})\frac{R^{(d)}\tilde{F}(\mathbf{x})}{\tilde{F}(\mathbf{x})}\right)^2 d\mathbf{x} \tag{7.2}$$

Here, $v^{(n)}$ represents a per-sample weight term and $w(\mathbf{x})$ is a function controlling the strength of the regularization in dependence on the coordinate $\mathbf{x}$. For each of the dimensions, λ_d controls the strength of the regularization. For simplicity of notation, we will include λ_d into the operators $R^{(d)}$ from here on. $\mathcal{V}$ is the subspace over which the function is defined.

7.2 Implementation

As suggested in [MPBM03a], we increase the resolution of θ_h near the lobe by parameterizing the BRDF over $\sqrt{\theta_h}$ instead of θ_h. Instead of using two coordinates for the spatial dimension, we combine them into a single coordinate p, since the two spatial dimensions usually do not exhibit a low separation rank and thus a large number of components would be needed to encode the complex spatial distribution of a material. This assumption can only be made, once the spatial dimension has been discretized. In the continuous setting we would have to use a bivariate function $f^{(c,d)}(x, y)$.

To practically solve the continuous optimization problem given in Equation (7.2), we represent each of the one-dimensional functions $f^{(c,d)}$ as a piece-wise linear function defined by the vectors

$$\mathbf{f}^{(c,d)} = \left(f_1^{(c,d)} = f^{(c,d)}(t_1^{(d)}),\dots,f_{M_d}^{(c,d)} = f^{(c,d)}(t_{M_d}^{(d)})\right),$$

which consist of the function values at M_d support points $t_i^{(d)}$. These support points induce a regular grid, $t_{\mathbf{i}} = \left(t_{i_1}^{(1)},\dots,t_{i_D}^{(D)}\right)$, where $\mathbf{i} \in I = [1 : M_1]\times\cdots\times[1 : M_D]$

is a multi-index. Evaluation of the function $\tilde{F}$ at each of these grid points, leads to a tensor $\tilde{\mathcal{F}}$, with $\tilde{f}_{\mathbf{i}} = \tilde{F}(t_{\mathbf{i}}) = \sum_{c=1}^{C} \prod_{d=1}^{D} f_{i_d}^{(c,d)}$ for all $\mathbf{i} \in I$. Furthermore, we approximate the differential operators $R^{(d)}$ using forward differences. Since the regularization only operates within one dimension, it can be applied to each of the mode-d fibers independently. This can be represented as a $M_d \times M_d$ regularization matrix $\mathbf{R}^{(d)}$, which, when multiplied with one mode-d fiber of the tensor $\mathcal{F}$, computes the regularization along dimension d for all entries of the fiber simultaneously. For the ϕ angles, we assume a wrap-around, and for the θ angles, a mirror symmetry to handle the boundary conditions was used for some of the datasets. The color channel is not regularized at all, and for the spatial dimension we use an AppProp-inspired regularization, which we are going to discuss in Section 7.2.3. The integral is finally replaced by a finite sum over the regularization evaluated at each of the grid points $t_{\mathbf{i}}$. This results in the following discretized version of the continuous functional:

$$\begin{aligned} E(\mathbf{f}^{(1,1)}, \ldots, \mathbf{f}^{(C,D)}) = & \sum_{n=1}^{N} \left(v^{(n)} \frac{s^{(n)} - \tilde{F}(\mathbf{p}^{(n)})}{s^{(n)}} \right)^2 \\ & + \sum_{d=1}^{D} ||\mathcal{W} \otimes (\mathcal{F} \times_d \mathbf{R}^{(d)}) \oslash \tilde{\mathcal{F}}||^2 \end{aligned} \tag{7.3}$$

Here, the tensor $\mathcal{W}$ represents the weights $w(\mathbf{x})$ evaluated at the grid positions $\mathbf{t}_{\mathbf{i}}$. The symbols $\otimes$ and $\oslash$ are used to denote element-wise multiplication and division. The weights are chosen as 0 outside of the valid space $\mathcal{V}$ to make sure that the regularization is only evaluated within the valid region. This is necessary, since the parameterization via half-angle and difference-angle is only defined on an irregularly shaped subspace. Therefore, the regularization is considered only for $\mathcal{V}$ and furthermore the samples also lie within this subspace, even though the approximation $\tilde{F}$ itself is defined over the full space.

To approach this optimization problem, we use *iteratively reweighted alternating least squares*. By keeping the divisor $\tilde{\mathcal{F}}$ in the regularization term constant, we can combine it with $\mathcal{W}$ to obtain new weights $\tilde{w}_{\mathbf{i}} = \frac{w_{\mathbf{i}}}{\tilde{f}_{\mathbf{i}}} \in \tilde{\mathcal{W}}$, simplifying the remaining problem considerably. Similarly, we define the weight $\tilde{v}^{(n)} = \frac{v^{(n)}}{s^{(n)}}$.We then use *alternating least squares* (ALS), iterating over the dimensions and successively updating each dimension l individually by keeping the functions $f^{(c,d)}$ constant for all other dimensions $d \neq l$. Once the ALS has converged, we update the weights $\tilde{\mathcal{W}}$ and iterate the whole process. Given noisy input data, better results might be obtained by also updating the weights $\tilde{v}^{(n)}$ using the function value from the current estimate instead of the original samples values. Otherwise the noise can result in overestimating the importance of small outliers. On the other hand, it is conceivable that this might reduce the stability and result in oscillations. We have not yet evaluated this and use the original sample values in our experiments.

7.2.1 Quadratic Subproblem

In the following, we will thus discuss how to solve the remaining subproblem under the assumption that the terms in the divisor are included in the weights, and that only the functions $f^{(c,l)}$ for the dimension l are to be updated while all other dimensions are considered constant.

Data Term

For a given sample point $\mathbf{p}^{(n)}$, evaluating the functions $f^{(c,d)}$ is in fact a simple linear interpolation between the values at two adjacent support points. In the following, we will use the index $j_d^{(n)}$ and the weight $\alpha_d^{(n)}$ to denote the parameters for this interpolation in dimension d. The evaluation of the model is hence given by:

$$\tilde{F}\left(\mathbf{p}^{(n)}\right) = \sum_{c=1}^{C} \prod_{d=1}^{D} (1 - \alpha_d^{(n)}) f_{j_d^{(n)}}^{(c,d)} + \alpha_d^{(n)} f_{j_d^{(n)}+1}^{(c,d)}$$

When only dimension l is updated, all factors but one in the product are constant and simply correspond to the evaluated function value $f^{(c,d)}(x_d)$. Therefore, the evaluation of the model is a linear operation:

$$\tilde{F}\left(\mathbf{p}^{(n)}\right) = \sum_{c=1}^{C} ((1 - \alpha_l^{(n)}) f_{j_l^{(n)}}^{(c,l)} + \alpha_l^{(n)} f_{j_l^{(n)}+1}^{(c,l)}) \underbrace{\prod_{d=1, d \neq l}^{D} f^{(c,d)}(x_d)}_{constant}$$

This operation can be expressed for all samples simultaneously via the product between a matrix $\mathbf{S}$ and a vector $\mathbf{f}^{(l)}$, which contains a concatenation of all $\mathbf{f}^{(c,l)}$. The matrix $\mathbf{S}$ is extremely sparse, containing only $2C$ non-zero entries in each row. Figure 7.2 illustrates this (under the assumption of sorted samples). When the samples $s^{(n)}$ are concatenated into one vector $\mathbf{s}$ and the sample weights $\tilde{v}^{(n)}$ are stored in a diagonal matrix $\mathbf{V}$, the data term is finally given as:

$$\left\| \mathbf{V}\mathbf{s} - \mathbf{V}\mathbf{S}\mathbf{f}^{(l)} \right\|^2$$

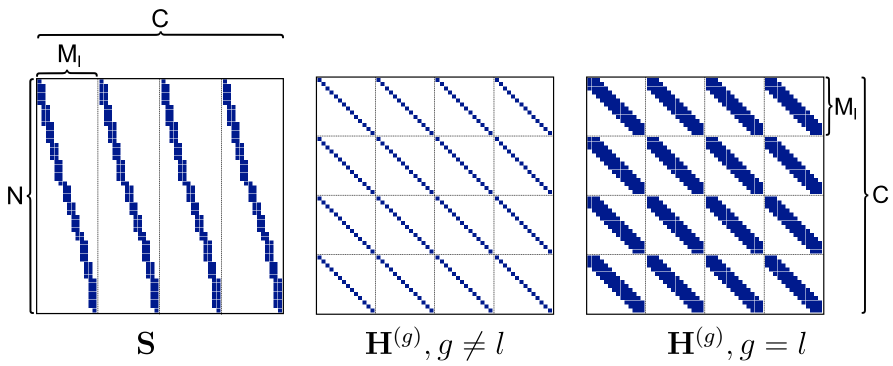


Figure 7.2: *Structure of the matrices* S *and* $\mathbf{H}^{(g)}$.

Regularization Term

The additional regularization term has to be considered for all dimensions $g \in \{1, \ldots, D\}$. Since evaluating it requires summing over all entries of $\tilde{\mathcal{W}} \otimes \left(\mathcal{F} \times_g \mathbf{R}^{(g)}\right)$, using this regularization with an arbitrary weight tensor $\mathcal{W}$ would be prohibitively costly. In this case, the computation time lies in $O(\prod_{d=1}^{D} M_d)$, which is not feasible for larger datasets. However, under the assumption that a factorized representation (with C_W components) $\tilde{\mathcal{W}}^2 \approx \sum_{c=1}^{C_W} \mathbf{w}^{(c,1)} \circ \cdots \circ \mathbf{w}^{(c,D)}$ is available, a much more efficient computation is possible. In the following, we will additionally use $\mathbf{W}^{(c,d)}$ to denote a diagonal matrix with the elements of $\mathbf{w}^{(c,d)}$ on the diagonal. In this case, when computing the regularization for dimension g, the following equality holds (a full derivation is given in Section 7.6):

$$\begin{aligned}
&\left\|\tilde{\mathcal{W}} \otimes \left(\mathcal{F} \times_g \mathbf{R}^{(g)}\right)\right\|^2 \\
&= \sum_{c_1=1}^{C} \sum_{c_2=1}^{C} \sum_{c_3=1}^{C_W} \left(\mathbf{f}^{(c_1,g)}\right)^T \left(\mathbf{R}^{(g)}\right)^T \mathbf{W}^{(c_3,g)} \mathbf{R}^{(g)} \mathbf{f}^{(c_2,g)} \\
&\prod_{d=1, d\neq g}^{D} (\mathbf{f}^{(c_1,d)})^T \mathbf{W}^{(c_3,d)} \mathbf{f}^{(c_2,d)} = \left(\mathbf{f}^{(l)}\right)^T \mathbf{H}^{(g)} \mathbf{f}^{(l)}
\end{aligned} \tag{7.4}$$

The regularization can thus be described by a quadratic form, with a sparse matrix $\mathbf{H}^{(g)}$. Here, we have to distinguish two cases. When $g \neq l$, the matrix $\mathbf{H}^{(g)}$ consists of $C \times C$ blocks, each containing weighted sums of the diagonal matrices $\mathbf{W}^{(c_3,d)}$ and thus the matrix $\mathbf{H}^{(g)}$ only has C non-zero entries in each row. Otherwise, each block contains weighted sums of $(\mathbf{R}^{(g)})^T \mathbf{W}^{(c_3,g)} \mathbf{R}^{(g)}$. Since we use a differential operator for $\mathbf{R}^{(g)}$, this matrix is also very sparse, as the regularization operator itself is a band matrix in this case and thus $(\mathbf{R}^{(g)})^T \mathbf{W}^{(c_3,g)} \mathbf{R}^{(g)}$ can be computed in $O\left(M_g\right)$. In Figure 7.2, the sparsity pattern is illustrated. The matrix $\mathbf{H}^{(g)}$ is symmetric because the equation is symmetric under exchange of c_1 and c_2 and each of the $C \times C$ blocks itself is also symmetric. Furthermore, since the quadratic form is equal to a non-negative number for each choice of $\mathbf{f}^{(l)}$, $\mathbf{H}^{(g)}$ is also positive semi-definite.

The matrix $\mathbf{H}^{(g)}$ can be computed efficiently, since the individual factors in the product can be computed independently for each dimension and then multiplied instead of computing the regularization for all $\prod_{d=1}^{D} M_d$ entries of the high-dimensional tensor. Thus, only $O\left(C^2 C_W \left(\sum_{d=1}^{D} M_d\right)\right)$ operations instead of $O\left(C^2 \prod_{d=1}^{D} M_d\right)$ operations are required.

Hence, the complete optimization problem has the following form:

$$\begin{aligned}
&\arg\min_{\mathbf{f}^{(l)}} \left\|\mathbf{V}\mathbf{S}\mathbf{f}^{(l)} - \mathbf{V}\mathbf{s}\right\|^2 + \sum_{g=1}^{D} \left(\mathbf{f}^{(l)}\right)^T \mathbf{H}^{(g)} \mathbf{f}^{(l)} \\
= &\arg\min_{\mathbf{f}^{(l)}} \left(\mathbf{f}^{(l)}\right)^T \left(\mathbf{S}^T\mathbf{V}^2\mathbf{S} + \sum_{g=1}^{D} \mathbf{H}^{(g)}\right) \mathbf{f}^{(l)} - 2\mathbf{s}^T\mathbf{V}^2\mathbf{S}\mathbf{f}^{(l)} + \mathbf{s}^T\mathbf{V}^2\mathbf{s}
\end{aligned} \tag{7.5}$$

This is a sparse, positive semi-definite quadratic programming problem since both $\mathbf{S}^T\mathbf{V}^2\mathbf{S}$ and $\mathbf{H}^{(g)}$ are positive semi-definite and sparse. In the absence of any further constraints, it can be solved efficiently with a direct sparse linear solver.

7.2.2 Weight Updates

In each iteration of the optimization, we have to update the weights $\tilde{\mathcal{W}}$ and possibly also $\tilde{v}^{(n)}$. We initialize $\tilde{v}^{(n)} = \frac{v^{(n)}}{\max(|s^{(n)}|,\varepsilon)}$ with the known sample values. If iterative updates are used for $\tilde{v}^{(n)}$, we update the weights during each iteration using the current estimate of $\tilde{\mathcal{F}}$. It is important to clamp the weights to a minimal value ε to avoid divisions by zero. Furthermore, it is possible to integrate a robust statistic into these weight updates to remove outliers. Currently, we simply set the weight to 0 for the 5% samples which have the largest error to remove outliers during the optimization, but a more sophisticated approach, such as e.g. Huber weights, could be employed.

In contrast to $\tilde{v}^{(n)}$, the update of $\tilde{\mathcal{W}}$ is not as straightforward. For the optimization, we need a factorized representation of $\tilde{\mathcal{W}}^2 = (\mathcal{W} \oslash \mathcal{F})^2$. Computing the update on the dense representation first and subsequently converting the result into a PARAFAC representation should be avoided, as this would require operating on all entries of the tensor, which does not scale to higher-dimensional datasets. Instead, we operate only on the factorized representations of $\mathcal{W}$ and $\mathcal{F}$.

Computing the product (and thus also the square) of two tensors in PARAFAC representation is straightforwardly possible by computing the element-wise products of all pairwise combinations of the components of the two tensors. Computing the element-wise inverse of $\mathcal{F}$, however, is a more difficult task. Here, we use ALS to solve the problem $\mathcal{F} \otimes \mathcal{F}^{-1} = \mathbf{1}$, where $\mathbf{1}$ is a tensor containing 1 in each entry. Since a factorized representation of $\mathcal{F}$ is already available, the ALS can be computed very efficiently. It is important, though, to enforce the non-negativity of the weights during this optimization, as otherwise $\mathbf{H}$ would no longer be positive semi-definite. Instead of clamping the tensor at ε prior to inversion, we add a penalty term for large values during the ALS, preventing instability for small tensor entries. This approach has the advantage that it works well for datasets with a high dynamic range since a relative error is optimized.

7.2.3 Spatial Regularization

So far, we have only considered derivative operators for the regularization to enforce local smoothness. A similar operator, such as the square of the gradient norm, could also be used for the spatial dimensions. However, this would result in smoothed and unsharp textures. On the other hand, a spatial regularization is quite useful since by exploiting the fact that many points on the surface have a similar material, it is possible to reconstruct BRDFs with a high angular resolution from a rather coarse sampling of view/light directions.

We therefore use an approach inspired by AppProp [AP08] to achieve a regularization which is not based on spatial proximity but instead explicitly models for all texels on the surface their pairwise similarity. For this, we use a symmetric, completely positive matrix $\mathbf{Z}$, where $z_{i,j}$ is a value between 0 and 1 describing the similarity between the texels i and j. Assuming, for simplicity of notation, the first mode of the tensor contains the spatial coordinates, we use the regularization term

$$\sum_{i=1}^{M_1} \sum_{j=1}^{M_1} z_{i,j}^2 \| \mathcal{W}_{i,:,\ldots,:} \otimes (\tilde{\mathcal{F}}_{i,:,\ldots,:} - \tilde{\mathcal{F}}_{j,:,\ldots,:}) \|^2. \tag{7.6}$$

This term could be described, like the other regularization operators, via a regularization matrix $\mathbf{R}^{(1)}$. This matrix would contain one row for each pair (i, j) with the i-th entry set to $z_{i,j}$ and the j-th entry set to $-z_{i,j}$. Unfortunately, this results in unfeasibly large matrices. The resulting regularization matrix $\mathbf{H}^{(1)}$ is a dense $CM_1 \times CM_1$ matrix. However, $\left(\mathbf{R}^{(1)}\right)^T \mathbf{R}^{(1)}$ is equal to $\mathbf{D} - \mathbf{Z}$, where $\mathbf{D}$ is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{M_1} z_{i,j}^2$. If a suitable factorization $\mathbf{Z} = \mathbf{M}^T\mathbf{M}$ is available, the resulting least squares problem (7.5) can be solved efficiently using the Woodbury matrix formula [Hag89]

$$(\mathbf{A} - \mathbf{U}\mathbf{V})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{I} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{A}^{-1}.$$

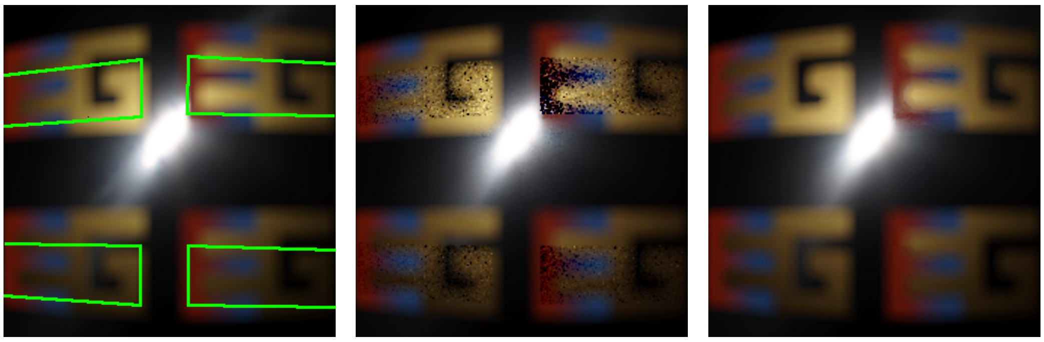
Setting $\mathbf{A} = \mathbf{D}$, $\mathbf{U} = \mathbf{M}^T$, $\mathbf{V} = \mathbf{M}$ we get the following identity, which is faster if $\mathbf{M}$ has a small number of rows:

$$\begin{aligned}\left(\left(\mathbf{R}^{(1)}\right)^T \mathbf{R}^{(1)}\right)^{-1} &= \left(\mathbf{D} - \mathbf{M}^T\mathbf{M}\right)^{-1} \\ &= \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{M}^T\left(\mathbf{I} - \mathbf{M}\mathbf{D}^{-1}\mathbf{M}^T\right)^{-1}\mathbf{M}\mathbf{D}^{-1}\end{aligned}$$

However, we also have to take the weight matrices $\mathbf{W}^{(c,1)}$ into account. For this, it is necessary to find an orthogonal $M_1 \times C_S$ matrix $\mathbf{U}$ spanning a sub-space with a low number C_S of dimensions, in which the matrices $\mathbf{W}^{(i,1)}\mathbf{M}$ can be approximated well for all $c \in 1 \ldots, C_W$. Once this orthogonal basis is available, the matrix $\mathbf{H}$ can be approximated as $\tilde{\mathbf{U}}\mathbf{A}\tilde{\mathbf{U}}^T + \mathbf{B}$, where $\tilde{\mathbf{U}}$ is a block-diagonal matrix with C blocks, each containing $\mathbf{U}$, $\mathbf{A}$ is a much smaller $CC_S \times CC_S$ dense matrix, which can be inverted directly, and $\mathbf{B}$ is a sparse matrix containing only diagonal entries in each of the $C \times C$ blocks.

To obtain the matrix $\mathbf{Z}$, we use a technique very similar to the one proposed in [AP08]. For a randomly selected subset of the texels we compute the distance to all other texels. We resample the irregular input data at each texel into a common regular sampling and then compute the mean of the regular samples. This way, we avoid a bias introduced by an uneven sampling. Then, we use the difference between these means as distance measure. However, in the future, a more sophisticated distance measure could be used here. Finally, we compute the similarity by applying a Gaussian function to the computed distance. We compute these pairwise distances only for a few randomly chosen points to obtain column samples of the matrix $\mathbf{Z}$, which are then used to find the low-rank factorization $\mathbf{Z} = \mathbf{M}^T\mathbf{M}$. It is important that the factorized matrix $\mathbf{M}^T\mathbf{M}$ remains a completely positive matrix as otherwise the quadratic optimization problem is no longer positive definite and negative energies can occur during the optimization. We use a NMF factorization to find $\mathbf{M}$ and $\mathbf{U}$ via alternating non-negative least squares.

In Figure 7.3, we demonstrate the effect of this regularization. For this, we have created a synthetic dataset, in which in a certain region (marked in green) all samples with $\theta_h < 70°$ were removed, creating a large hole in the dataset. As can be seen in Figure 7.3b, the local smoothness regularization is not sufficient to



(a) Reference **(b)** Without spatial regularization **(c)** With spatial regularization

Figure 7.3: *Effect of spatial regularization for a dataset with insufficient samples in the green regions.*

reconstruct this area correctly, resulting in a noisy reconstruction with black spots. When the spatial regularization (here, with the exact neighborhood matrix $\mathbf{Z}$) is additionally used, the holes in the sampling can be filled.

After a few iterations of our fitting algorithm, we update the regularization by using our current fitting result. For this, we compute the exact pairwise distances between the texels of our fitting result (which is efficiently possible for our factorized representation) and compute a new factorization $\mathbf{Z} = \mathrm{M}^T\mathrm{M}$ from these distances. Our current result was obtained using both the previous regularization and the available samples. We therefore expect that, at texels for which enough samples are available, this provides a much better approximation of the correct pairwise distance matrix than the original one obtained from mean colors. On the other hand, when the number of samples is not sufficient, the fitted result should still closely resemble the original distance matrix. Currently, for performance reasons, we use just one update of $\mathbf{Z}$. However, it might be worth further investigation whether several iterated updates provide better results.

7.3 Results

To demonstrate that our fitting algorithm is generally capable of obtaining similar results as the PARAFAC decomposition we employed in the previous chapter, we again have evaluated our technique on several example materials from the MERL BRDF database (see Section 2.1.2). These BRDFs have a dense, regular $\sqrt{\theta_h}, \theta_d, \phi_d$ sampling of $90 \times 90 \times 180$ and contain 3 color channels. From these datasets, we have drawn samples at randomly chosen locations (obtained via linear interpolation

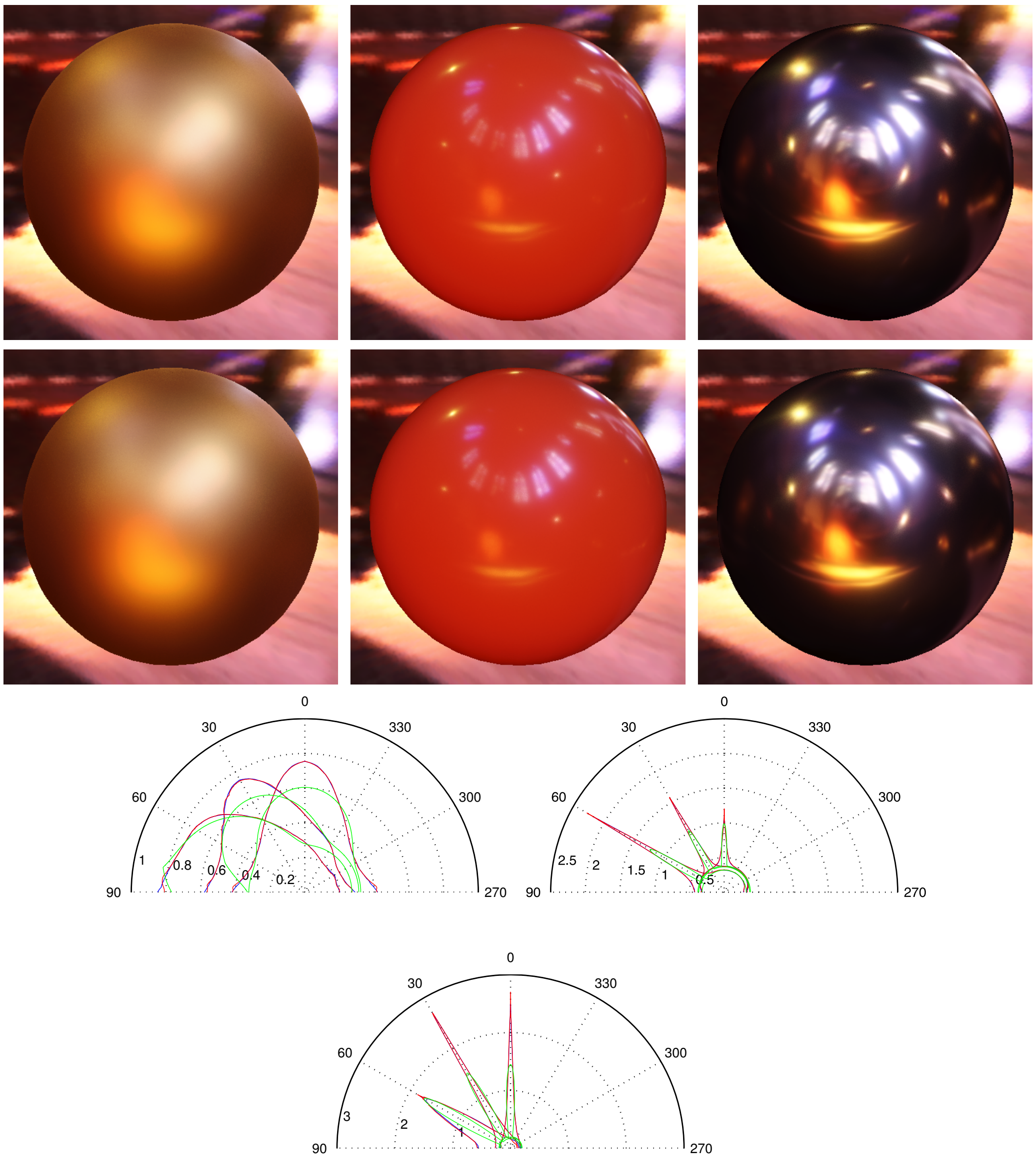
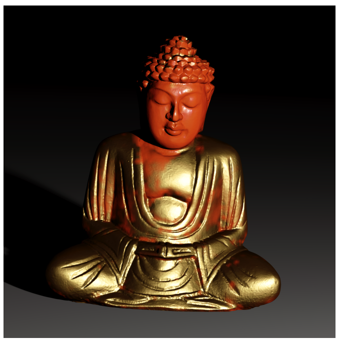
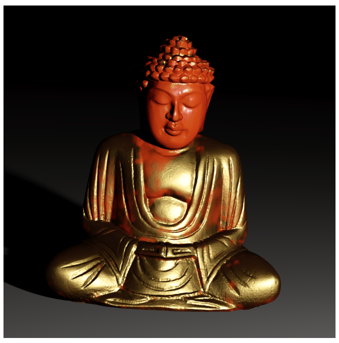
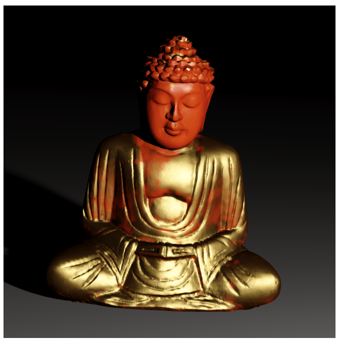
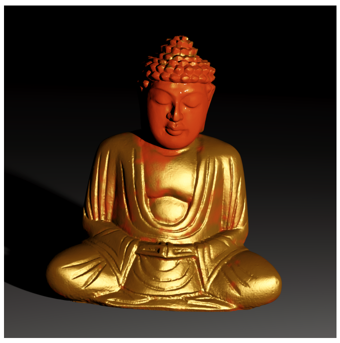


Figure 7.4: *Comparison of the original (top) BRDFs and results (middle) obtained by fitting with our method against 100,000 samples drawn randomly from the materials* `gold-metallic-paint`, `red-phenolic` *and* `grease-covered-steel`*. The corresponding plots show that the reference (red) is very accurately reproduced by our fit (blue). In green, a comparison to a Cook-Torrance fit is shown (parameters for* `gold-metallic-paint` *and* `red-phenolic` *are taken from [NDM05],* `grease-covered-steel` *was fitted by us). The fourth root was applied to the plots, which show the BRDFs for $\theta_i = 0°, 30°, 60°$.*


(a) Ground truth


(b) Our technique


(c) BTF


(d) Cook-Torrance [NDM05]

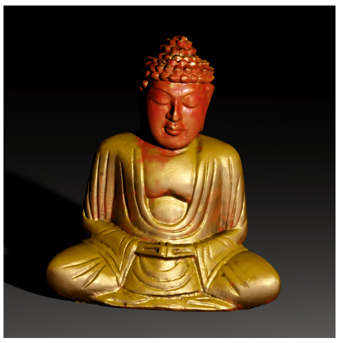

(e) Cook-Torrance

Figure 7.5: *Comparison for a synthetic ground truth dataset. This image compares our technique with a BTF (c), a spatially varying linear combination (d) of the Cook-Torrance fitting results reported in [NDM05] (using the original distribution map), and a linear combination of eight Cook-Torrance basis-BRDFs in (e) fitted against the same samples as our representation.*

to get samples not lying on the grid of the original dataset), which were then used to fit our model. For the approximation, we used $C = 12$ components and the same angular resolution as the original dataset, resulting, when stored as single precision floats, in a size of $12 \cdot (90 + 90 + 180 + 3) \cdot 4\text{B} \approx 17\text{KB}$, which corresponds to a compression by a factor of about $1{,}000$ compared to the 16.5MB needed for the original dataset. The results in Figure 7.4 show that we still achieve a very exact reproduction of the original BRDF. Even the irregular highlight shape of the `grease-covered-steel` example, resulting in unevenly blurred highlights, is well-preserved.

In Figure 7.5, we show an evaluation of our technique on a synthetic dataset. For this, we created a realistic test case, closely resembling a real-world dataset while

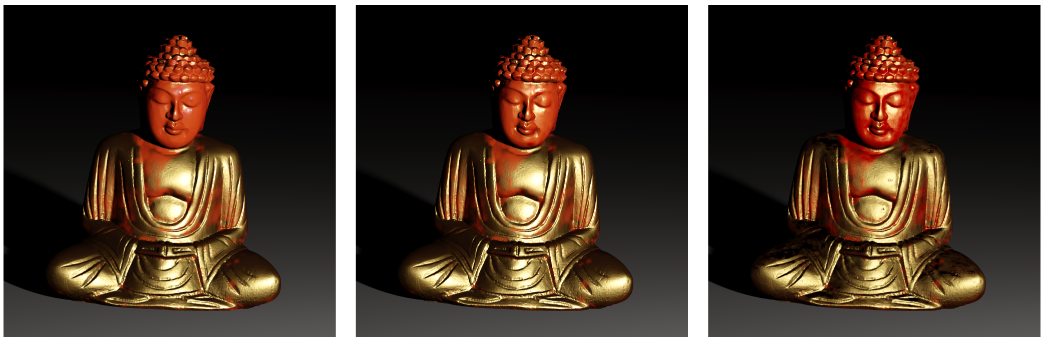

(a) $37 \times 37 = 1,369$ **(b)** $18 \times 18 = 324$ **(c)** $9 \times 9 = 81$

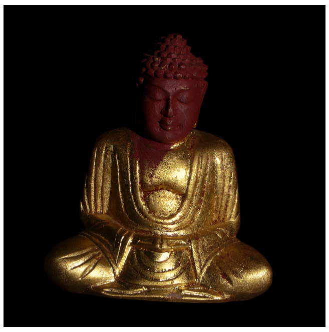
Figure 7.6: *Results for coarser sets of input images covering fewer view and light directions.*

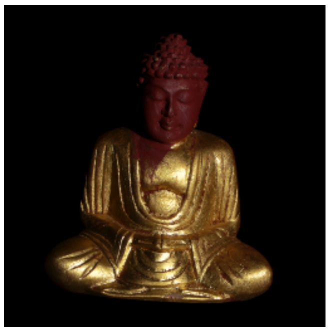
still fulfilling all assumptions of our approach. We used a scanned mesh from the Buddha dataset provided in [SWRK11]. To obtain an SVBRDF, we derived a distribution map from the color texture of the original dataset (with a resolution of 128×128) and then used spatially varying linear combinations of measured isotropic BRDFs (`red-phenolic` and `gold-metallic-paint` from the MERL BRDF database). This way, we obtained a synthetic dataset, which looks similar to the original one, but with perfectly isotropic, spatially varying BRDF. We rendered synthetic images under point-light illumination, employing the same imaging conditions as in the real-world setup, with the exception that only a subset of $75 \times 75 = 5{,}625$ view and light direction combinations is chosen. From these images, we then extracted the input samples, discarding those which are invalid due to occlusions and shadowing. The initialization for the neighborhood graph was computed on all remaining samples. However, we only drew 150 samples for each texel of the final texture randomly (using a simple binning to obtain uniformly drawn samples) to reduce the computational load during the actual optimization. We then used our technique to fit a $16{,}384 \times 90 \times 90 \times 180 \times 3$ tensor with $C = 32$ components (16,384 entries for the 128×128 spatial dimension, $90 \times 90 \times 180$ for the angular dimensions $\sqrt{\theta_h}, \theta_d, \phi_d$ and 3 color channels). For this resolution, the uncompressed dataset would already require about 267 GB when stored as single precision floating point values, making techniques based on an explicit representation of the whole dataset very impractical. In contrast, our compressed representation requires only $32 \cdot (180 + 90 + 90 + 16384 + 3) \cdot 4\text{B} \approx 2\text{MB}$.

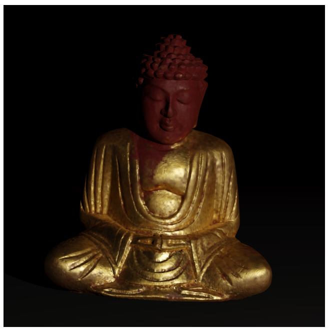
In Figure 7.5b, we show that using our technique a nearly exact fit to the ground truth data (Figure 7.5a) can be obtained. The result of resampling the input-data to a BTF representation with a direction sampling of 151×151 (using the technique from [SWRK11]) is demonstrated in Figure 7.5c. Even though the dataset is shown without any compression, the resampling itself already introduced considerable errors. Especially the highlights in the face could not be resolved by the available angular resolution and thus were lost during the resampling. Still, the dataset already requires $128 \times 128 \times 151 \times 151 \times 3 \times 4\text{B} \approx 4\text{GB}$ when stored as single precision floats and would grow considerably, if higher angular resolutions were to be used. In Figure 7.5d, we show the result of approximating the ground-truth dataset as good as possible via a Cook-Torrance model. For this, we used the original distribution map with two Cook-Torrance basis-materials which were fitted directly against the original MERL BRDF data (the parameters were taken from the supplementary material of [NDM05]). Even in this idealized setting, an exact reproduction was not possible. In Figure 7.5e, we show a result of fitting a linear combination of eight Cook-Torrance BRDFs against the same samples used for our approximation. We initialized the Cook-Torrance fitting by clustering the mean colors we used for creating our neighborhood graph. We then iterated between optimizing the BRDF parameters and the distribution maps using an similar approach as in [LKG*03].

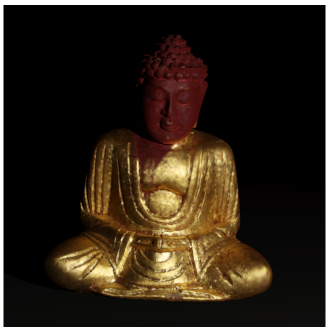
In Figure 7.6, the effect of a reduction of the angular sampling of the input images is shown. While a very good reconstruction is still possible with $37 \times 37 = 1,369$ input images, the highlights get blurred when only $18 \times 18 = 324$ images are used. With only $9 \times 9 = 81$ images, artifacts (black spots in the highlights) become visible. We think that the main reason for the unsharp highlights is the neighborhood regularization and expect that a more sophisticated approach for its extraction might improve the results. The black spots in the highlights occur when the sampling in the highlight region gets too sparse.

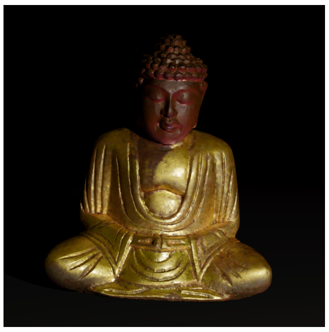
To evaluate our technique on real-world datasets, we use three different datasets. The results are shown in Figure 7.6. The first one is the Buddha dataset from [SWRK11], which has been captured with our Multi-View Dome (see Section 2.1.3). It consists of a quite accurate geometry and a total of $151 \times 151 = 22{,}801$ images. We used the same fitting approach as for the synthetic dataset, but this time a spatial resolution of 256×256 was employed. This is a challenging example, since this dataset does not fully comply with our assumptions: First, the reconstructed geometry and camera calibration is not completely accurate. Second, due to the scratches on the surface the material is not perfectly isotropic. Third, the object exhibits considerable interreflections and there are still many surface details finer than the resolution of our texture and geometry. Still, our reconstruction provides a good approximation and is capable of reproducing the sharp highlights. The other two datasets use a sampling of $264 \times 198 = 52{,}272$ view and light directions and have been captured with our Mobile Dome (see Section 2.1.4). The red billiard ball demonstrates that our technique can cope with very specular materials. In contrast to the BTF, which does not have a sufficiently high angular resolution, we

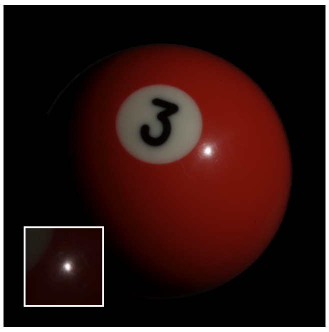

(a) Input image

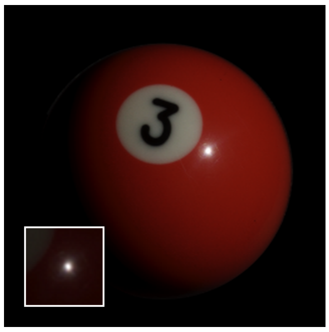

(b) Low-Res input image

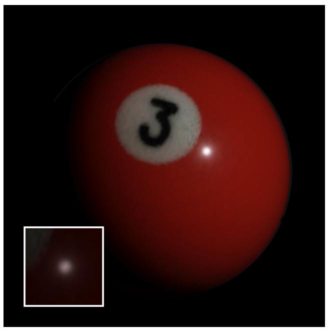

(c) Our technique

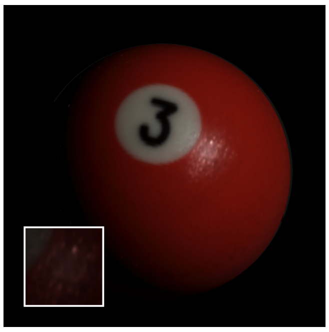

(d) BTF

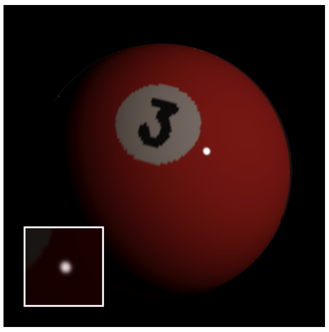

(e) Cook-Torrance

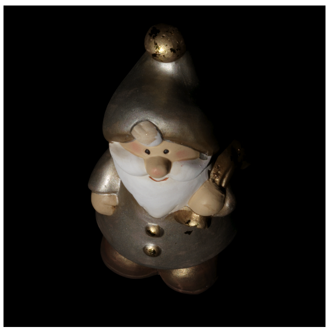

(f) Input image

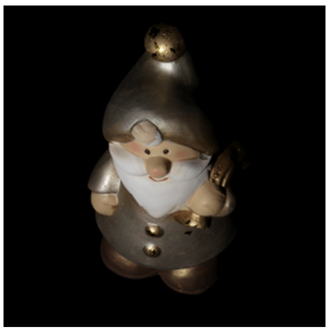

(g) Low-Res input image

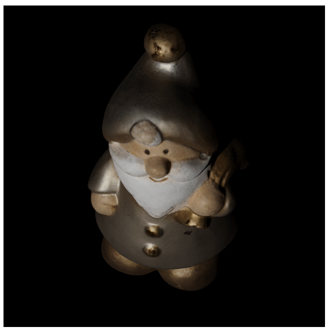

(h) Our technique

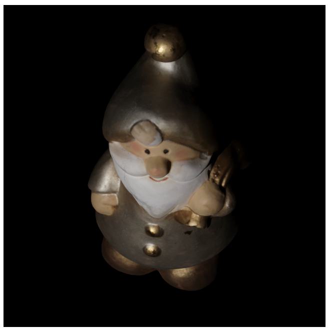

(i) BTF


(j) Cook-Torrance


(k) Input image


(l) Low-Res input image


(m) Our technique


(n) BTF

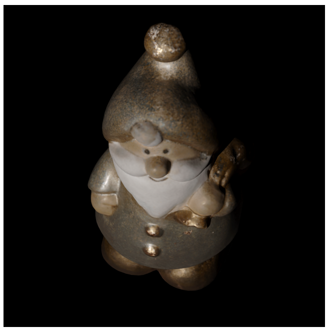

(o) Cook-Torrance

Figure 7.6: *Comparison for real-word datasets. Here, we evaluate our technique using real-world datasets. (a),(f),(k) show each one of the original input images and (b),(g),(l) are lower resolution version of these images to remove high-frequency details which are not represented due to the used texture resolution. We compare this to our technique (c),(h),(m), a BTF approximation (d),(i),(n) and a spatially varying linear combination of eight (three with ideal distribution map for the billiard ball) Cook-Torrance Basis-BRDFs (e),(j),(o).*

are able to reproduce the shape of the highlight. Unfortunately, its intensity is still underestimated, which is visible in the close-up of the billiard ball with different tonemapping. This could probably be improved by a more thorough calibration of the measurement setup and using a better sampling strategy to get more samples in the highlight. The last example uses a higher spatial resolution of 512×512 and demonstrates the performance of our approach for objects with several materials. All materials are reproduced well, apart from the few sparsely distributed spots of very specular gold paint.

Due to the nature of the alternating least squares optimization, the algorithm is not guaranteed to find a global minimum. However, as our results demonstrate, when a sufficient number of samples is given, a good fit can be obtained robustly (we initialized with random values and did not use any repeated fitting). The interior optimization without weight updates always converges as each of the individual least squares problems is globally optimally solved and thus can never increase the error. For the outer weight update loop, a convergence cannot be guaranteed, since each time the weights are updated, the error might increase again. Still, we nearly always observed a strictly decreasing error, when comparing the errors after each of the weight update steps. For the synthetic and real-world Buddha dataset, we used 20 iterations for the inner update loops and eight iterations for the outer ones (including one update of the spatial regularization). The use of a convergence threshold, however, might further improve the computation times. As a preprocessing step we need 3h for the 128×128 texture resolution and 12h for the 256×256 dataset to extract the samples from the input images (including shadowing/occlusion calculation and resampling for the mean colors). With these settings, fitting the 32 component SVBRDF against the approximately $1,200,000$ samples for the 128×128 dataset requires about 4h (and about 3GB RAM) and about 16h (and about 9 GB RAM) are needed for the about 5,000,000 Samples of the 256×256 dataset (150 Samples per texel for a texture atlas with an occupancy of about 50%). The timings were obtained on an Intel I7 950 CPU with 3.07Ghz and 12GB of RAM. The algorithm was mainly implemented in MATLAB, using moderately optimized C++ to compute $\mathbf{S}$ and parts of $\mathbf{H}$. With a Geforce GTX 580, the largest dataset can be rendered under a point-light with at least 80 FPS at a resolution of 1900×1000.

7.4 Limitations and Future Work

Since our approach is completely data-driven and does, apart from the regularization, not utilize any prior knowledge about SVBRDFs, it is not applicable if an extremely coarse angular sampling is available. In these cases, very implausible results are possible and analytical models should be preferred, as they specify more prior knowledge and thus can cope with smaller numbers of samples.

One of the main problems with the sample density are black artifacts in the highlight regions. This effect is probably due to a bias in the utilized regularization. Since we punish the second derivative along all of the coordinate axes, it is advantageous during the optimization to draw the factorized function for one of the modes towards 0 to reduce the price of the regularization along the other modes. The problem is especially worsened by the $\sqrt{\theta_h}$ parameterization, which reduces the sample density in parameter space near the highlight. For the few support points nearest to $\theta_h = 0$, there were often no samples available at all, even in the datasets with denser angular input samplings. Since we cannot use a wrap-around for the regularization of this mode, the function is only constrained by the smoothness regularization between the smallest input sample and $\theta_h = 0$, which leaves it free to diverge against 0 at the boundary. We found that it helps to add an additional regularization which punishes the gradients of the functions $f^{(c,d)}$ of the θ_h mode for the few support points which are smaller than the smallest available sample to prevent these functions from diverging towards the highlight.

In the future, a solution which solves the underlying problem would be desirable. Either a different regularization without this bias would have to be applied or additional regularizations which constrain the functions better in sparsely sampled regions should be included. For example, in 3D reconstruction sometimes a ballooning term is added to counteract the shrinkage bias of the minimum surface regularization (e.g. [VTC05]). Though a ballooning term is probably not desirable for BRDFs, it might be possible to find a suitable additional regularization.

We investigated only isotropic SVBRDF datasets so far, but we expect that our representation could be extended by an additional dimension and used to also represent anisotropic materials efficiently. The main problem here is that suitable tangent directions are needed to compress and fit these anisotropic BRDFs. This could probably be tackled by integrating an additional step which optimizes the tangent directions into the iterative optimization algorithm.

7.5 Conclusion

We have presented an algorithm for fitting a spatially varying BRDF represented as a sum of separable functions to a set of irregularly sampled data points obtained from input images taken under varying view and light directions. As we have demonstrated, this representation is very compact, well-suited for real-time rendering and capable of representing spatially varying BRDFs very accurately. By using a regularization which favors locally smooth and spatially coherent solutions, the representation can be fitted even when only a rather small number of samples is available, e.g. due to a coarse angular sampling of the input images. The fitting algorithm is very efficient both in terms of memory requirements and runtime performance, as it does not require at any time to compute the fully decompressed dataset but instead only operates on the factorized representations.

7.6 Derivation of the regularization energy

Here, we give the full derivation of the quadratic form for the regularization energy (Equation 7.4).

$$
\begin{aligned}
& \left\| \tilde{\mathcal{W}} \otimes \left(\mathcal{F} \times_g \mathbf{R}^{(g)}\right) \right\|^2 \\
&= \left[\tilde{\mathcal{W}}^2 \otimes \left(\mathcal{F} \times_g \mathbf{R}^{(g)}\right) \otimes \left(\mathcal{F} \times_g \mathbf{R}^{(g)}\right) \right]_S \\
&= \Bigg[\left(\sum_{c=1}^{C_W} \mathbf{w}^{(c,1)} \circ \cdots \circ \mathbf{w}^{(c,D)} \right) \otimes \\
&\qquad \left(\left(\sum_{c_1=1}^{C} \mathbf{f}^{(c_1,1)} \circ \cdots \circ \mathbf{f}^{(c_1,D)} \right) \times_g \mathbf{R}^{(g)} \right) \otimes \\
&\qquad \left(\left(\sum_{c_2=1}^{C} \mathbf{f}^{(c_2,1)} \circ \cdots \circ \mathbf{f}^{(c_2,D)} \right) \times_g \mathbf{R}^{(g)} \right) \Bigg]_S \\
&= \Bigg[\sum_{c_1=1}^{C} \sum_{c_2=1}^{C} \sum_{c_3=1}^{C_W} \left(\mathbf{f}^{(c_1,1)} \otimes \mathbf{f}^{(c_2,1)} \otimes \mathbf{w}^{(c_3,1)} \right) \circ \cdots \circ \\
&\qquad\qquad \left(\left(\mathbf{R}^{(g)} \mathbf{f}^{(c_1,g)}\right) \otimes \left(\mathbf{R}^{(g)} \mathbf{f}^{(c_2,g)}\right) \otimes \mathbf{w}^{(c_3,g)} \right) \circ \cdots \circ \\
&\qquad\qquad \left(\mathbf{f}^{(c_1,D)} \otimes \mathbf{f}^{(c_2,D)} \otimes \mathbf{w}^{(c_3,D)} \right) \Bigg]_S \\
&= \sum_{c_1=1}^{C} \sum_{c_2=1}^{C} \sum_{c_3=1}^{C_W} \left[\mathbf{f}^{(c_1,1)} \otimes \mathbf{f}^{(c_2,1)} \otimes \mathbf{w}^{(c_3,1)} \right]_S * \cdots * \\
&\qquad\qquad \left[\left(\mathbf{R}^{(g)} \mathbf{f}^{(c_1,g)}\right) \otimes \left(\mathbf{R}^{(g)} \mathbf{f}^{(c_2,g)}\right) \otimes \mathbf{w}^{(c_3,g)} \right]_S * \cdots * \\
&\qquad\qquad \left[\mathbf{f}^{(c_1,D)} \otimes \mathbf{f}^{(c_2,D)} \otimes \mathbf{w}^{(c_3,D)} \right]_S
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{c_1=1}^{C}\sum_{c_2=1}^{C}\sum_{c_3=1}^{C_W}(\mathbf{f}^{(c_1,1)})^T\mathbf{W}^{(c_3,1)}\mathbf{f}^{(c_2,1)} * \cdots * \\
&\qquad\qquad \left(\mathbf{f}^{(c_1,g)}\right)^T \left(\mathbf{R}^{(g)}\right)^T \mathbf{W}^{(c_3,g)}\mathbf{R}^{(g)}\mathbf{f}^{(c_2,g)} * \cdots * \\
&\qquad\qquad \left(\mathbf{f}^{(c_1,D)}\right)^T \mathbf{W}^{(c_3,D)}\mathbf{f}^{(c_2,D)} \\
&= \left(\mathbf{f}^{(l)}\right)^T \mathbf{H}^{(g)}\mathbf{f}^{(l)}
\end{aligned}
$$

Part II

Material Interpolation

CHAPTER 8

BRDF INTERPOLATION

In Chapter 6, we have shown that a PARAFAC based representation allows for a very accurate and compact representation of a wide range of measured BRDFs. However, editing such a data-driven representation is much more difficult than model based ones. For the latter, several techniques for the intuitive editing have been proposed. For example, in [PFG00] the Ward model is reparameterized perceptually to achieve a more intuitive selection of the parameters, in [CPK06, NSRS13] the user can paint the lobes of a BRDF directly or in [NDM06] the parameters are chosen by repeatedly selecting materials from a set of example images.

It would be desirable to perform similar edits for data-driven BRDF representations as is possible for model based ones, such as changing the shape, width and intensity of the lobe and the diffuse color of the material. One approach for this is the use of a factorization into a small number of terms, that are presented as curves which can then be edited (e.g. [LBAD*06, BAOR06]). However, this is not a very intuitive interface, and the small number of factors limits the complexity of the BRDFs that can be edited this way. An alternative approach is to select several samples from a database of measured materials and then to interpolate between them to change the desired characteristics. For example, in [WAKB09], the BRDFs from the MERL BRDF database are arranged perceptually uniformly in two dimensions via multi-dimensional scaling and then barycentric interpolation in a Delaunay triangulation is used to create new BRDFs.

However, the barycentric interpolation results in a linear blending between BRDFs, which can produce rather unintuitive results. It does not create intermediate lobe shapes, as for example interpolations between the parameters of model based representations would do. Instead, the width of the lobes does not change continuously, but only the intensity of the two blended lobes changes during the interpolation.

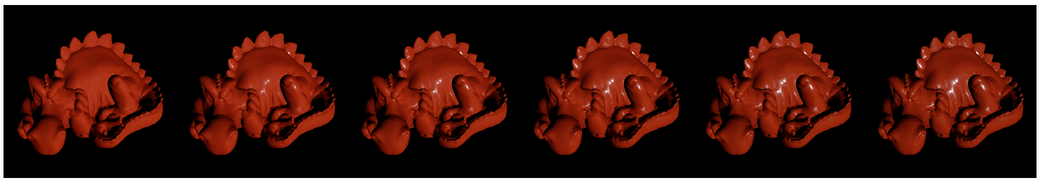


Figure 8.1: *Linear interpolation between* `Dark Red Paint` *and* `Red Phenolic`

An example for this limitation is shown in Figure 8.1, where we perform an interpolation between a very specular and a diffuse material. This does not produce materials with varying microscopic roughness, resulting in different lobe shapes, but instead only the intensity of the specular lobe changes.

Matusik et al. [MPBM03a] proposed to use a non-linear embedding of the space spanned by the samples in a database into a lower-dimensional space. This space is then parameterized according to perceptual traits like *redness* or *roughness*. However, this approach requires enough intermediate samples to perform a smooth interpolation between materials with different lobe shapes and thus needs a quite large database. Furthermore, for each trait used during the editing a perceptual annotation of this database is necessary.

In this chapter, we follow a different approach, which can interpolate the lobe shape even when only two materials are given. Extending upon this, we are also able to interpolate between several materials. Our technique uses continuous dynamic time warping to first align the lobe shapes of two BRDFs with each other and then performs an interpolation. This way, we achieve smooth transitions between different lobe shapes. By precalculating these warpings and taking advantage of our compact PARAFAC representation, real-time rendering of interpolated BRDFs is possible. This way, we can offer the user an interactive graphical user interface, which allows for a very intuitive exploration of the space spanned by a set of BRDFs.

The work presented in this chapter has already been published as part of the technical report "A compact and editable representation for measured brdfs", by Roland Ruiters and Reinhard Klein (Technical Report CG-2010-1, University of Bonn, December 2010).

8.1 Interpolation

In order to allow for a transition between different lobe shapes, we perform an additional warping of the BRDFs prior to the interpolation, which aligns the shape of the two BRDFs with each other. If we would allow an arbitrary warping transform, this would itself be a high-dimensional function which would have to be stored in an efficient way. Instead, we decided to use warping transformations along the coordinate axes, which can then be stored and accessed in a very efficient way. This is achieved by introducing a set of coordinate transformation functions $m^{(i)}_{A\overset{\alpha}{\to}B}(x)$. Each of these functions remaps the coordinate i in dependence on the interpolation parameter α during an interpolation operation from BRDF A to BRDF B. Using these mappings a warped BRDF is thus given by

$$\rho_{A\overset{\alpha}{\to}B}(\theta_h, \theta_d, \phi_d, c) = \rho_A(m^{(1)}_{A\overset{\alpha}{\to}B}(\theta_h), m^{(2)}_{A\overset{\alpha}{\to}B}(\theta_d), m^{(3)}_{A\overset{\alpha}{\to}B}(\phi_d), c).$$

By warping the BRDF A by α towards the BRDF B and by warping B by $1-\alpha$ towards A, we get two BRDFs with aligned features. The final interpolated BRDF is then obtained by interpolating between the two warped BRDFs. We found that because of the large dynamic range of the BRDFs a linear interpolation does not lead to good perceptual results here. The lobe of a specular material is often by several orders of magnitude larger than the diffuse parts of a BRDF. When the warping enlarges the size of this lobe, performing a linear blending results in materials with unrealistically bright lobes. Instead, we decided to interpolate in log-space, which is, similar to the difference function from Chapter 6, again motivated by the Weber-Fechner law, from which follows that the intensity perception is better described by a logarithmic relationship than a linear one.

We thus obtain the following interpolated BRDF ρ':

$$\begin{aligned}\rho'(\theta_h, \theta_d, \phi_d, c) = \exp[&(1-\alpha)\log \rho_{A\overset{\alpha}{\to}B}(\theta_h, \theta_d, \phi_d, c) \\ &+\alpha \log \rho_{B\overset{1-\alpha}{\to}A}(\theta_h, \theta_d, \phi_d, c)]\end{aligned}$$

Using linear blending would have had the advantage, that the resulting interpolated BRDF could directly be represented as a PARAFAC decomposition by warping and scaling the functions $f_i^{(j)}$ of the two BRDFs and then combining these into a new PARAFAC decomposition. When using non-linear blending, an additional recompression step is necessary to represent the interpolated BRDF in the same PARAFAC decomposition we use for the materials in the database. This interpolation technique could generally be applied to interpolate any kind of multi-dimensional data. However, since the warping is performed for each of the coordinates independently, this approach is obviously heavily dependent on the parameterization. The warping is only possible when the parameterization is chosen in such a way that the coordinate axes are mostly independent from each other. This is similar to the requirements for good compression results via a tensor

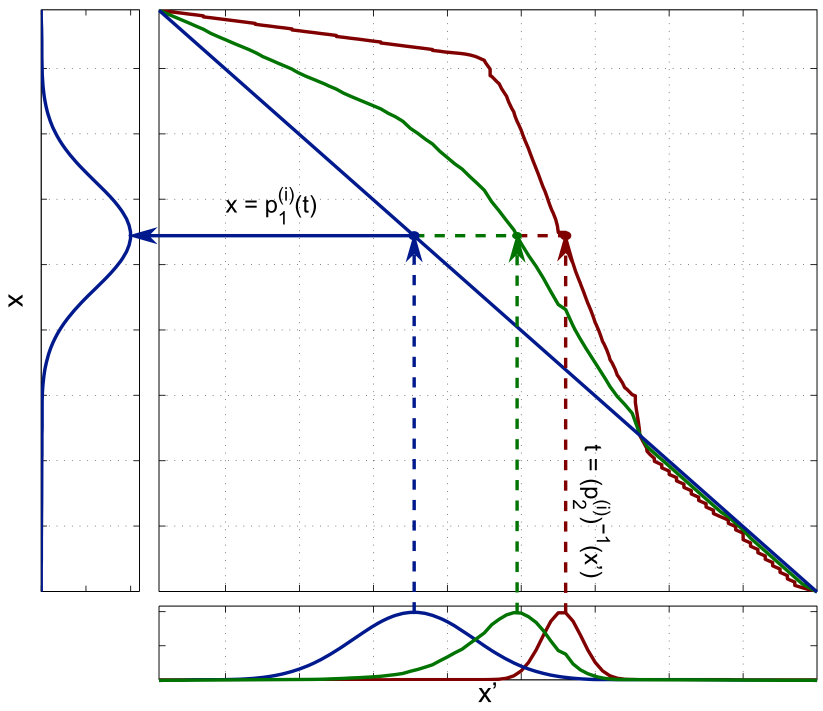


Figure 8.2: *Example of our interpolation technique. Warped path (red), path for interpolation with* $\alpha = 0.5$ *(green) and unwarped path (blue) are shown in the main area. The corresponding warped functions are given at the bottom.*

factorization. Therefore, the half-diff parameterization, which aligns typical BRDF features with the coordinate system, gives also good results here.

8.1.1 Coordinate Transformation Functions

To find the coordinate transformation functions $m^{(i)}_{A \overset{\alpha}{\to} B}(x)$ we use *dynamic time warping*. Dynamic time warping is a technique which uses dynamic programming to find the best alignment between two sequences of time data in a way that minimizes a given error function. First used for speech recognition (e.g. [SC78]), it has since found many applications in pattern recognition and information retrieval. Given two sequences $A(t)$ and $B(t)$ and a function $d(x, y)$ measuring the difference between two sequences, it finds two functions $p_1(t)$ and $p_2(t)$ which minimize $d(A(p_1(t)), B(p_2(t)))$, usually under the constraint that p_1 and p_2 are monotonous.

To apply this technique to higher-dimensional datasets, such as tensors representing BRDFs, we perform the warping along each mode independently, while keeping all other modes fixed. Since the warping along one mode can influence the result of the warping operations along other modes, we iterate this process several times.

For each mode i, the current warping functions along all the other modes are applied to the tensor and then the warping functions $p_1^{(i)}$ and $p_2^{(i)}$ are updated. For this, the two tensors $\mathcal{A}$ and $\mathcal{B}$ which are to be aligned are unfolded along mode i and the resulting matrices are considered as two vector-valued functions $\mathbf{a}_i(x)$ and $\mathbf{b}_i(x)$. For these functions, the one-dimensional dynamic time warping can then be performed. For the dynamic time warping, we do not use the full color BRDFs but instead only the luminance, since we want the calculated warpings to be independent from the color of the two materials.

Once the warping path given by $p_1^{(i)}$ and $p_2^{(i)}$ has been calculated, the mapping between coordinates is obtained by first projecting the coordinate x' onto the warping path by using the inverse of $p_2^{(i)}$: $t = \left(p_2^{(i)}\right)^{-1}(x')$. Then the coordinate is projected from there back onto the parameter space by calculating $x = p_1^{(i)}(t)$ (see Figure 8.2 for an illustration).

So far, the warping path only warps the BRDF A onto B. To instead smoothly interpolate between the two BRDFs we define an interpolation function $I(f, x, \alpha) = (1-\alpha)x + \alpha f(x)$, which performs a linear interpolation between the identity function, representing an unwarped path, and the function f representing the warped path.

The interpolated mapping is then obtained by applying this interpolation function to both mapping steps. We thus define

$$m^{(i)}_{A \xrightarrow{\alpha} B}(x) = I\left(p_1^{(i)}, I\left(\left(p_2^{(i)}\right)^{-1}, x, \alpha\right), \alpha\right).$$

This choice of the mapping function is motivated by the necessity to perform the interpolation efficiently with two lookups, as it allows to precalculate and store the inverted function $\left(p_2^{(i)}\right)^{-1}$.

8.1.2 Continuous Dynamic Time Warping

When the warping functions and the sequences are defined for discrete values t and the distance function d is defined as a sum over the element-wise difference at each of these values, the problem can efficiently be solved using dynamic programming. However, we found that a discrete solution is often not sufficient to perform the mapping between two BRDFs. Even though we represent the BRDF as a tensor, during the rendering it is interpreted as a piece-wise linear function and not a set of tabulated values. This is a problem since the lobe for very specular materials often has a steep slope and thus the whole lobe is represented by just a few samples.

When performing the warping only on these discrete samples instead of taking the fact into account that we perform a linear interpolation between the samples, the resulting BRDFs contain step-like artifacts, which can clearly be seen in the renderings.

One way to overcome this limitation would be to oversample the tensor. By increasing the resolution before the time warping is performed, these problems can be reduced. However, because of the steep lobes, the resolution has to be increased considerably to obtain good results. Since the runtime complexity of dynamic time warping is $O(n^2)$ in the resolution this can become quite expensive. This could probably be overcome by using an adaptive refinement scheme. However, since this would also further increase the complexity of the algorithm and the results would always only remain an approximation to the correct warping path, we decided to apply the continuous dynamic time warping algorithm from [MP99] instead.

This algorithm matches two curves, each given as a set of samples, onto each other. For each of the discrete points in both curves, a corresponding point in the other curve is searched. But in contrast to the discrete dynamic time warping, this point is allowed to lie in between two points, assuming a piece-wise linear curve. Thus, for each discrete value of t only one of the two functions p_1 and p_2 has an integral value, while the other value describes a point obtained by linearly interpolating between two points of the curve. A second difference to dynamic time warping is that the two curves are compared in a translation invariant manner, by calculating the difference d not using the absolute values but instead relative differences between two points on the curve.

The algorithm in [MP99] is only given for two-dimensional curves, but it can be straightforwardly extended to curves in a high-dimensional space, such as our unfolded tensors. Furthermore, analogous to the compression we had to introduce an additional weight term $w(x, y)$ to compensate for the large dynamic range of the BRDFs. $w(x, y)$ gives a weight for the error when a point between $\mathbf{a}_i(x-1)$ and $\mathbf{a}_i(x)$ is warped onto a point between $\mathbf{b}_i(y-1)$ and $\mathbf{b}_i(y)$.

With these adaptions, the algorithm thus minimizes the cost function

$$\begin{aligned} D(\mathbf{a}_i, \mathbf{b}_i) = \sum_{t=2}^{T} w(\left\lceil p_1^{(i)}(t) \right\rceil, \left\lceil p_2^{(i)}(t) \right\rceil)^2 * \\ \|(\mathbf{a}_i(p_1^{(i)}(t)) - \mathbf{a}_i(p_1^{(i)}(t-1))) - \\ (\mathbf{b}_i(p_2^{(i)}(t)) - \mathbf{b}_i(p_2^{(i)}(t-1))\|_2^2. \end{aligned}$$

Here, the weights should again be selected in such a way that the error function is evaluated relative to the original value. Thus, when x were the reference value the

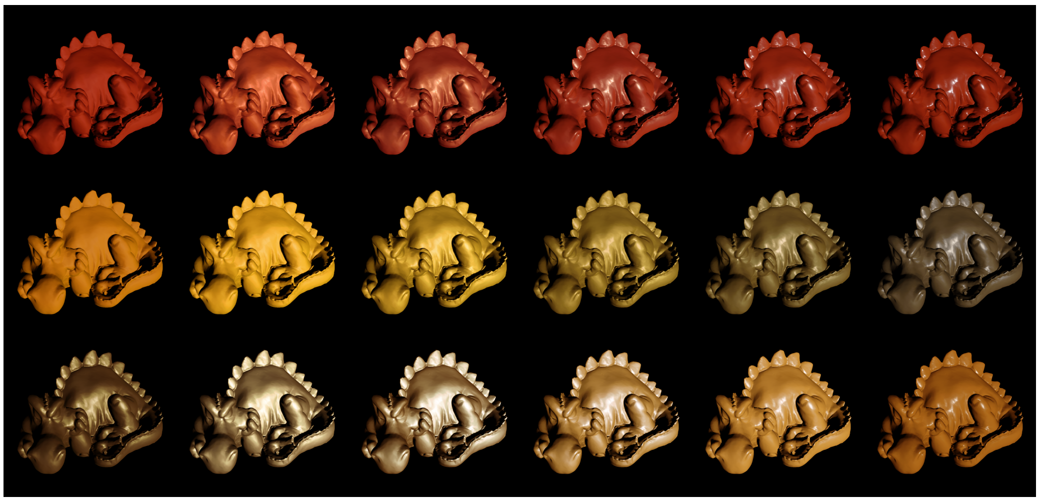


Figure 8.3: *Example interpolation sequences with our technique (from top to bottom* `Dark Red Paint→Red Phenolic, Orange Paint→Fruitwood-241, Gold Metallic Paint→Yellow Matte Plastic`*). In contrast to the linear blending, the shape of the lobe changes during the interpolation.*

weight would have to be chosen as $x^{-\frac{1}{2}}$. However, in contrast to the compression, we do not have one fixed reference value but instead four in relation to which the error could be considered. We found that setting $w(x, y)$ to the mean of $\mathbf{a}_i(x)^{-\frac{1}{2}}$,$\mathbf{a}_i(x-1)^{-\frac{1}{2}}$, $\mathbf{b}_i(y)^{-\frac{1}{2}}$ and $\mathbf{b}_i(y-1)^{-\frac{1}{2}}$ gives reasonable results.

The continuous dynamic time warping minimizes an error consisting of the differences between elements adjacent along the currently processed mode. Therefore, each time it is applied to a different mode of the tensor, the algorithm minimizes a different error function. Thus, it is not guaranteed that iterating this algorithm does converge. In most cases, we found that usually after a very small number of iterations (usually less than three iterations) the algorithm had converged to a local minimum. However, sometimes it oscillated between two solutions, in which case we simply aborted after a fixed number of iterations. When using four iterations usually between one and four minutes are required on a computer with a Q6600 CPU and 4 GB RAM to compute the warping between two BRDFs.

8.1.3 Results

A few interpolation sequences obtained with our approach are shown in Figure 8.3. Note that, in contrast to the linear interpolation in Figure 8.1, here the interpolation between a diffuse and specular material results in intermediate images with varying

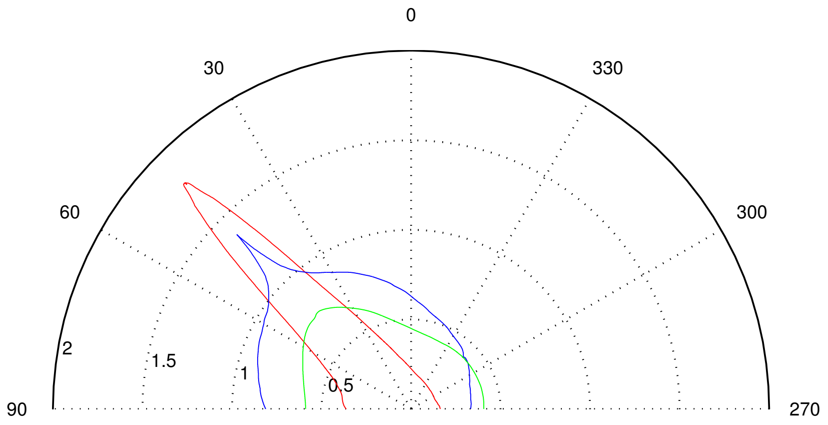


Figure 8.4: *A case in which a non-plausible intermediate BRDF (blue) is generated. Instead of becoming more diffuse, the material is even more specular than the original one.*

glossiness. We found that for most material combinations our technique resulted in plausible interpolation sequences. However, there are some combinations for which the warping created non-plausible results (such an example is shown in Figure 8.4). Furthermore, we cannot guarantee energy preservation for intermediate BRDFs.

8.2 Real-Time Rendering

Though the calculation of the warping paths $p_1^{(i)}$ and $p_2^{(i)}$ is rather expensive, they can be precomputed. The interpolation itself can then be performed for an arbitrary α in a shader. This shader first computes the half vector and the difference vector from the light and view direction. The warping can then be performed by evaluating the coordinate transformation functions $m^{(i)}_{A \xrightarrow{\alpha} B}(x)$ for both participating BRDFs to obtain the warped parameter sets. These are then used to evaluate the two BRDFs and the results are finally interpolated.

We represent the BRDFs using the tensor decomposition from Chapter 6, which allows for a very accurate representation, requires only a small amount of memory and can be evaluated efficiently in the shader. However, since the warping and interpolation are only performed before and after the evaluation of the BRDF, this technique could be combined with an arbitrary BRDF representation. Even the use of two different representations for the BRDFs participating in the interpolation would be possible.

So far, we have only considered the interpolation between two BRDFs. To obtain a space of BRDFs which can be explored by the user, it is necessary to interpolate between more than two BRDFs. First, the pairwise warping paths for all participating BRDFs have to be precomputed. We then consider the interpolation between a set of BRDFs $\{B_i\}_{i\in\{1\ldots N\}}$, each of which has a weight β_i with $\sum_i \beta_i = 1$.

During the interpolation, we have to compute for each BRDF B_i warped coordinates which take all other participating BRDFs into account. For this, we first consider for each B_i pairwise interpolations towards all the remaining BRDFs B_j. For each of them, we calculate the warping parameter $\alpha_{i,j}$ as $\alpha_{i,j} = \frac{\beta_i}{\beta_i+\beta_j}$ and then determine the corresponding warped coordinates. Once this has been done for all other B_j, the following weighted sum is evaluated to combine these pairwise coordinates into the final coordinates for an interpolation between B_i and all the other BRDFs $\{B_j\}$:

$$m^{(l)}_{B_i \xrightarrow{\{\beta_j\}} \{B_j\}}(x) = \sum_{j\in\{1..N\}\setminus i} \frac{\beta_j}{1-\beta_i} m^{(l)}_{B_i \xrightarrow{\alpha_{i,j}} B_j}(x)$$

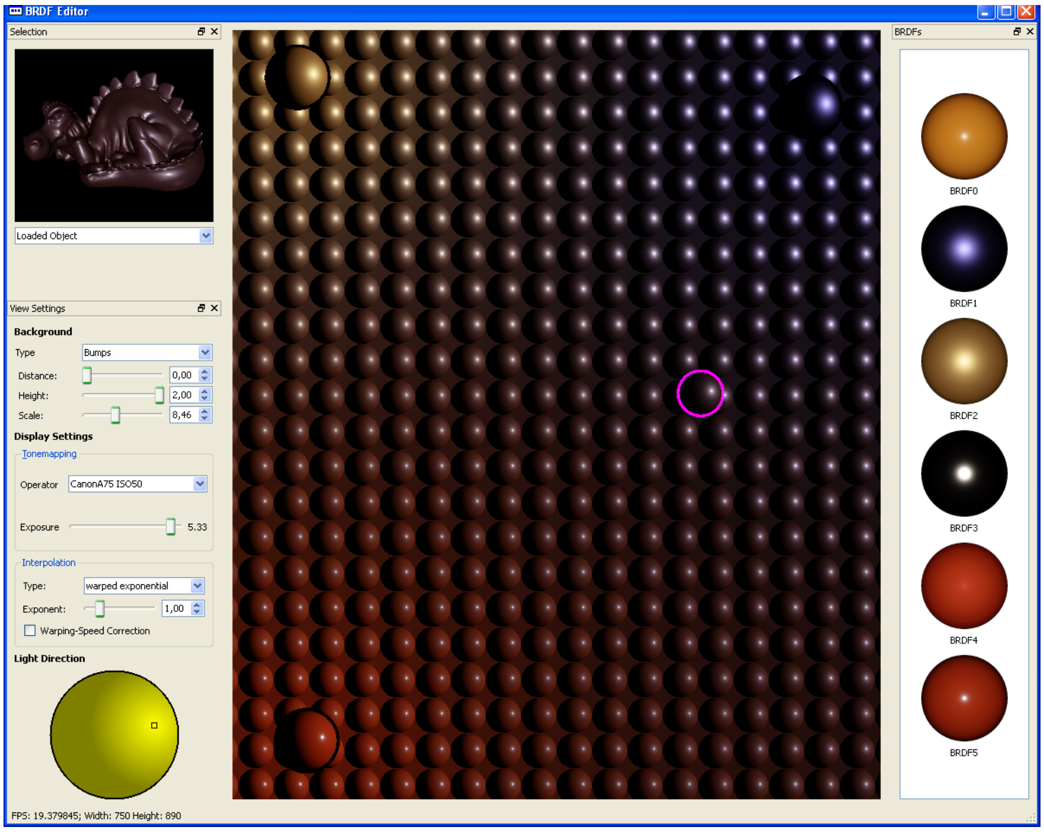
Once the final coordinates have been calculated for all B_i, these are used to evaluate the BRDFs and then the actual interpolation between the BRDFs is performed in log–space.

8.3 Interactive Exploration of the BRDF Space

When a database of measured materials, such as the isotropic MERL BRDF database (see Section 2.1.2), is available, a high-dimensional space of materials is spanned by interpolating between these BRDFs. However, each additional BRDF added to the interpolation increases the dimension of the space. Specifying the interpolation weights directly in such a way that a BRDF with a desired appearance is obtained is therefore very difficult.

To allow for a more intuitive selection of the interpolation weights, we suggest an interactive graphical user interface (shown in Figure 8.5). The user is presented with the BRDFs available in the database on the right side of the screen. From this database, she can select a number of BRDFs which are similar to the desired material and place these on the surface in the center of the screen. The placement of these BRDFs specifies then for each point of the surface a set of interpolation weights. When the BRDFs B_i are placed at the coordinates $\mathbf{x}_i$ the interpolation weights at a point $\mathbf{y}$ are given by

$$\alpha_i(\mathbf{y}) = \frac{\|\mathbf{x}_i - \mathbf{y}\|_2^{-E}}{\sum_{j=1}^{N} \|\mathbf{x}_j - \mathbf{y}\|_2^{-E}},$$



Loaded Object

Type: Bumps
Distance: 0,00
Height: 2,00
Scale: 8,46
Operator: CanonA75 ISO50
Exposure: 5.33
Type: warped exponential
Exponent: 1,00
☐ Warping-Speed Correction

FPS: 19.379845; Width: 750 Height: 890

Figure 8.5: *Graphical user interface for the interactive exploration of a space spanned by BRDFs*

where E is a user selected exponent, which describes whether the interpolation is rather local or if BRDFs also influence distant points strongly. Thus by moving the placed BRDFs the user changes interactively the interpolation weights at each point of the surface simultaneously.

Once the warping paths have been precomputed, our technique allows to evaluate the interpolated BRDF model completely within the pixel shader. We are therefore able to render interpolated BRDFs interactively and thus give the user an instant feedback how the BRDFs would look like at all points on the surface at once. To help the user judge the BRDFs, we render the background with a bump mapped pattern (in our case small half spheres). Instead of using just one preview image, showing the current selection, the user thus gets an overview over a large number of different BRDFs and can interactively explore the possible combinations by moving the selected BRDFs around. A BRDF is then selected by moving a small sphere over the surface. Here again, the user is given instant feedback in a preview window on the left, where she can directly see how this BRDF would look like when applied to the object she is currently working with.

Since the shader has to perform the lookup of the warped coordinates for each pair of the interpolated materials, the runtime of the shader is actually $O(n^2)$ in the number of materials. Thus, this technique is not suited to interpolate between a very large number of materials at the same time. However, on a computer with a Q6600 CPU, 4 GB RAM and a GeForce 8800 GTX we can render the editing surface (which has a size of about 800x900 pixels) at 18-19 FPS while interpolating between six materials. Since a higher number of materials is probably seldom required, this should be sufficient for interactive editing in many practical cases.

8.4 Conclusion

In this chapter, we demonstrated how dynamic time warping can be used to interpolate between BRDFs and how this interpolation allows for an intuitive and interactive exploration of the space spanned by a database of BRDFs.

BRDFs often are extremely non-linear in their parameters. Thus, even when the warping itself is linear with α in the parameter space, it still results in a perceptually non-linear interpolation speed between the two materials. This might be improved by introducing an additional mapping for the parameter α, similar to the approach described in [NDM06].

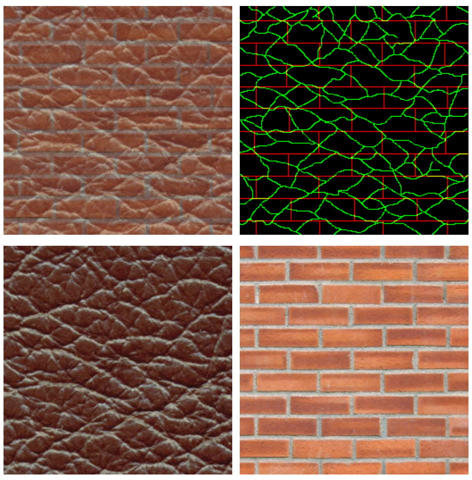
CHAPTER 9

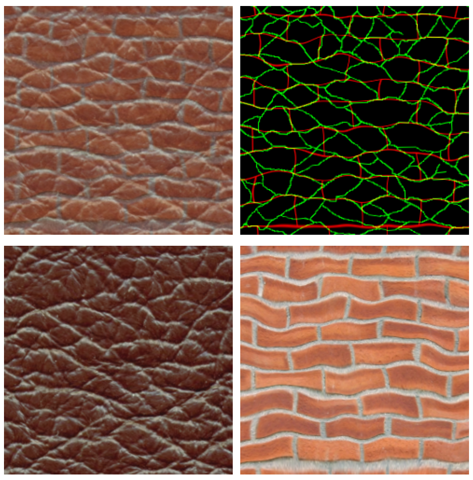
TEXTURE INTERPOLATION

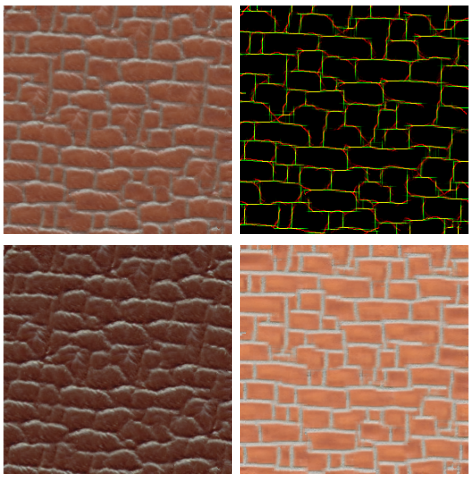
In the previous chapter, we have considered the interpolation of homogeneous materials. However, the characteristic appearance of many materials cannot be captured by a BRDF but is also dependent on the spatial variation of the reflectance behavior. In this chapter, we will therefore try to develop a similar data-driven interpolation technique for textures. This will provide the foundation for the interpolation technique for BTFs described in the next chapter. However, since textures are probably the most widely used material representation in computer graphics, the interpolation of textures is an interesting topic in itself.

Manually designing textures - procedurally or painted - is extremely time consuming, while relying on photographs limits the designer to available material samples. Moreover, data acquisition from real-world samples often requires tedious manual postprocessing and editing to achieve the desired appearance. Thus, a data-driven interpolation technique which allows the designer to create new textures by interpolating between a few captured samples would be one possibility to overcome these problems. A second important application for texture interpolation is the creation of spatially varying textures. Many objects show continuous transitions between two textures, either naturally occurring, like for example natural variations in fur or leather, or caused by wear or weathering. To create such effects, it is not sufficient to only create an interpolated material, but a smooth transition between two materials is also necessary.

However, creating meaningful interpolations for textures is a difficult problem. A simple linear interpolation between textures usually does not give reasonable results, as is illustrated in Figure 9.1a. Blending only works satisfactory when features, like for example edges, ridges or cracks, in both textures are aligned to each other. In [MZD05], it was therefore proposed to solve this problem by first computing a global warping field, which aligns corresponding features in the two texture samples, and then interpolating between the warped images. However, this approach suffers from two problems: First, the amount of deformation has to be


(a) Linear Blending


(b) Warping


(c) Linear patch blending

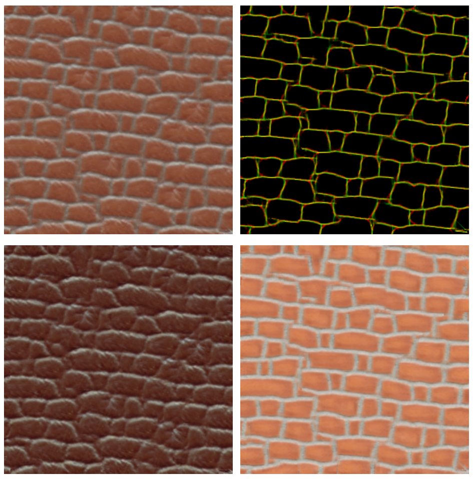

(d) Warped patches (our technique)

Figure 9.1: *Comparison of different texture interpolation approaches. The top left image is the final result, the top right demonstrates the feature alignment obtained by the techniques. The bottom row shows input images after warping/synthesis but prior to blending. Note: For our technique these are only shown for demonstration purposes. The technique interpolates the neighborhoods prior to the synthesis and performs the synthesis with the blended patches. Therefore these images are usually not created. The images correspond to an interpolation amount of 0.4.*

kept low, as otherwise the appearance of the texture changes considerably and artifacts occur in strongly warped regions. Thus, a strong regularization must to be added to penalize strong warping. As a result of this, not all features can be brought into exact alignment. Second, this approach can only be used if a continuous warping field can be found. If there are large topological differences between the layout of features in the two images, this is not possible since the topology of the features cannot be changed by applying a continuous mapping. Figure 9.1b shows an example where, due to these reasons, it was not possible to bring all features into alignment.

Based on the assumption that a texture can be modeled as a Markov-Random-Field (MRF), which is characterized by a collection of local image neighborhoods only, we propose to overcome these drawbacks by performing the interpolation directly on these local neighborhoods. We first find corresponding neighborhoods with similar feature topology, and then warp and blend these patches to create a new set of neighborhoods defining the MRF of the interpolated texture. In the next step, we use texture optimization [KEBK05] to synthesize a new image from this interpolated MRF. In contrast to approaches finding one global warping transform, our local approach requires less deformation on the patch level, since corresponding patches are chosen to have a similar feature layout and topology. Moreover, topology changes are resolved during the texture optimization which reassembles different patches in a suitable manner. Finally, we combine the patch-based synthesis technique with statistical texture synthesis [PS00] to preserve high-frequency features, which would otherwise be lost because of the patch blending and resampling necessary for warping.

As shown in Figure 9.1d, using this approach, a texture with very exact feature alignment and an intuitive looking and consistent overall structure can be synthesized. Note, that it is not sufficient to only linearly blend patches with similar topology, as this way it is not possible to bring the features in good alignment (Figure 9.1c). Furthermore, the overall structure of the resulting texture is still dominated by the right angles in the brick texture, even though the interpolation amount is set to 0.4 and thus the result should be closer to the leather.

Using our approach, interpolation is thus possible for a large class of materials, even when dealing with considerable topological differences in the layout of the features in the two textures.

This chapter corresponds to the paper "Patch-based Texture Interpolation" by Roland Ruiters, Ruwen Schnabel, and Reinhard Klein, published in *Computer Graphics Forum (Proc. of EGSR)*, 29(4):1421–1429, June 2010.

9.1 Previous Work

9.1.1 Texture Interpolation

Existing approaches to texture interpolation can roughly be grouped in three classes.

One approach is to compute a warping field which maps one texture onto the other to align features in the two images and then blend between the two warped images. In [LLSY02], manually specified feature correspondences are used to create the warping field, whereas in [MZD05] the warping is computed using an automatically extracted feature channel. In [LW07], a similar warping technique is used to create smooth transitions in synthesized video sequences of natural phenomena such as water, clouds and fire. However, computing one warping transformation for the whole image is only possible when the images are topologically similar and the features are distributed in such a way that unique correspondences and a continuous warping field can be found.

A second class of techniques is based on texture synthesis. In [Wei02], textures are interpolated by minimizing an energy function, which contains a weighted sum of the differences to the nearest patch in each of the input images. To improve the stability of the synthesis, an additional user defined feature channel is used. In [ZZV*03], a binary texton map is used instead of the feature channel. To interpolate two textures, first an intermediate texton map is created. Then, constrained texture synthesis is used to synthesize from the input textures two textures with aligned features which are finally blended. The interpolated texton mask can be obtained via linear blending [ZZV*03, TW04]. Recently, it was instead suggested [RLW*09] to use advection to create the interpolated mask. However, all of the approaches based on texton masks suffer from the problem, that the texton masks are interpolated globally. The alignment of the features is either – in the case of linear blending – not taken into account, or can only be corrected locally via the advection performed in [RLW*09].

Approaches which first warp an input image and then perform texture synthesis to restore details lost by the warping have also been proposed. In [FH07], a user specified warping is used, whereas in [LMWY09] a similar approach is used to create an image mixing the appearance of one texture with the feature distribution of a different texture.

A third class of techniques [HB95, PS00, BJEYLW01] is based on the extraction and interpolation of image statistics. From these interpolated statistics a new image can then be synthesized. However, for many complex images, statistical synthesis alone does not provide very satisfactory results. These techniques can also be combined with other synthesis techniques to help to preserve image details, which would otherwise be lost [MZD05].

9.1.2 Texture Synthesis

Given a small exemplar image, texture synthesis techniques create a larger texture with a similar visual appearance. In the last years, texture synthesis has received much attention and a considerable amount of work in this area has been published. We will therefore only cite those works directly related to our technique and refer the reader to a state-of-the-art report [WLKT09] for a more comprehensive overview.

Texture synthesis approaches can be classified into techniques based on image statistics and neighborhood-based approaches. The first neighborhood-based techniques [EL99, WL00] synthesize the image by sequentially selecting the color for each pixel from the best corresponding neighborhood in the input images. An alternative approach is used by patch-based techniques (e.g. [PFH00, LLX*01, EF01, KSE*03]), which work by copying larger patches into the synthesized texture. In [WY04], the individual patches are warped by aligning a binary feature map via a thin plate spline transformation to ensure pattern continuity during patch assembly. This approach is similar to our warping technique, but only used during synthesis and not for texture interpolation. Texture optimization techniques [KEBK05, HZW*06, KFCO*07] combine pixel- and patch-based approaches by considering texture synthesis as an optimization problem, which is solved by minimizing an energy function. A different approach is used by techniques, which decompose the image into several subbands and then enforce statistics like color histograms [HB95] and cross-correlation between subbands [PS00].

9.2 Overview

Given two images I_1 and I_2 which are samples of textures T_1 and T_2 respectively, we want to synthesize a new image I' of arbitrary dimensions sampled from a texture T' which perceptually lies between T_1 and T_2. This is controlled by a user specified interpolation amount $\alpha \in [0, 1]$, where 0 corresponds to T_1 and 1 to T_2. Based on the common assumption that a texture can be modeled as a Markov-Random-Field, it is possible to synthesize a new texture only from a collection $\mathcal{N} = \{N_j\}_{j=1...M}$ of local neighborhoods extracted from the input image. Since these local neighborhoods characterize a texture completely, interpolation between textures can also be regarded as the creation of a new interpolated set of

```
Function: TextureInterpolation($I_1$, $I_2$, $F_1$, $F_2$, $\alpha$ )
Add distance channel to $I_1$ and $I_2$
Initialize image $I$ with noise
foreach resolution $r$ and neighborhood size $s$ do
    $\mathcal{N}'$ = InterpolateNeighborhoods($I_1$, $I_2$, $F_1$, $F_2$, $\alpha$,$r$,$s$)
    for $i = 1...N$ do
        Find nearest neighbors in $\mathcal{N}'$ for patches in $I$
        Update neighborhood centers via relaxation
        Update pixel colors in $I$ via Mean-Shift
        Perform statistical synthesis
    end
    return $I$;
end

Function: InterpolateNeighborhoods($I_1$, $I_2$, $F_1$, $F_2$, $\alpha$, $r$, $s$ )
foreach $l = \{1, 2\}$ do  $\mathcal{N}_l$ = ExtractNeighborhoods($I_l$, $r$, $s$)

foreach $l = \{1, 2\}$ do
    $\hat{l} = 3 - l$
    foreach $N_j^{(l)} \in \mathcal{N}_l$ do
        Find $k$ nearest neighbors $N^{(\hat{l})}_{c^{(l)}(j,1)}, \cdots, N^{(\hat{l})}_{c^{(l)}(j,k)}$ of $N_j^{(l)}$
        foreach $i = 1 \cdots k$ do
            Find warping $W$ between $N_j^{(l)}$ and $N^{(\hat{l})}_{c^{(l)}(j,i)}$
            Create interpolated patch $N'$
            $\mathcal{N}' = \mathcal{N}' \cup \{N'\}$
        end
    end
    return $\mathcal{N}'$
end
```

Pseudocode 9.1: *Our texture interpolation algorithm*

neighborhoods. To interpolate between textures T_1 and T_2, we thus first extract the neighborhood sets $\mathcal{N}_1$ and $\mathcal{N}_2$ from I_1 and I_2. Based on these, a new set of interpolated neighborhoods $\mathcal{N}'$ is computed which is then used to synthesize I'.

In Pseudocode 9.1, an overview of the different steps of our algorithm is given. More details on the individual steps are then given in the following sections.

To obtain $\mathcal{N}'$, we search for each $N \in \mathcal{N}_l$, $l \in \{1, 2\}$, k corresponding neighborhoods in the other set $\mathcal{N}_{\hat{l}}$, with $\hat{l} = 3 - l$. The interpolated neighborhoods are then created using a warping transformation to first align features between N and these k corresponding neighborhoods and then linearly blending the patches. For this, we use binary feature maps F_1 and F_2 marking the positions of relevant features in both input images which should be aligned during the interpolation.

Our texture synthesis is based on the texture optimization algorithm of Kwatra et al. [KEBK05]. The texture is synthesized in multiple steps from coarse to fine with successively increasing image resolution and decreasing patch size to allow the algorithm to synthesize textures with features at different scales. For each combination of resolution and patch size, we create the set of interpolated neighborhoods $\mathcal{N}'$ from the input textures. Compared to ordinary texture synthesis, the set $\mathcal{N}'$ is less consistent due to the warping of the individual patches. In particular, it is no longer guaranteed that for each patch neighboring patches which are identical in the region of overlap exist. To compensate for this, we therefore do not employ a regular grid of patches during the synthesis, but allow for a more irregular distribution of the pasted neighborhoods. In addition, a relaxation step moves the pasted neighborhoods in such a way that the error between overlapping neighborhoods is minimized. Finally, to preserve high-frequency details, we perform an additional statistical synthesis step based on the technique described in [PS00] in each iteration of the algorithm.

9.3 Neighborhood Interpolation

The patch interpolation creates an interpolated set $\mathcal{N}'$ of neighborhoods by finding corresponding neighborhoods in the two input textures I_1 and I_2 and then interpolating between them.

9.3.1 Correspondence Search

First, the sets $\mathcal{N}_1$ and $\mathcal{N}_2$ of all possible neighborhoods of the desired size are extracted from I_1 and I_2. To find corresponding patches, we then search for each $N_j^{(l)} \in \mathcal{N}_l$, $l \in \{1, 2\}$, the nearest k corresponding neighborhoods in the other set $\hat{l}$: $N^{(\hat{l})}_{c^{(l)}(j,1)}, \dots, N^{(\hat{l})}_{c^{(l)}(j,k)} \in \mathcal{N}_{\hat{l}}$, where $c^{(l)}(j, i)$ is the index of the i-th nearest neighbor of $N_j^{(l)}$ in $N^{(\hat{l})}$. For this, we use two binary feature maps F_1 and F_2. These maps could be generated automatically using a feature detector. However, depending on the input images very different elements might constitute features.

We tried both the compass operator [RT99], which was suggested in [MZD05] for this purpose, as well as several edge detectors. Still, for many images the results were not very satisfactory and thus we so far use manually created feature maps.

Ideally, we would like to find corresponding neighborhoods which have features that can be aligned under minimal deformation. For instance, a good distance measure would probably be the warping error D_W given in Equation (9.1). However, computing all pairwise warpings is prohibitively expensive and cannot be performed in reasonable time. A faster alternative is to compute the difference of the binary feature channels. However, this results in large errors even for only slightly misaligned features. Thus, as was suggested in [LH06], we perform a distance transform to add an additional distance channel to the neighborhoods, in which for each point the distance to the nearest feature point is stored. The L_2 error in this distance channel is a considerably better measure of how well the features in the two images correspond to each other. We use the metric $\|W_C \bullet (N_i^{(1)} - N_j^{(2)})\|_2$ to compute the distance between two neighborhoods $N_i^{(1)}$ and $N_j^{(2)}$. Here, W_C specifies weights for the channels of the neighborhood. We thus include both the color and distance channels during the search to prefer warpings between patches with similar color and feature distribution. Using this distance measure, the k nearest patches can easily be found via PCA accelerated KD-tree search and thus a very fast lookup becomes possible.

It is important to perform these searches for the neighborhoods in both textures, first using $\mathcal{N}_1$ as input and searching in $\mathcal{N}_2$ and then the other way around. If the searches are not performed symmetrically it may be that many neighborhoods in the second image are not nearest neighbor to any of the neighborhoods from the first texture and would not be taken into account at all. This is a problem, since we want to ensure that the extracted patches are not only similar to the input images but also complete in the sense that everything visible in the two input images is also represented in the extracted patch set. In [WHZ*08] and [SCSI08], this problem is described in more detail in the context of the creation of image compactions.

9.3.2 Patch Interpolation

Once two corresponding neighborhoods $N_j^{(l)}$ and $N_{c^{(l)}(j,i)}^{(\hat{l})}$ have been found, we perform a color adjustment. To this end, the mean and variance of the color channels in both input textures are computed, and then linearly interpolated according to α. The color values of the two neighborhoods are then shifted and scaled accordingly.

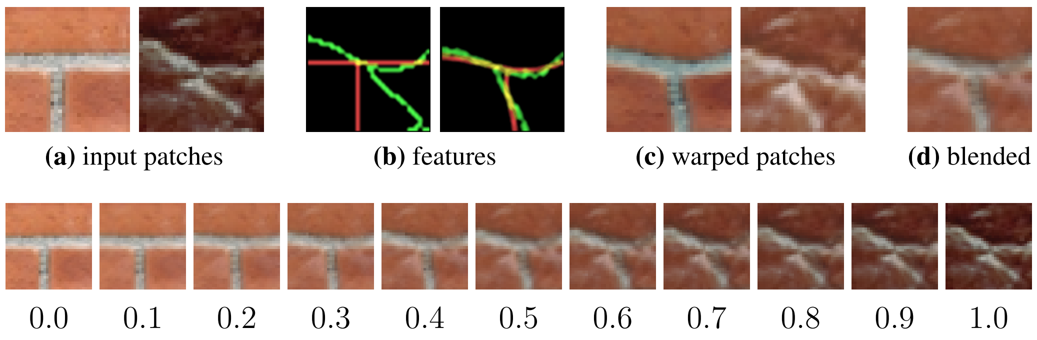


Figure 9.2: *Illustration of the patch interpolation. The two input patches are shown in (a). In (b), the feature maps of the two patches are shown overlaid before and after the warping for $\alpha = 0.5$. The resulting warped patches are shown individually in (c) and then after blending in (d). The full interpolation sequence is shown in (e).*

The interpolation is then performed by first warping the patches and then blending the warped images to obtain the interpolated result. This is illustrated in Figure 9.2. To compute the feature aligning warping, we search for a *thin-plate spline* (TPS) transform θ [Duc77] that maps the features in one neighborhood onto features in the other neighborhood. We create two point sets $P^{(1)}$ and $P^{(2)}$ from the sections of the two feature maps F_1 and F_2 which correspond to the currently processed neighborhoods. For this, we store the coordinate of each non-zero pixel in the point sets ($P^{(i)} = \{(x, y)|F_i(x, y) \neq 0\}$). We then use the algorithm of Chui and Rangarajan [CR00], which uses deterministic annealing to simultaneously find the correspondence between $P^{(1)}$ and $P^{(2)}$ and the aligning transform.

The algorithm iterates between updating the correspondences between points and calculating a new TPS according to the found correspondences. It is based on soft assignments, captured in a matrix M. Each entry m_{ij} of M gives the probability that the pair of points $P_i^{(1)}, P_j^{(2)}$ correspond to each other. The algorithm thus minimizes the error function

$$D_W(M, \theta) = \sum_i \sum_j m_{ij} \|\theta(P_i^{(1)}) - P_j^{(2)}\|^2 + R(\theta), \tag{9.1}$$

where R is a regularization term penalizing strong deformations. During the annealing, as the temperature parameter is decreased, the soft assignments in M become successively more distinct until they finally converge against one-to-one correspondences. This technique allows to find the correspondences in a considerably more robust way than algorithms such as *iterated closest points* [BM92], which are based on hard assignments only. To obtain the interpolated patches, we compute both TPS transformations, $\theta_{1,2}$ mapping P_1 to P_2 and $\theta_{2,1}$ mapping P_2 to P_1, during the point matching algorithm, using the same correspondences for both. The interpolated transforms are given by interpolating the identity transform and the TPS, i.e. $\theta'_{1,2} = (1-\alpha)\theta_{1,2} + \alpha\mathbf{I}$ and $\theta'_{2,1} = \alpha\theta_{2,1} + (1-\alpha)\mathbf{I}$. These two transformations are then used to warp one patch by α and the other patch by $1-\alpha$ to align the features and the following blended patch is added to $\mathcal{N}'$:

$$N'(x) = (1-\alpha)N_1(\theta'_{2,1}(x)) + \alpha N_2(\theta'_{1,2}(x))$$

Even though we only consider patches, especially for larger neighborhood sizes it is still possible that one patch has features that cannot be mapped onto any feature in the other patch. Thus, we need a robust handling of outliers during the computation of the TPS. We slightly modified the original algorithm by assigning outlier points to a point at their own position during the last iteration for the computation of the TPS. This adds a certain inertia to the iteration, avoiding large deformations that align outliers with very distant features. In our experiments, this improved the robustness of the point matching.

We only want to use those neighborhoods during the synthesis which achieve a good alignment of the warped feature maps and for which the actual images fit well to each other. Thus, we compute the similarity between the two warped neighborhoods and only keep those where the similarity is above a certain threshold. For this similarity, we combine the difference of the two warped distance channels and the normalized cross correlation of the color channels. The relative weight of distance channel and color channels is a user selectable parameter controlling the importance of the feature channel in relation to the images themselves. Since it is rather difficult to decide in advance on a similarity threshold, we instead use a certain percentile of the similarities of all patches as threshold. To improve the quality of the results, it is possible to choose a high value of k, to create a large number of candidate neighborhoods, and then use a small percentile to keep only the best ones. However, we found that it is usually sufficient to use only one neighborhood ($k = 1$) and then keep the better 50% of the generated neighborhoods. All images shown in this thesis were generated this way.

9.4 Texture Synthesis

The interpolated patches are then reassembled via texture synthesis. For this, a neighborhood-based synthesis approach and a statistical texture synthesis approach are combined.

9.4.1 Neighborhood-based Synthesis

Since we perform the actual interpolation of the image on the neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$, any synthesis algorithm which works with a collection of neighborhoods $\mathcal{N}'$ as input could be used together with our interpolation technique. We decided to use texture optimization [KEBK05] since it achieves high-quality results and iteratively optimizes the whole texture to avoid the accumulation of errors. This is especially important in our case, since the input might contain incoherent neighborhoods which cannot be combined with any other neighborhood in a sensible way. When no interpolation is performed, all possible neighborhoods are extracted from the input image. This means, that the input patches can be consistently overlapped. However, when interpolation is used, this is not necessarily the case. When two neighborhoods, which are overlapping in one image, are warped against different patches in the other image the resulting patches are no longer consistent. Therefore, a rather tolerant synthesis algorithm, which can compensate for these patches, is needed.

The texture optimization algorithm iterates over different combinations (r, s) of image resolutions, obtained by downsampling the input images I_1 and I_2 by the factor r, and neighborhood sizes s to be able to synthesize features on different size scales. For each of these combinations, we independently create a new set of interpolated neighborhoods $\mathcal{N}'$. Depending on the size of features in the input images, different choices of r and s might be necessary. We usually perform the synthesis in three steps with $(r = 2, s = 33), (r = 2, s = 17), (r = 1, s = 17)$. Only for images with very large and regular features, we use an additional iteration with $(r = 2, s = 49)$.

For a given combination (r, s), an image is synthesized by repeatedly choosing matching neighborhoods from $\mathcal{N}'$ for a set of center points in the current synthesis result I. Then I is updated by pasting the matching patches into I while averaging their contributions in the respective regions of overlap. In the original algorithm, the matching is performed for neighborhoods centered around points located in a regular sparse grid, in such a way that for each pixel several overlapping neighborhoods are available. However, using a regular grid does not give good results in our case. In traditional texture syntheses, all possible neighborhoods are extracted from the input image. This means, that input neighborhoods are available for all possible positions of a patch center. However, when using the interpolated neighborhoods, it might happen that a good patch is available for one center point, but not for the center directly adjacent to this point. When this is not taken into account during the synthesis, the synthesized images become blurred and only a small subset of the available neighborhoods is used, resulting in repetitive images. Thus, we do not search for neighborhoods centered around points on a regular grid,

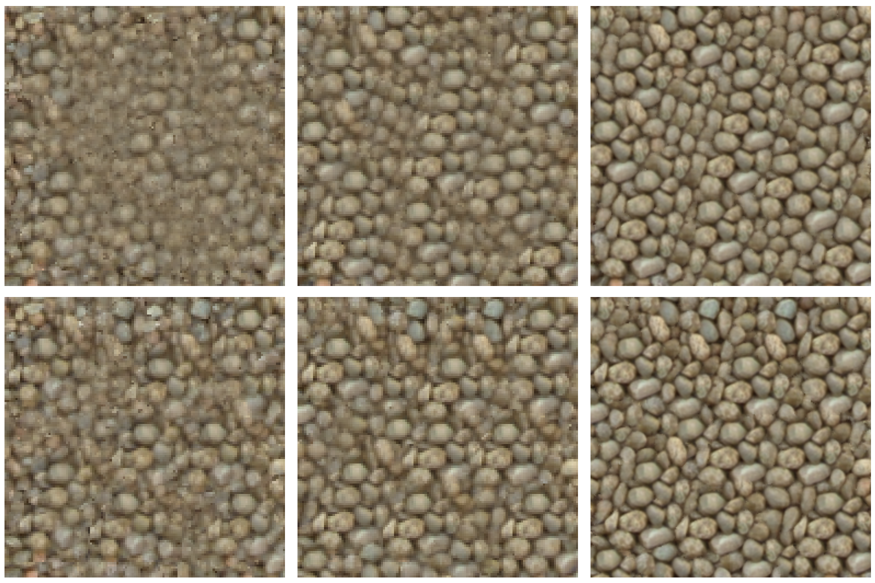


Figure 9.3: *Difference between neighborhoods aligned to a regular grid (top) and using irregular neighborhood centers (bottom) after each of the synthesis iterations.*

but instead use an irregular distribution. Within each cell of the original grid, we search the one neighborhood center C_i which results in the lowest error and then use this point instead. This way, we can ensure that each pixel is overlapped by enough neighborhoods, but on the other hand allow for a more flexible placement of the neighborhoods. Unfortunately, this slows down the synthesis, as a higher number of neighborhood searches is needed. Still, as can be seen in Figure 9.3, it improves the quality of the synthesized result considerably.

To improve the coherence between the available neighborhoods in $\mathcal{N}'$, we also looked into enforcing spatial coherence during the interpolation by assigning neighborhoods adjacent in one image to neighborhoods adjacent in the other image and using similar warpings for both. However, we did not notice a considerable difference in the resulting textures.

After the selection of the neighborhood centers C_i and the corresponding patches, we perform an additional relaxation step. Here, the matched neighborhood patches are shifted relative to each other in such a way, that the difference between overlapping parts is minimized. For this, for each neighborhood center the optimal position within a small search radius of a few pixels is searched and each neighborhood is shifted to this new position. To prevent the centers from moving too far apart, we add an additional spring term which penalizes deviations from the old position

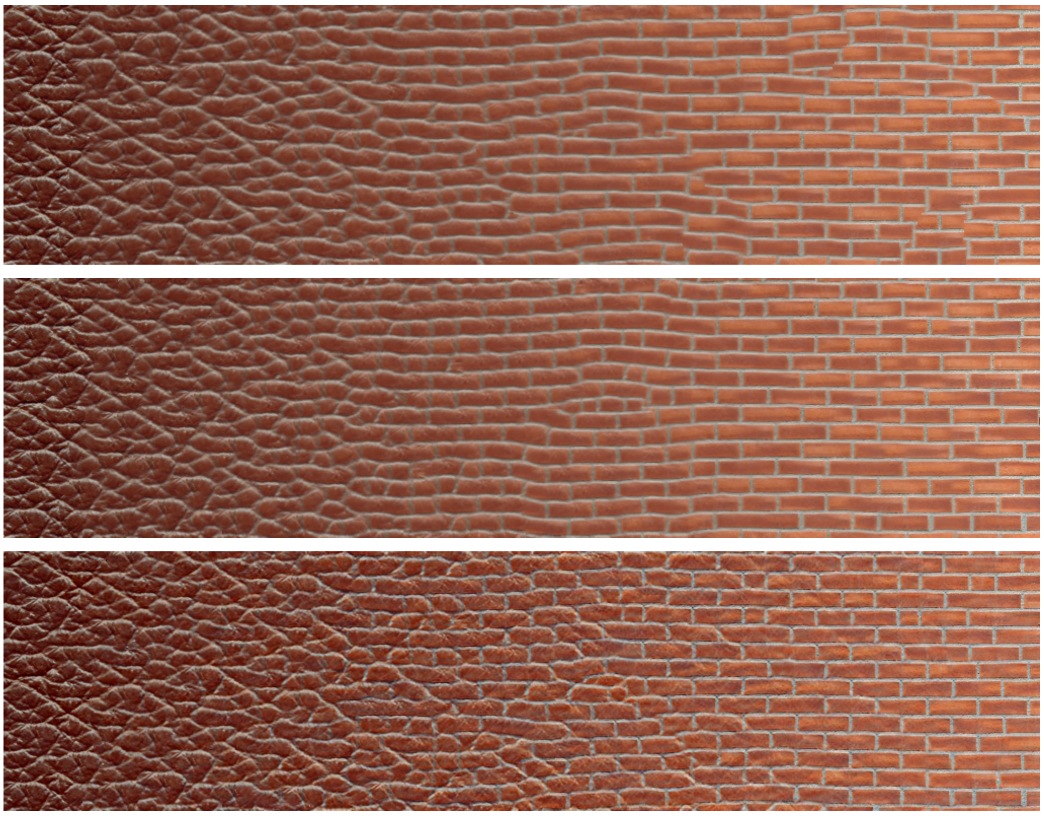


Figure 9.4: *Improvements obtained via relaxation (center) and additional statistical synthesis (bottom) compared to the result (top) without these additional steps.*

relative to the neighbor patches. This process is repeated until a local minimum is reached for all centers. This relaxation process improves the alignment of the patches, especially when there are regular structures in the texture (see Figure 9.4, central image). Though the relaxation step helps, the synthesis result still sometimes show discontinuities.

In the next step, the synthesized image is updated by averaging the pixel value of the overlapping neighborhoods. Here, we use iteratively reweighted least squares to obtain weights for the neighborhoods, as was suggested in [KEBK05]. Instead of averaging the colors, we perform an additional Mean Shift clustering [CM02] for each pixel to find the dominant mode and use the center of this mode as new color, as was suggested in [KFCO*07]. This helps to obtain sharper images and to cope better with incoherent patches.

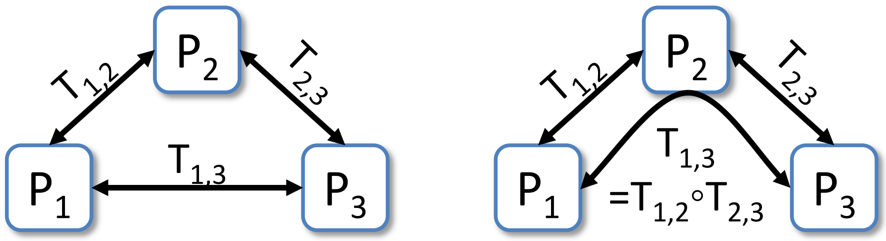
9.4.2 Statistical Synthesis

Finally, we perform a statistical synthesis step, which helps to preserve high-frequency details, which are otherwise easily lost during the patch interpolation. For this, we use the synthesis algorithm from [PS00] which works by enforcing image statistics and cross correlations between bands of a steerable pyramid decomposition of the input image. For each iteration of our synthesis loop, we perform one iteration of the statistical synthesis algorithm, using the implementation kindly made available by Portilla and Simoncelli on their homepage `http://www.cns.nyu.edu/~lcv/texture/`. We slightly extended the original implementation, which only works for greyscale images, by adding an additional cross correlation between the three color channels for each band of the pyramid. This correlation is then also enforced during the synthesis. Currently, we use linear interpolation to obtain the image statistics. A simple linear interpolation was already described in [PS00] and often resulted in rather unsatisfactory results when used directly for texture synthesis. Thus, further research on how the image statistics should be interpolated correctly is certainly necessary. Still, as shown in Figure 9.4, when used together with our neighborhood-based synthesis result as initialization, the additional statistical synthesis step helps to preserve high-frequency details in the interpolated textures.

9.4.3 Spatially Varying Interpolations

To create spatially varying interpolations, we create interpolated patches for several different interpolation amounts α (we use 20 equidistantly spaced steps). We then encode the α values in an additional channel of the patches and use all patches together during the synthesis. By enforcing this channel during synthesis to resemble the user-supplied distribution mask, we then generate the spatially varying textures. The weight for this channel controls during synthesis whether the distribution mask is represented exactly, or whether smooth transitions and better synthesis results should be favored by the algorithm.

We currently perform the statistical synthesis independently for several equidistantly sampled values of α, each time only on the relevant parts of the image according to the distribution map. Then, the results are blended using blending weights derived from α. This approach is rather slow, but worked quite well in our experiments and did not require large changes to the original synthesis algorithm.



(a) direct **(b)** chaining

Figure 9.5: *Proposed chaining of warping transformations. Instead of directly computing all pairwise warpings (a), we only compute warpings between neighboring patches and create missing warpings by concatenating the transformations (b).*

9.5 Multi-Material Interpolation

So far we have only considered the interpolation between two textures. To allow for a real texture design, it is necessary to enable the interpolation between k different texture samples. For this, we search for k-tuples of patches that are corresponding in all k input textures. All input patches in one tuple are then warped and blended to form one interpolated patch. However, the straight forward extension to compute all pairwise warpings to register the patches to each other would not be efficient. Instead, we only use an approximation. First, we arrange the materials and then sequentially compute the warping transformations only between direct neighbors. The transformations between all pairs of patches in a tuple are then obtained by chaining the computed warpings (see Figure 9.5). To create the interpolated patch, the resulting warping transformations are linearly combined according to the interpolation weights and then applied to the k input patches. Finally, the warped patches are blended together.

When creating textures with spatially varying interpolation weights, a further adaption is necessary for the statistical synthesis. When only two textures have to be interpolated, we perform the statistical synthesis independently for several equidistantly sampled values of α and used linear blending between the resulting images to obtain the final image. However, this would not scale well to the interpolation between several materials, as sampling the higher-dimensional space of interpolation weights equidistantly would be very expensive. Instead, we utilize clustering on the input image with the spatially varying weights to find a smaller set of representative weights. For each of those clusters, the statistical synthesis is performed. Finally the results of this synthesis are blended. The blending weights are obtained by computing a Delaunay triangulation and then using the

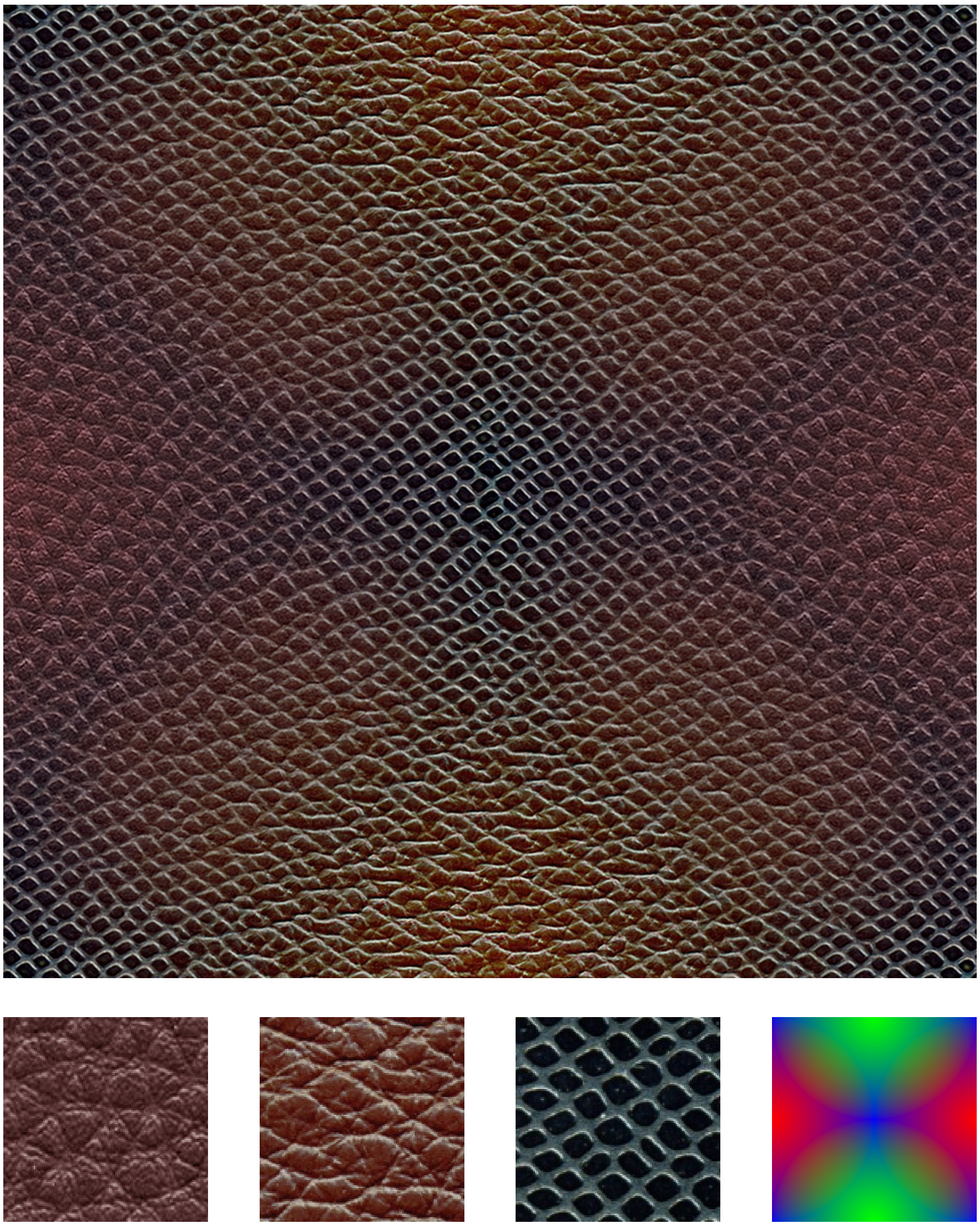
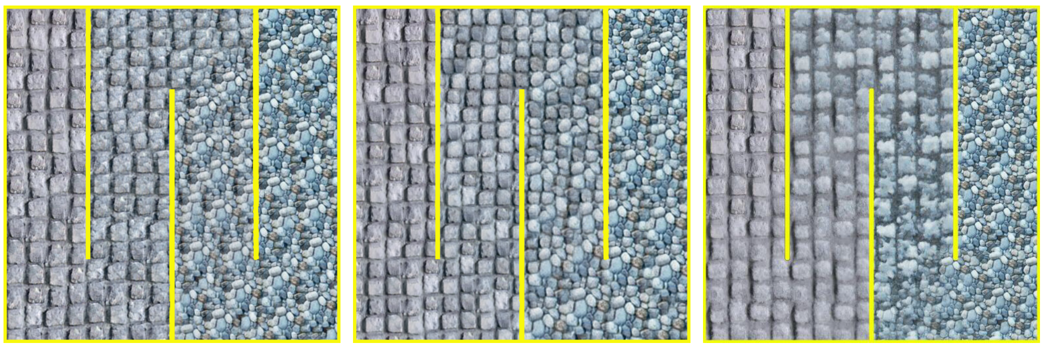


Figure 9.6: *Interpolation between three textures according to a complex spatially varying distribution map. The input images and the color coded distribution map are shown in the bottom row.*



(a) Without scale compensation **(b)** With scale compensation **(c)** [RLW*09]

Figure 9.7: *Comparison between our results without and with scale compensation and those from [RLW*09].*

barycentric weights in the corresponding simplex. Figure 9.6 shows an example of the interpolation between three textures.

9.6 Compensation for Different Feature Scales

A limitation of the approach described so far is the interpolation between materials with large differences in feature scale. Since the computed warping transformation is as rigid as possible, an enlargement or shrinking of features does not take place. To achieve this effect, we additionally compute one global scaling factor for each of the participating materials. This factor is derived from the average value of the channel containing the distance to the feature map and can then be used to bring the different image scales into alignment. During the interpolation, we determine the scale of an interpolated patch by weighting the scales of the input patches according to the interpolation weights. Prior to the computation of the warping transform and blending, all input patches are then scaled to correspond to this interpolated scale. Similarly, prior to the computation of the image statistics during the statistical synthesis, the input images are also scaled to this interpolated scale.

In Figure 9.7, we show one of the examples from [RLW*09] which has a considerable difference in the feature scales of the two input images. Without the scale compensation, our interpolation produces rather unintuitive results since the features cannot be brought into good alignment. In contrast, when additionally scaling the patches prior to interpolation, the size of the features continuously increases during the interpolation sequence and the feature can be brought into good alignment.

9.7 Results

In Figure 9.8, we show several examples of texture interpolation sequences obtained with our algorithm. The corresponding input images and feature maps are shown together with the results. As can be seen, the algorithm is able to create seamless transitions, which remain sharp and detailed during the whole sequence, for a wide range of different materials. Even when there are considerable differences in the topology and structure of the features, such as for example in the transition between the leather and the brick wall, a continuous and plausible interpolation is obtained.

In Figure 9.9, we show several materials designed by interpolating between two samples of one material class. Apart from the failure case in the lower right, the algorithm is able to create plausibly looking materials, which perceptually lie in between the input textures. Both the feature structure as well as the material details are interpolated.

In Figure 9.7 and 9.10, we directly compare our results with those reported in [RLW*09], a recently published texture interpolation technique based on texton masks. Our algorithm is able to create smooth and detailed interpolation sequences, in which there is a continuous and seamless transition between the structures of the two materials. The advection algorithm from [RLW*09] interpolates the whole texton mask, which results in completely new structures not present in either of the input images. In contrast, our approach works on a more local level and creates a more direct transition with intermediate images which adhere considerably more to the input textures.

In Figure 9.11, we show an example for a spatially varying texture with a more complex distribution map. It can be seen, that not only the color and fine-structure of the material is varied spatially, but also the feature shape, which is interpolated from a irregular elongated pattern to a more regular triangular one. Furthermore, discontinuities are handled plausibly, with features continuing over the discontinuities.

Since we perform the warping and interpolation on the individual patches, our technique is mainly suited for materials with regular or semi-regular structures, which can be characterized by these patches and for which an alignment of corresponding features is possible. For materials with large-scale structures or with purely stochastic behavior, different techniques are probably better suited. Furthermore, it is required that the material can be characterized via the feature mask. The interpolation between two wood samples in Figure 9.9 shows a case where the feature map does not characterize the material well and where the structure of the features is very different for the two samples.

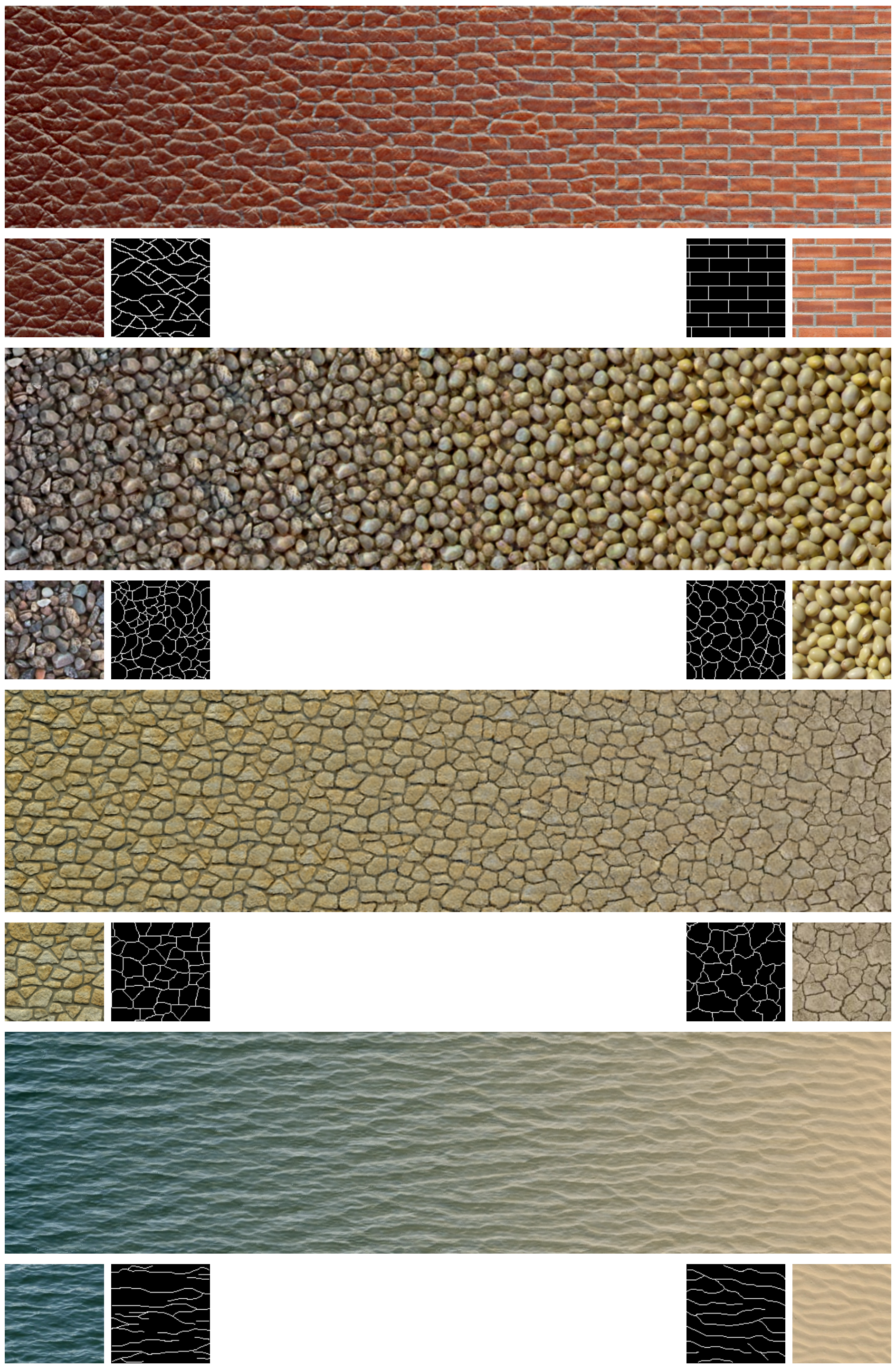


Figure 9.8: *Sample interpolation sequences that were created with our technique. Input textures and feature maps are shown.*

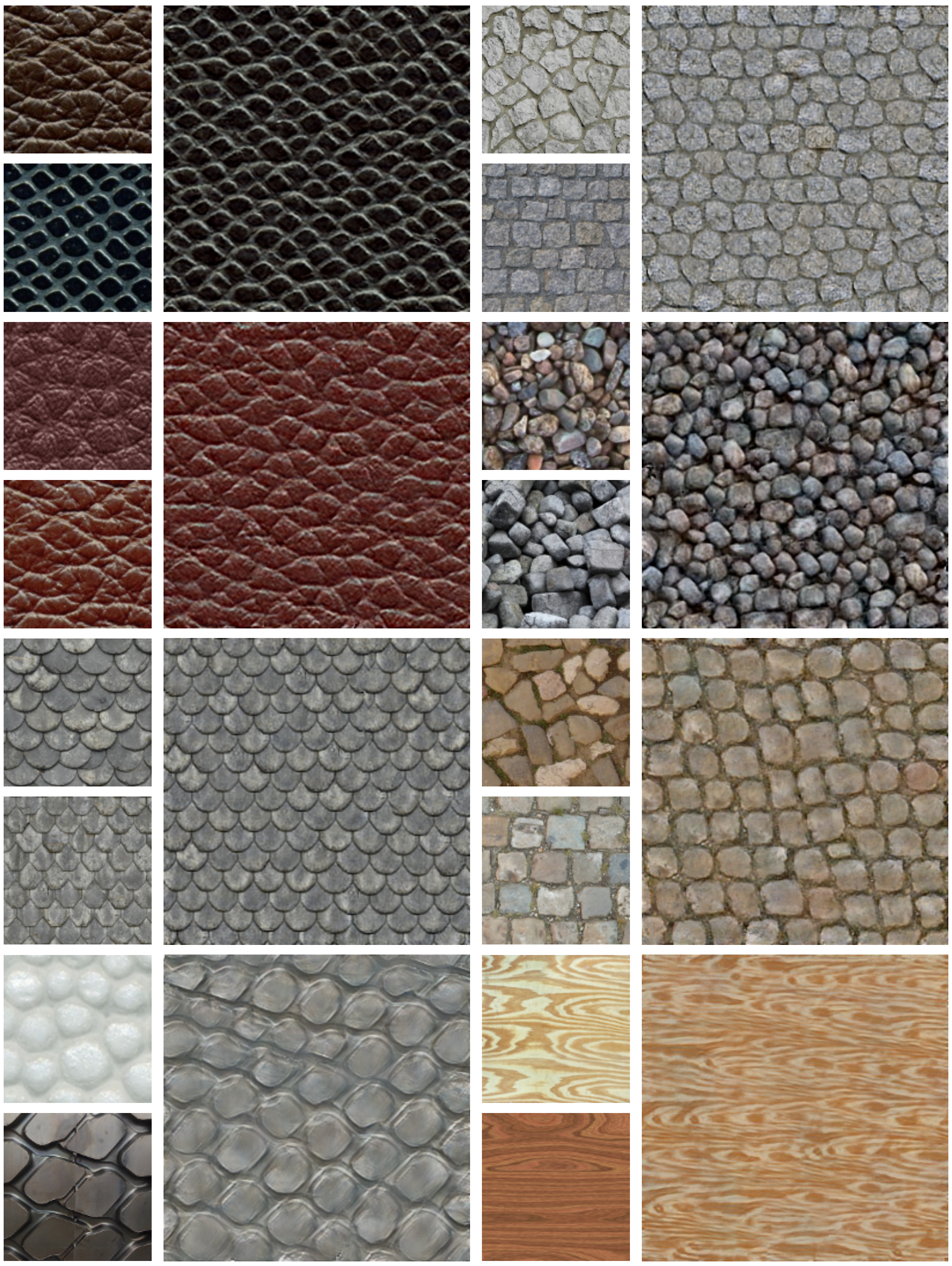


Figure 9.9: *Several new materials designed by interpolating between two texture samples (All images are for $\alpha = 0.5$). The lower right image shows a failure-case for a material which cannot be characterized well by line features.*

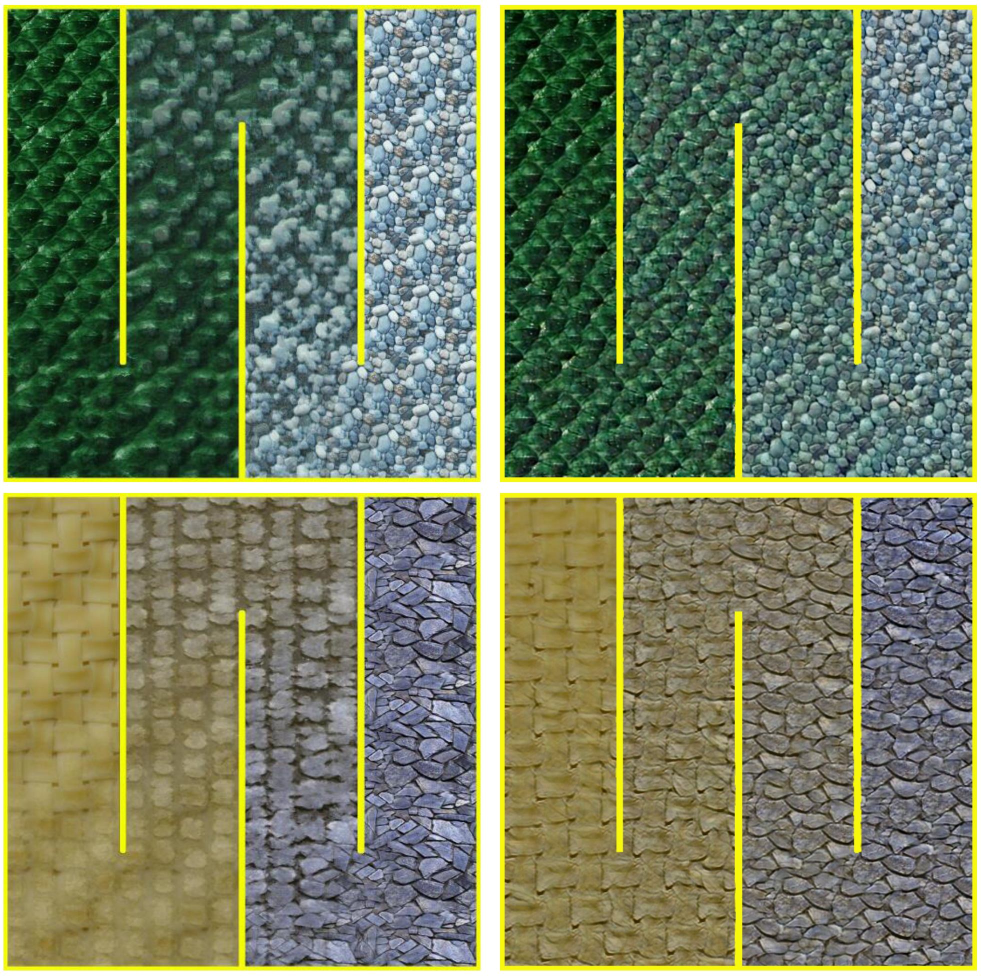


Figure 9.10: *Comparison of interpolation results from [RLW*09] with our results.*

After the initial publication of this work, two other techniques for texture interpolation have been published in [KPRN11] and [DSB*12]. The authors of these works compared their results with ours. In Figure 9.12, we include some of the comparisons from these papers. In contrast to our method, their techniques do not need a feature map. However, as the comparison shows, with a feature map our techniques is able to preserve the feature structure better and to create smoother interpolation sequences.

Using a Q6600 CPU, the computation of one of the sequences in Figure 9.8 (in a resolution of 1024x256 pixels) requires about 3.5 hours, using 10 iterations for each resolution and patch size. About 20 minutes are needed for the patch interpolation, which is already optimized moderately, being implemented in C++ and parallelized

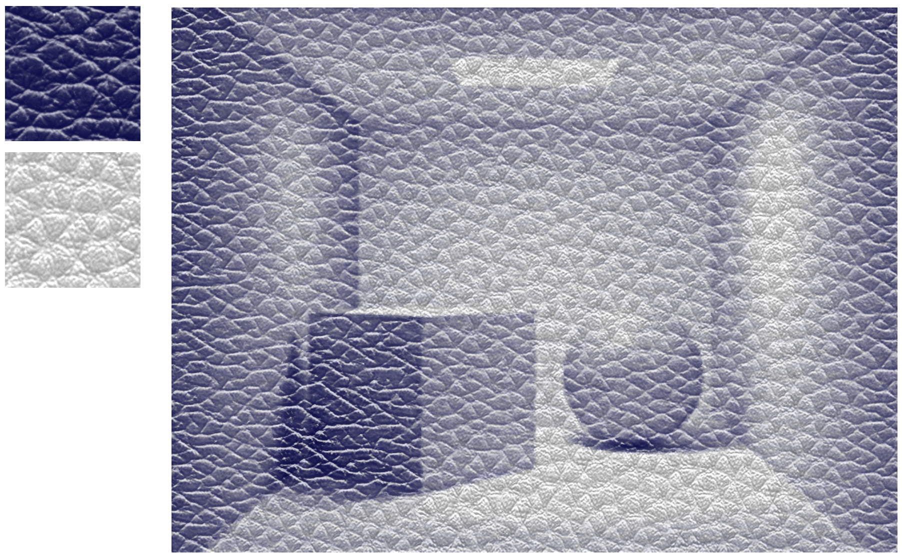


Figure 9.11: *Spatially-varying texture created by interpolating between the two shown material samples using a grayscale image of a Cornell Box as distribution map controlling the interpolation amount α.*

on 4 cores. In contrast, large parts of the synthesis code were not parallelized and a MATLAB implementation of the statistical synthesis was employed.

9.8 Conclusion

In this chapter, we presented a novel texture interpolation algorithm, which locally warps and blends individual neighborhoods of the input textures and then reassembles the image using the texture optimization algorithm. In contrast to techniques based purely on warping, we can create consistent images this way, even when it is not possible to continuously warp one input image onto the other. By combining this algorithm with statistical texture synthesis we are furthermore able to preserve high-frequency details.

A interesting avenue of research would be a perceptual reparameterization of the resulting interpolation sequences. Often, the sequences do not seem to linearly interpolate between the two materials, but instead keep the characteristics of one of the two materials for a rather long time. A perceptual study investigating these effects might lead to techniques that reparameterize α in a way which allows for more linear transitions.

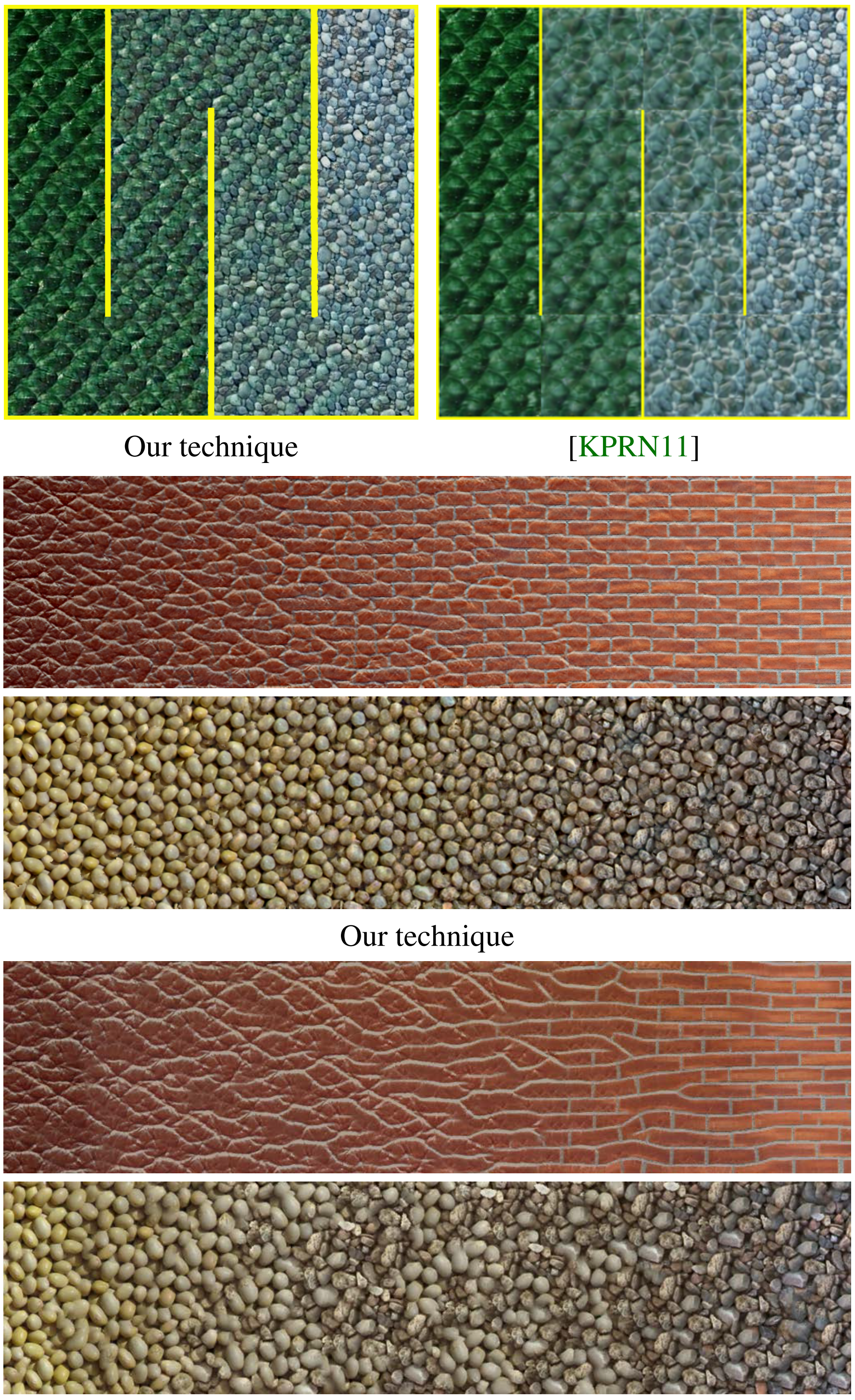


Figure 9.12: *Comparison with the results from [KPRN11] and [DSB*12].*

CHAPTER 10

BTF Interpolation

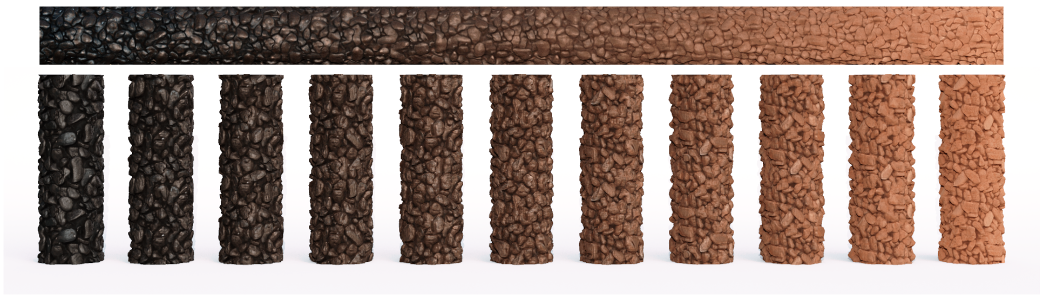


Figure 10.1: *Example BTF interpolation sequence*

In this chapter, we will extend the texture interpolation approach from the previous chapter to enable the interpolation of BTFs. In contrast to textures and even SVBRDFs, BTFs are far more complex datasets. This is due to the fact that BTFs are capable of representing effects arising from the inherently contained light interaction with the surface geometry, such as parallax, shadowing, masking, rotated lobes attributable to normal variations, interreflections and sub-surface scattering. These effects can be non-local, i.e. the appearance at one point on the surface is influenced by other parts of the material.

To describe the dependence of the appearance on the underlying material surface, we distinguish three major scales. First, there are the clearly dominant *material features*, such as the year-rings in wood, bumps in leather or contours in stones, ranging over several texels of a surface texture. Second, there are the micro-geometry scattering effects, i.e. *reflection properties*. These correspond to a feature size far below the size of one texel and are commonly described by their statistical properties, e.g. a BRDF. Last but not least, there are other fine geometric details in the size just below a fraction of a texel but still clearly recognizable as individual features. Examples are small fibers in wool yarn or small holes in the surface of an eggshell, leading to its cavernous appearance. These are called *meso-scale effects*.

Due to this inherent complexity of the material surface, BTFs are usually not modeled explicitly but instead measured from real-world samples, which is a time consuming and expensive process that requires specialized hardware equipment, such as the devices described in Section 2.1. To overcome the restriction to measured BTFs, several methods have been devised to edit and manipulate BTFs in recent years allowing for impressive material edits. Despite this progress in BTF editing, there are currently still no methods that allow interpolating between given BTFs on all three different scales in a consistent way. An adaption of the texture interpolation approach from Chapter 9 to BTFs, could lead to similar intuitive schemes for the creation of new content based on exemplars as we demonstrated in Chapter 8 and as those presented for BRDFs and textures in the works by Matusik et al. [MPBM03a, MZD05]. These techniques allow a user to navigate the space of materials spanned by a few measured exemplars. This paradigm is desirable, as it simplifies the design process of new materials significantly. The user can manipulate the whole material appearance at once according to intuitive, predictable and possibly semantically meaningful directions in this space (e.g. "more similar to this leather"). This is in contrast to other editing techniques where one is concerned with individual aspects such as changing parameters (e.g. specularity, roughness, Fresnel effect) or rearranging the distribution of features.

Even though a BTF can be regarded as a large collection of textures which depict the appearance of a surface under a set of different view and light directions, the interpolation of all aspects of the material appearance is a more complex task than texture interpolation. This is due to three reasons: First, with the BTFs showing different reflectance characteristics over the surface, an implicitly contained meso-scale geometry and different spatial distributions of material features, it is necessary to interpolate all of these in a consistent manner. Second, the representation contains many complex non-local effects as well as effects due to the implicitly captured geometry, which explicitly have to be taken into account during the interpolation process. Finally, the sheer amount of data that is stored in a BTF presents a technical challenge of its own.

In this chapter, we present a novel technique which allows a continuous interpolation between multiple BTFs such as the one shown in Figure 10.1, simultaneously interpolating all aspects of the material appearance. To the best of the authors' knowledge, there is only one publication [MSK07] tackling a similar problem. However, this approach is not able to interpolate material features but uses a procedural model for leathers to generate interpolants. We propose the separation of the BTF into a heightfield and a parallax-compensated BTF that is parameterized over the heightfield, removing parallax and shadows and bringing local frames into alignment. In this fashion, problems during synthesis caused by non-local and geometry effects are vastly reduced. The representation has the additional advantage that the silhouettes of objects are depicted in a more realistic way (compare Figure 10.6). Further, to cope with the large amount of data, all computations are performed within a factorized representation of the parallax-compensated BTF. We then utilize our texture interpolation algorithm from Chapter 9 to enable the simultaneous interpolation of reflectance, meso-scale geometry and material features. Since this algorithm is based on texture synthesis, it also has the ability to generate seamlessly tileable BTFs.

With these techniques at hand, we explore several applications such as the design of new materials by interpolation of measured BTFs, continuously spatially varying interpolation sequences between multiple materials, complex spatially varying interpolation patterns as well as separating the interpolation of material features from the interpolation of reflectance and fine meso-scale details. The effectiveness of our approach is demonstrated on a range of different material classes.

This chapter corresponds to the paper "Example-based Interpolation and Synthesis of Bidirectional Texture Functions" by Roland Ruiters, Christopher Schwartz, and Reinhard Klein, published in *Computer Graphics Forum (Proceedings of Eurographics)*, 32(2), 2013.

10.1 Previous Work

Several techniques for the editing of BTFs have been proposed. One class of techniques is based on analytic BRDF models that are fitted to the measured data, using either heightfields as in our approach described in Chapter 5 and in [MG09] or rotated local frames [WDR11] to approximate the underlying geometry. These BRDFs are then subject to editing operations that change reflectance characteristics like color, Fresnel effect or specularity, eventually editing the appearance of the fitted material. For SVBRDFs, similar edits have also been performed via a non-negative factorization [LBAD*06], which allows editing of a material via changes to 1D curves, characterizing its reflectance behavior. Alternatively, a completely data-driven approach [KBD07, XWT*09] uses image processing operations to enable edits such as color changes, making a material rougher or softer, changing specularity, warping or removal of shadows. These operations mainly influence the reflectance behavior or local aspects like roughness and shadows, but do not affect large-scale structures in the material. Another class of techniques that is capable of changing the spatial layout of the material features is based on the assumption that one input sample either consists of a composition of overlapping layers [LL11] or exhibits multiple weathering states [WTL*06]. The different features are identified by user-scribbles. The new material is then generated by redistributing these features along a user-provided map and using texture synthesis to fill in gaps.

Our approach also relates to several techniques for BTF synthesis. These are concerned with the creation of larger or tiling-free BTFs from small exemplars. Many existing approaches are based on textons [LM01, TZL*02, LHZ*04] or texture quilting [KM03, HH05, ZDW*05, KSOF05, LPF*07]. We refer to the comprehensive overview by Haindl and Filip [HF11] for more details. These techniques thus only rearrange the texels of the input BTF. This means, that the same transformation is applied to all view directions simultaneously, without compensation for view dependent parallax effects. If the BTF synthesis is performed for each view independently, it is necessary to enforce consistency between the views. Liu et al. [LYS01] do this by constraining the synthesis to a rendered heightfield, while Neubeck et al. [NZG04] utilize a firstly synthesized support-view to constrain the texture synthesis for all other views. However, performing a separate synthesis for each view is rather inefficient.

Mueller et al. [MSK07] synthesize two BTFs by re-ordering the texels according to a common heightfield-derived appearance space, but disregarding parallaxes. Once their features are aligned by the synthesis, the two materials are linearly blended to create an interpolated result. Here, the heightfield of the interpolated material needs to be known prior to the interpolation itself and is either user-specified or created with a procedural leather model.

None of these techniques provides a direct and automatic interpolation between two or more materials taking both the reflectance characteristics and the spatial structure of the material into account.

10.2 Approach

Since all the individual textures of the BTF depict the same surface from different view and light directions they are heavily interdependent. Applying an independent interpolation to each texture can lead to incorrect results. Therefore, a consistent interpolation of the textures is mandatory. Only if all images are interpolated consistently, a meaningful interpolation of the reflectance behavior is performed at the same time. One way to achieve this is to perform the synthesis on all images simultaneously. In theory, this would be possible with any available texture interpolation algorithm by considering the BTF as one image with an extremely large number of channels (many tens of thousands). In practice, such an approach would not be feasible because of the extremely large amounts of data that would have to be processed in this case. Instead, it is desirable to operate on a more compact representation.

A common technique for the compression of BTFs is a factorization via Singular Value Decomposition (SVD), (see Chapter 3 for a more detailed description). It takes advantage of the fact that a BTF can be represented compactly via a low-rank approximation. The function values are written as a matrix $\mathbf{M}_{i,j}$, where the row indices i encode the combination of view/light direction and color whereas the column indices j specify the position on the surface. For compression, a truncated SVD $\mathbf{M} \approx \mathbf{U}\Sigma\mathbf{V}^T$ is computed, where the matrices $\mathbf{U}$ and $\mathbf{V}$ each contain only c columns. The columns of $\mathbf{U}$ are called Eigen-ABRDFs and represent the angular reflectance behavior of the BTF and the columns of $\mathbf{V}$ are called Eigen-Textures and describe the spatial distribution of the reflectance. Koudelka et al. [KM03] observed that this representation can also be utilized for BTF synthesis and quilting was used to synthesize new Eigen-Textures.

We consider the matrix $\mathbf{V}' = \mathbf{V}\Sigma$, representing the spatial distribution of the reflectance as an image with c channels which serves as input for a texture interpolation algorithm. However, it is not a priori clear, whether performing an interpolation in this representation is valid. Fortunately, the SVD factorization has three important properties that potentially enable this approach: First, it preserves scalar products and thus distances under the L_2 norm. When a truncated SVD is used, distances are obviously not preserved exactly, but according to the Eckart-Young Theorem a truncated SVD is the best rank-c approximation of a matrix under the L_2 norm and thus the computed distances are good approximations. Second, a linear transformation which only affects the spatial domain can be applied directly to the matrix $\mathbf{V}'$. These transformations can be represented as a matrix $\mathbf{T}$, which is right-multiplied with the BTF matrix $\mathbf{MT} = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{T}$. The third important property is that linear combinations of two materials represented in the same basis can also be applied directly to the spatial distribution map, i.e. $\lambda_1\mathbf{U}\mathbf{V}_1'^T + \lambda_2\mathbf{U}\mathbf{V}_2'^T = \mathbf{U}\cdot(\lambda_1\mathbf{V}_1'^T + \lambda_2\mathbf{V}_2'^T)$.

10.2.1 Interpolation Algorithm

Following these considerations, one could choose any texture interpolation algorithm that is based on operations utilizing scalar products, L_2 norms, linear transformations in spatial dimensions and linear combinations. We will utilize our texture interpolation algorithm from Chapter 9. As was described in the previous chapter in more detail, the algorithm performs the following operations: First, the input textures are cut into small patches which are then brought into correspondence, warped and blended. To determine the correspondences, hand-drawn

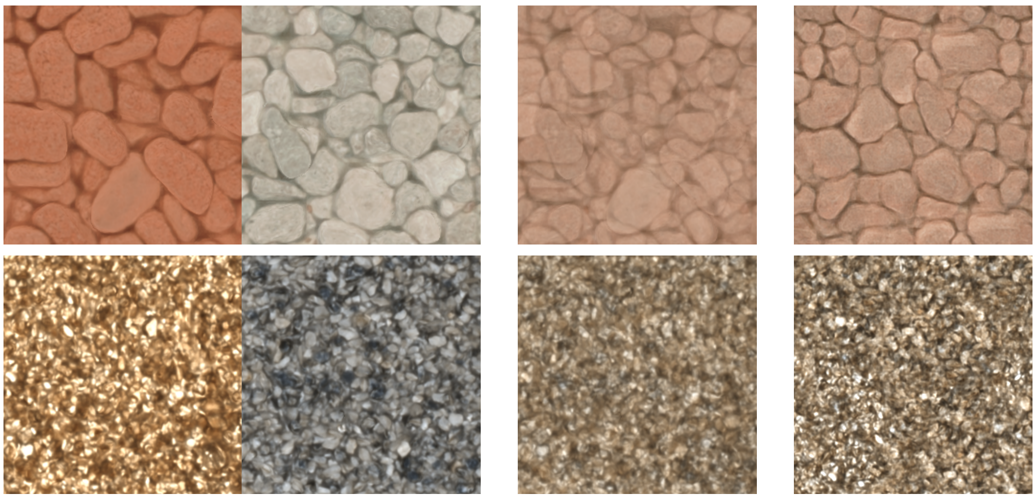


Figure 10.2: *Comparison of 50% interpolation results with simple alpha blending (b) and our material synthesis approach (c). All figures show a cut-out of the BTF with light and view from directly above. The upper row shows that our approach aligns the features of the input materials and creates a meaningful interpolation result. Even in the second case, which does not exhibit such clear features, our synthesis approach creates a much sharper result.*

binary feature maps are employed. The interpolation result is then generated by re-assembling the blended patches. Finally, a statistical synthesis step is performed to enforce interpolated image statistics of the input exemplars. Figure 10.2 shows that this interpolation technique, in contrast to linear blending, preserves features and even gives better results on unstructured materials e.g. sand.

Almost all parts of our algorithm are based on the set of operations allowed by the properties of the SVD. To determine the correspondences, similarity between the input patches under the L_2 norm is employed. During a warping transformation a warped image is created as a bi-linear interpolation of the pixel values of the original image. When the discrete values of the original image are unfolded into a row-vector this operation can be expressed as a multiplication with a matrix $\mathbf{T}$, which contains the four interpolation weights for one result pixel in the corresponding column. As argued in Section 10.2, it is valid to apply a matrix $\mathbf{T}$ to the factorized representation $\mathbf{V}$ to apply the warping to all BTF textures simultaneously. During the re-assembly of the blended patches, the L_2 distance is employed to find suitable overlapping patches. These are then again blended to create the final image. To this end, a mean-shift based clustering is used which is also based on the L_2 distance between the cluster center and the input pixels.

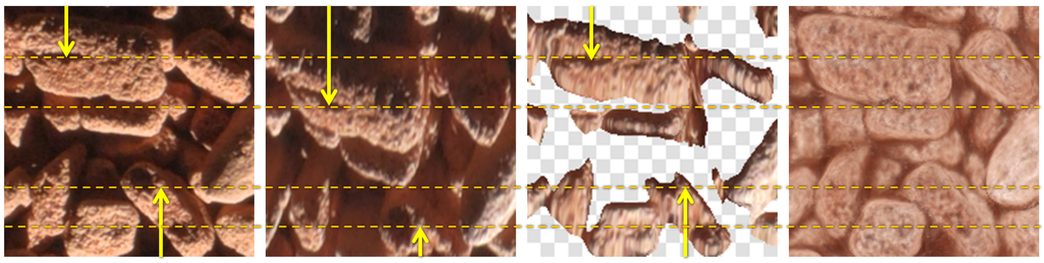
The only operation that cannot be transferred equivalently to the factorized representation is the final statistical synthesis step, based on [PS00]. This step restores certain image statistics of the input exemplars, such as pixel statistics and auto- and inter-channel correlations. However, it is a reasonable assumption that the statistics within the Eigen-Textures of the input data should also be present in the Eigen-Textures of the interpolated materials. Furthermore, the statistical synthesis tries to preserve the linear correlation between the channels. Thus, applying the statistical synthesis directly on the Eigen-Textures should also give reasonable results.

To enable the utilized linear combinations, it is mandatory that all BTFs participating in the interpolation are represented in the same Eigen-ABRDF basis $\mathbf{U}$. One way, this can be achieved is to compress all input exemplars with one SVD. However, it is also possible to compute a combined basis for several factorized BTFs and re-project the data into this basis without the need for re-compression. For more details on the merging of several SVDs, we refer to Section 3.1.2.

Please note that in this chapter, we do not directly use RGB values as input. Instead, we first apply the logarithm to our measured reflectance values prior to the SVD compression, as proposed by Matusik et al. [MPBM03a]. Accordingly, we apply the inverse operation after decompression. This has the advantages that it reduces problems with the dynamic range of the input dataset, which can be quite large for specular materials. Additionally, the non-negativity of the result is ensured. Furthermore, this way the interpolation follows the human brightness perception, which is according to the Weber-Fechner law logarithmic. However, in many cases this is not sufficient to obtain a perceptually linear interpolation of the whole material appearance. This is a very complex problem, even for a single material parameter, such as gloss [WAKB09], and even more difficult for materials with distinctly visible material features and thus should be further explored by future work.

10.2.2 Parallax Compensation

In most current approaches, BTFs are measured from planar samples and the data is then rectified by projecting it on a plane of reference. However, those parts of the material that significantly protrude from the reference plane will cause a parallax induced disparity in the textures (see Figure 10.3b). This means that the same texel represents different points on the implicitly captured surface geometry for different view directions. As long as these interdependencies between neighboring texels



(a) 0° view **(b)** 60° view, uncorrected **(c)** 60° view, corrected **(d)** 60° view, resampled

Figure 10.3: *Effects of parallax correction illustrated by a clay pebbles sample (feature height: $4.8\,\mathrm{mm}$). (a) displays the material as seen from above. (b), (c) show the same region at 60° declination. (b) illustrates the result without parallax compensation: Although a plane of reference that minimizes the disparity was chosen, features on the pebbles (yellow arrows) are not aligned with the ones in (a). (c) demonstrates the improved results obtained when using a proxy geometry (occluded, shadowed or back-facing pixels are masked), (d) finally shows the corrected content after resampling and hole-filling.*

are correctly preserved, e.g. during rendering, this is no problem. However, in the case of texture interpolation, the result is created by dividing the input image into small patches, which are warped, rearranged and finally blended, and thus these interdependencies are not necessarily preserved. As a consequence, the algorithm applies exactly the same warping and reordering operations to images depicting the material from all view directions. However, if these images contain strong parallaxes, material features in different view directions are not aligned with each other (see Figure 10.4). Therefore, a different transformation would be needed for each view in order to create a consistent result. Performing independent but constrained synthesis steps for each view (e.g. [LYS01]) would tackle this problem but prevent the use of a factorized representation and thus be far more expensive. Instead, we propose to compensate the parallaxes prior to the synthesis by re-projecting the BTF onto a proxy geometry and resampling it into local coordinate systems.

Using a non-planar reference-geometry yields a couple of technical obstacles that need to be overcome. First, it is necessary to represent the BTF in the local coordinate systems of the reference-geometry. For this, an angular resampling of the reflectance values needs to be performed. Otherwise, the blending of different patches would linearly interpolate between ABRDFs with different local coordinate systems. Furthermore, since a new geometry is created during the interpolation, the resulting BTFs would no longer be consistent with the local frames of the

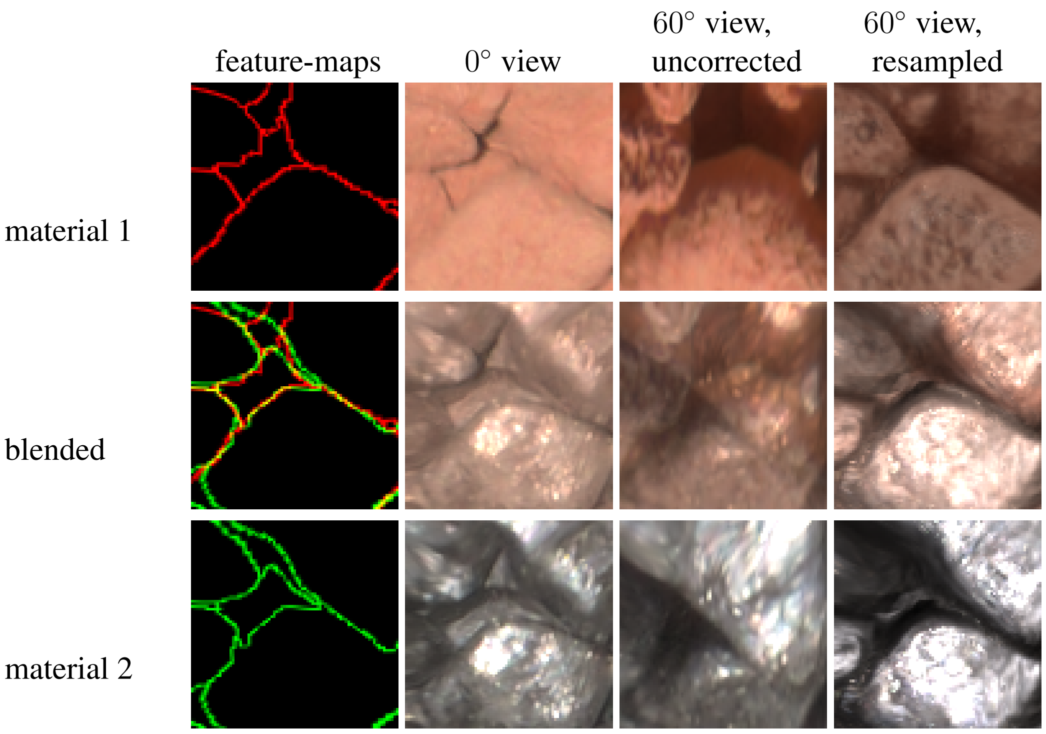


Figure 10.4: *Effects of parallax correction for blending two corresponding warped material-patches (top,bottom). The central row shows the blending between the patches. From the feature-map it becomes apparent that the automatic correspondence search and the warping manage to bring the features into alignment. The* 3^{rd} *and* 4^{th} *columns show that the uncorrected* $60°$ *views cannot be blended in a feature-preserving manner, but the resampled can. See Figure 10.6 for a rendering of the interpolated materials.*

generated geometry. Second, it is necessary to remove shadows from the input dataset and synthesize them based on the new geometry. Finally, holes in the re-projected images, resulting from occlusion and the shadow removal, have to be filled-in, based on information imputed from other parts of the measured data. For more details on these projection, resampling, shadow removal and hole filling steps we refer to our corresponding publication [SWRK11].

It is important to note that the result of this operation is not an SVBRDF (such as the one we created in Chapter 5 or was used in [MG09]), but a BTF parameterized over a different reference-geometry. Contrary to SVBRDFs, the reflectance functions in our proposed representation remain data-driven ABRDFs. This still enables the reproduction of non-local effects as well as meso-scale geometry that is not resolved by the reference-geometry. For the latter case, the re-projection removes

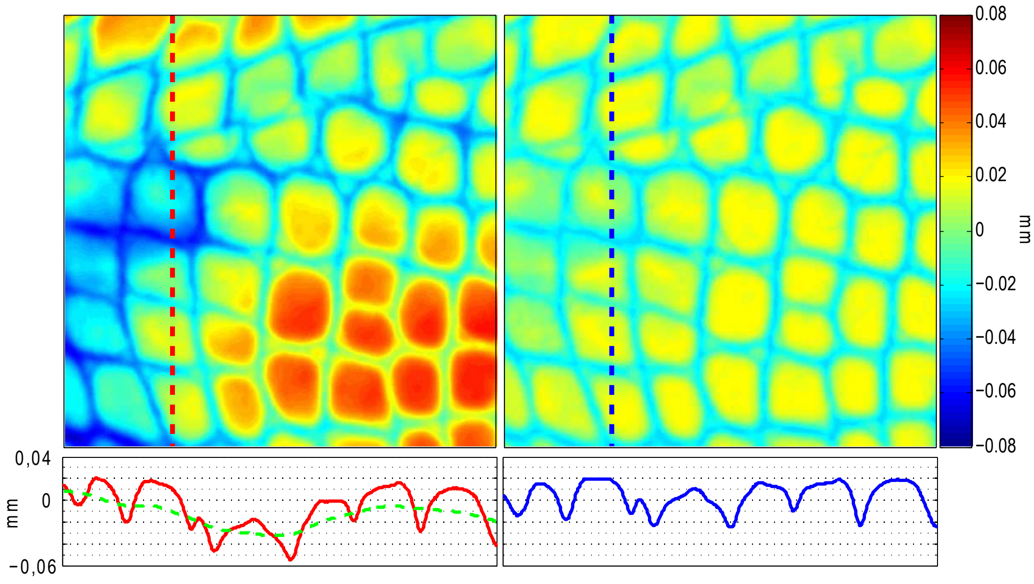


Figure 10.5: *Example for the heightfield correction. The top-left plot shows the heightfield of a leather material exhibiting large-scale height variations. The plots at the bottom show the height profiles for one slice (dashed lines in top figures). The dashed green line indicates the underlying low-frequency variation. The figures on the right illustrate the corrected heightfield. The low-frequent drift is removed while the height variations of local features are faithfully preserved.*

the major parallaxes and the small remaining disparities pose no serious obstacle for the synthesis.

For the interpolation of materials it is necessary to use a geometry representation that allows for interpolation in conjunction with the BTF, creating a consistent new synthesis result. We therefore use heightfields, since they can be easily included into the multi-channel images used during the synthesis.

While there are several techniques to extract heightfields directly from BTFs (e.g. [NZG05, MG09] or our approach from Chapter 5), for the examples used in this chapter we directly acquired them during the measurement via our structured light super-resolution approach [WSRK11], as this techniques is very robust and works for a wide range of materials. For this, we employed the Multi-View Dome described in Section 2.1.3. This approach allowed us to obtain sufficiently accurate geometry of the materials surface, allowing a high-quality parallax compensation.

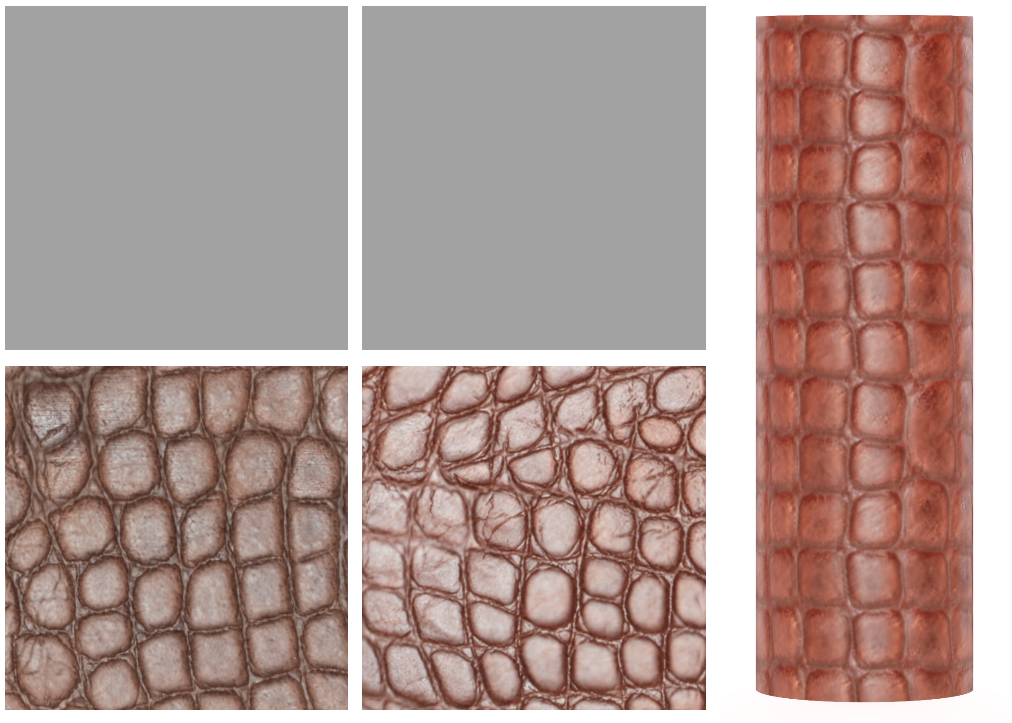
To evaluate the influence of the quality of the heightfield onto the interpolation results, we provide a comparison of results from different heightfield reconstruction techniques in Figure 10.5. In each case, we show the interpolation result and for both input materials the reconstructed heightfields and a single image from the resampled BTF (view and light direction exactly perpendicular to the sample). In the first example, we have performed no parallax correction at all. Then, we have used a low-quality heightfield reconstructed with the multiview-stereo reconstruction webservice ARC3D (`www.arc3d.be`). Next, we show the result for our heightfields obtained via structured light. Finally, we have used a very detailed heightmap reconstructed with our technique from Chapter 5.

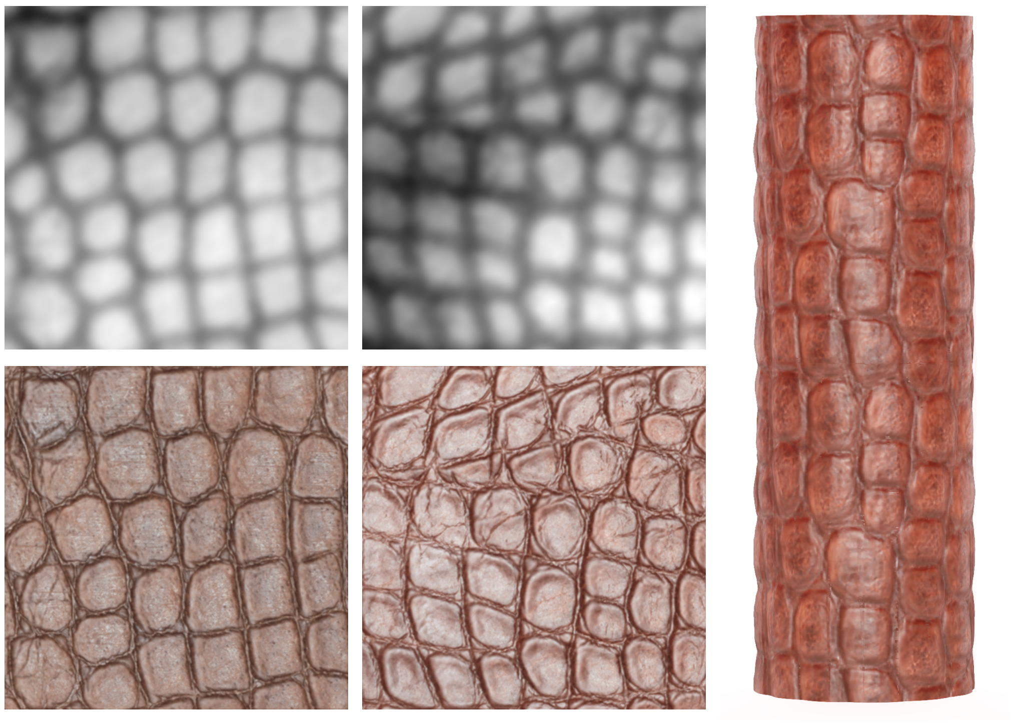
The BTF images show that with increasing heightfield quality less and less of the meso-structure has to be encoded in the BTF. Furthermore, without parallax correction, the interpolation resulted in a blurred image with few details and a very regular feature structure. In contrast, when a heightfield was utilized, more details were preserved and a more complex feature structure was generated. Even the low-quality heightfield from ARC3D improved the quality considerably.

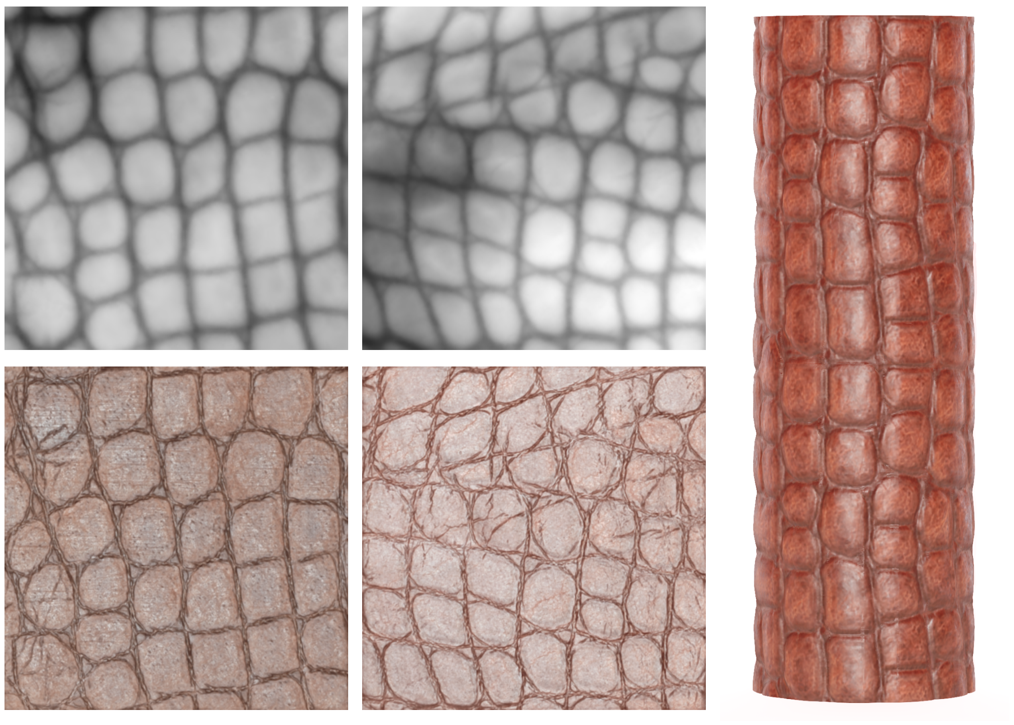
Many real material samples contain low-frequency large-scale height variation, such as bulges or waves, that are not considered part of the material features. Such large-scale changes in the input data violate the fundamental assumption of stationarity made in texture synthesis. Having these effects in the heightfield would impede the synthesis process: Synthesizing torroidal textures is impaired and also for the non-torroidal case, corresponding features at different surface positions could lie at different height levels and would thus not be matched. For these reasons, we remove low-frequency components from the heightfields prior to synthesis. We achieve this by solving a constrained least squares optimization, penalizing deviations of the surface normals while enforcing a maximal total height difference. The bound is chosen to still allow the natural height variation found within the material features by taking the 90% percentile of the maximum height-differences in all local neighborhoods. The neighborhood size is material dependent and was manually selected in our examples. For an example, please refer to Figure 10.5.

Some results of the parallax correction for the synthesis are shown in Figure 10.6. There, we compare an interpolation performed on BTFs that were rectified using a reference-plane and BTFs that were re-projected onto a heightfield. The interpolation result from rectified BTFs (Figure 10.6b) shows a blurred appearance for the parts of the surface that protrude from the reference plane. In contrast, when the parallaxes are corrected prior to the interpolation, a considerably sharper and more detailed result (Figure 10.6a) is obtained.

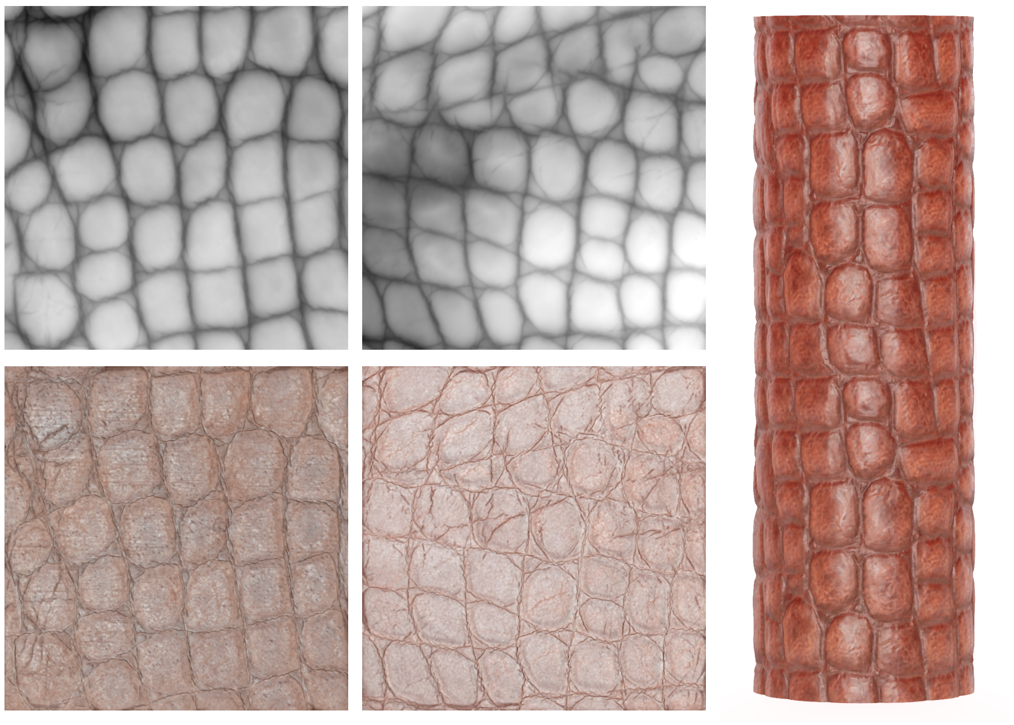
The final interpolation result consisting of the new heightfield and Eigen-textures can directly be used for rendering. In addition to the synthesized BTF, which is in compressed form and can be evaluated by following the approach by Koudelka


(a) No parallax correction (planar heightmap)


(b) Heightmap reconstructed with ARC3D (www.arc3d.be)

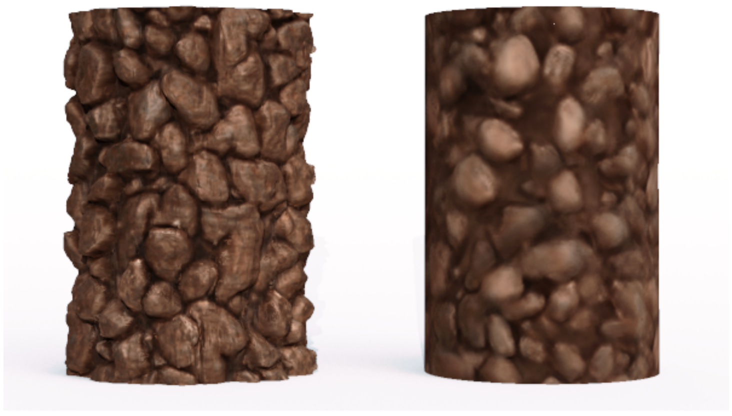


(c) Heightmap from structured light measurement [WSRK11]



(d) Heightmap reconstructed with technique from Chapter 5

Figure 10.5: *Results with different heightmap reconstruction techniques*



(a) With parallax correction **(b)** Without parallax correction

Figure 10.6: *Comparison of interpolation results with and without parallax correction. Without the correction, the result is blurred and fine details are lost during the synthesis.*

and Magda [KM03], the heightfield needs to be taken into account as well. For fast rendering we therefore use OpenGL tessellation and geometry shaders to perform displacement mapping directly on the GPU, whereas our high-quality still renderings presented in this thesis are created with mental-ray, which also supports displacement mapping. Using displacement mapping holds the additional advantages that the silhouettes of the materials (see Figure 10.7 for examples) are reproduced and that considerably fewer SVD components are needed for the compression due to the reduced parallax.

10.3 Results

We evaluate our approach on real-world data coming from a number of material samples from multiple material classes. To show the flexibility and usefulness of the system, we investigate several different application scenarios. All timings are given for a system with two Intel Xeon E5645 at 2.4 GHz (12 cores total), 144 GB of RAM and a GeForce GTX 570 GPU.

As input for all results presented in this chapter, we use BTF measurements captured with our Multi-View Dome (see Section 2.1.3). These have a texture resolution of 512×512 texels, covering 4.65 cm $\times$ 4.65 cm at 280 DPI, captured under a full sampling of 151×151 view and light directions. Additionally, for each of them, a registered heightfield at the same resolution was captured via structured-light. The parallax correction from Section 10.2.2 was then computed

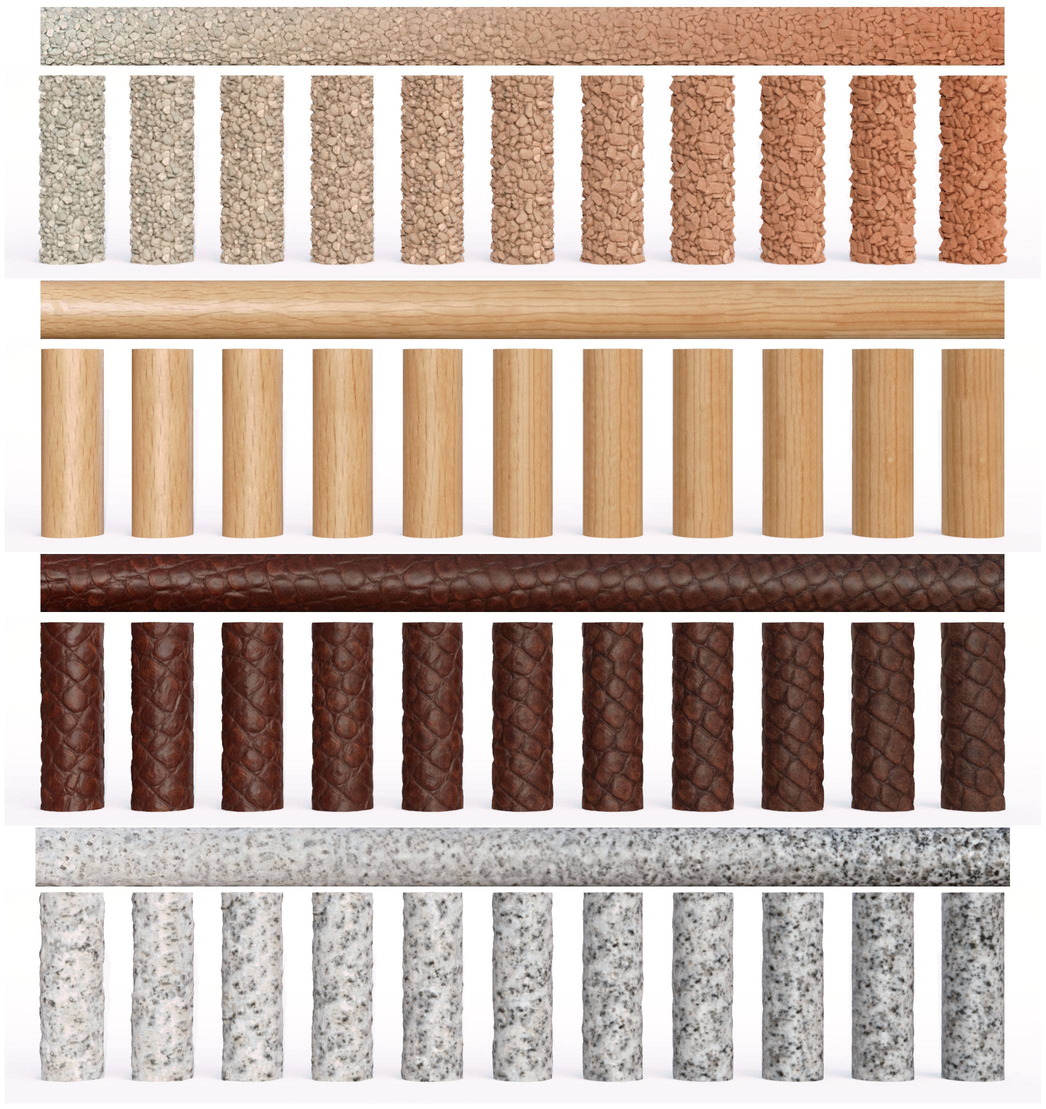


Figure 10.7: *Example interpolations between measured materials. The horizontal cylinder shows a continuous interpolation sequence between the materials. The vertical cylinders each feature a synthesis result at one discrete interpolation step. The images are generated using monte-carlo path tracing with an environment-illumination and an additional point-light source to emphasize highlights. Further results are shown in Figure 10.8.*

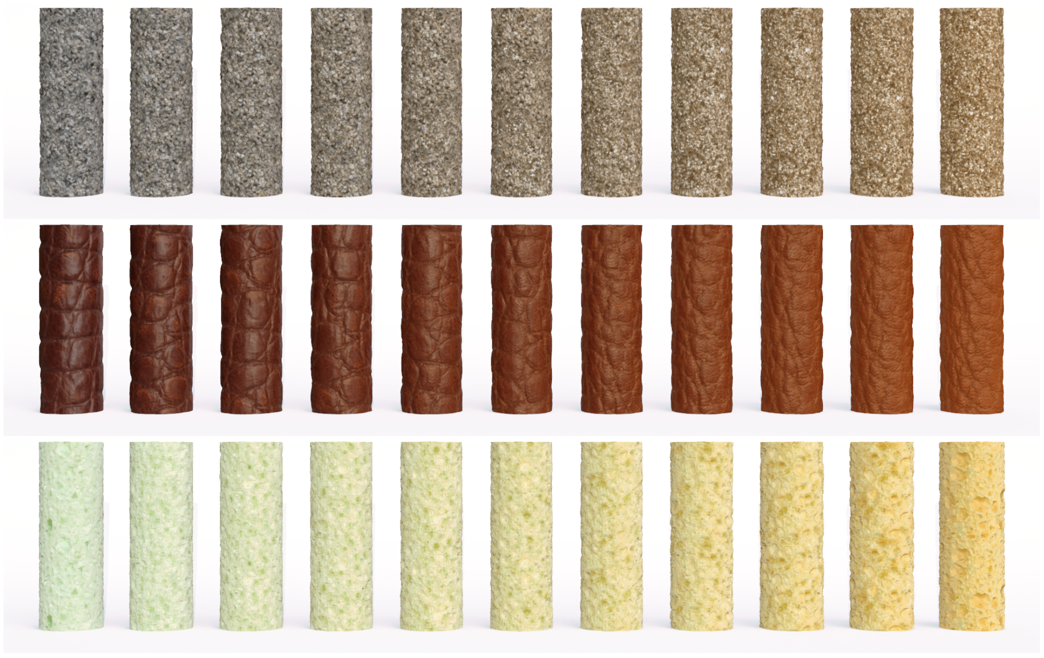


Figure 10.8: *Additional interpolation results. The two sponges show a failure case in which the assumptions of our algorithm were no longer fully satisfied. Although the algorithm manages to reproduce the appearance of the individual sponges (left and rightmost cylinders), the intermediate interpolants are not convincing.*

once as a preprocessing step. Preprocessing the data took about 5.5 hours per input sample. We also created hand-drawn binary feature images to mark the parts which should be aligned during synthesis. Due to the parallax correction, the features in all of the input images and the heightfield are in alignment, and thus the feature maps can be created using any of these input images as reference. This simplifies the creation of the feature maps, as it depends strongly on the utilized material, whether the features can be easier recognized in the heightfield or in the images, and which view and light direction is best suited in the latter case.

For each synthesis, all participating BTFs were converted into one common basis containing $c = 16$ Eigen-components. This rather low number in comparison to ordinary BTF compression is possible due to the parallax correction and represents a compromise between quality and synthesis time. The reported timings were achieved using a moderately optimized and mostly parallelized C++ implementation of the algorithm (with some parts in MATLAB). Due to the quite large amount of memory required for the candidate patches, up to tens of GB, we compute the PCA on the GPU with the technique presented in Chapter 3. Furthermore, we also accelerated the statistical synthesis utilizing CUDA.

Material Design

By mixing two or more input materials with user-specified blending amounts, it is possible to create a smooth sequence of all intermediate materials between the input exemplar. In Figures 10.1, 10.7 and 10.8, we demonstrate that for a wide range of different material classes, this enables the creation of perceptually plausible intermediate materials. This allows a designer to create new materials by exploring the space of many possible interpolations between a small set of measured BTFs. The creation of one interpolated result at a resolution of 512×512 pixels takes less than 1 hour. The synthesis results are seamlessly tileable and can directly be rendered on arbitrary scenes under any illumination.

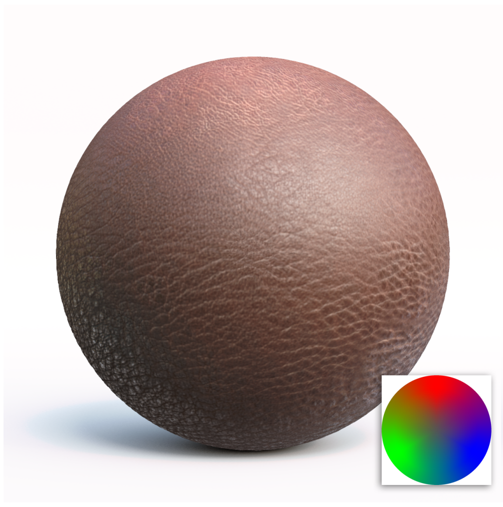
Spatially Varying Interpolation

The continuous sequences in Figure 10.7 and the more complex spatial interpolations in Figure 10.9, demonstrate the use of our technique for the creation of complex spatially varying interpolated materials. In each of the cases, the user specified the distribution of interpolation ratios of the different participating materials as an input. The algorithm then creates interpolated patches according to the given ratios, following the description in Section 9.5. This allows the creation of complex new BTFs exhibiting both smooth and hard material transitions, with material-features, such as leather bumps or tree rings, continuing in a meaningful way throughout the whole image. The creation of a $1024 \times 1024\,\text{pixel}$ interpolation sequence with three different input materials takes about 4 hours and 15 minutes.

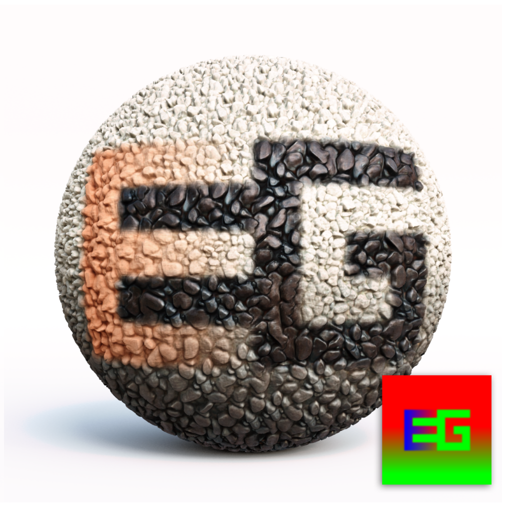
Separation of Feature Interpolation and Blending Factor

The described interpolation algorithm creates results in which the overall structure of the material features, resulting from the distribution of the individual patches, is always interpolated together with the blending of the reflectance values and meso-scale effects. It is desirable to separate these two aspects to give the designer more creative leeway. One could thus create a material with the features interpolated according to one mixture of samples but the reflectance characteristics coming from a different combination. In Figure 10.10, we illustrate this concept by performing the material feature interpolation along the horizontal axis and then the blending of reflectance and meso-scale effects along the vertical axis.

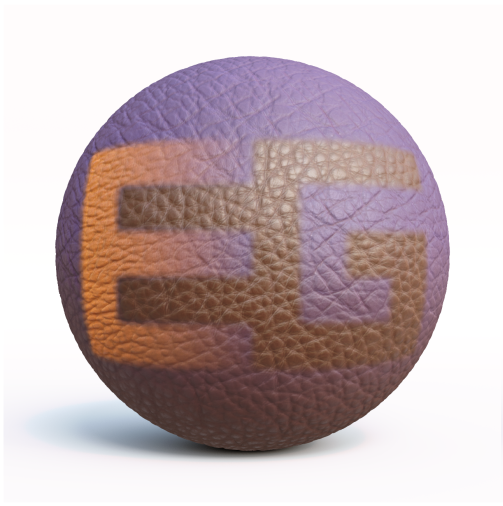
In order to do this, we want to generate intermediate results which follow the feature topology of one interpolation but still maintain their original reflectance



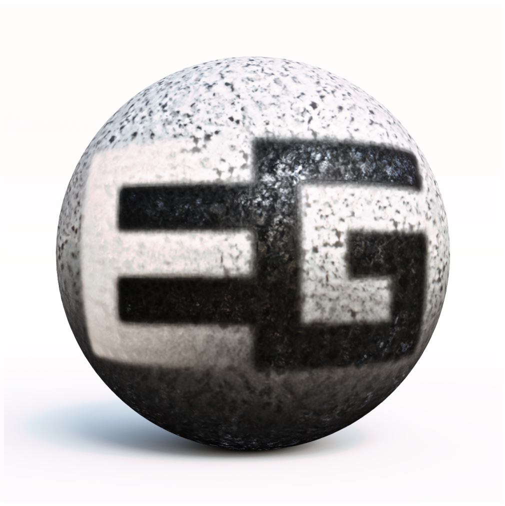
(a) 3 material interpolation



(b) complex spatial distribution (gravel)

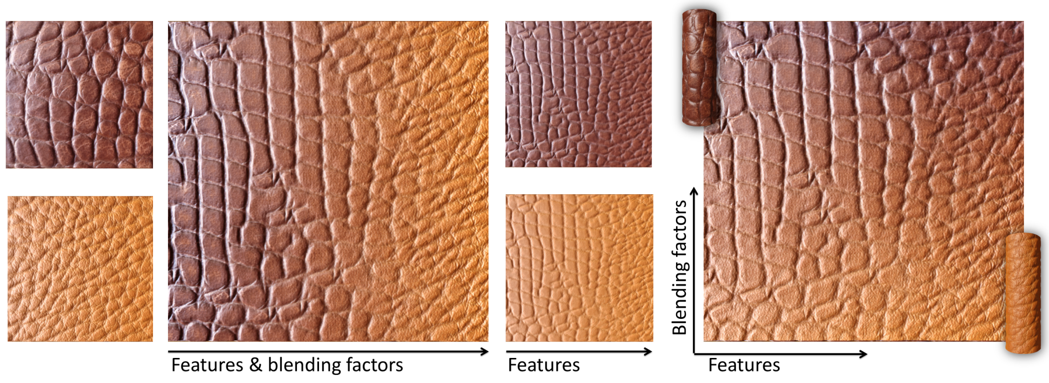


(c) complex spatial distribution (leather)



(d) complex spatial distribution (stones)

Figure 10.9: *Interpolations of multiple materials. Figure (a) shows path traced renderings of a smooth interpolation between three leathers. In Figures (b), (c) and (d), three materials are interpolated following a complex distribution pattern. For both cases, the distribution map is given in the inset.*



(a) input **(b)** basic interpolation **(c)** reassembled input **(d)** independent interpolation

Figure 10.10: *Independent material feature and reflectance characteristics interpolation. The basic algorithm is used to interpolate the input materials (a) along the horizontal-axis resulting in (b). The operations during patch warping and assembly are tracked and applied to the input materials individually to create intermediate materials (c) that follow the interpolated material features but each exhibit the reflectance values of only one of the leathers. The result (d) is obtained by blending the reflectance and meso-structure of the intermediate materials along the vertical-axis and applying a statistical synthesis.*

characteristics and meso-structure (see Figure 10.10c). The latter can then be independently interpolated in a consecutive step. Arising from the nature of the basic interpolation algorithm, the distribution of the patches in the synthesized result is strongly influenced by the blending ratio during the patch combination: In dependence on the blending weights, features from one material or the other are more dominant in the patch assembly. To obtain the desired interpolated feature topology, it is thus necessary to use blended patches throughout the synthesis.

However, by keeping track of the warping and assembling operations during synthesis, it is possible to apply the same operations to the input materials to reassemble them into the same spatial layout separately (see Figure 10.10c). These reassembled intermediate materials are then linearly blended according to different weights than used for the synthesis. The statistical synthesis is then also performed according to the reflectance blending weights since it mostly influences reflectance and meso-scale structures. As Figure 10.10d demonstrates, it is possible to create a wide range of interesting material variations from just two input samples. The synthesis of this 1024×1024 pixel BTF required about 3.5 hours.

10.4 Limitations and Future Work

Our technique only makes sense for opaque materials for which the large-scale features can be represented as a heightfield. If the material has a very complex self-overlapping structure or exhibits considerable sub-surface scattering, this representation will no longer give satisfactory results. In the case of sub-surface scattering, the reconstruction of the previously removed shadows from the heightfield results in an unnaturally hard appearance, especially under point-light illumination. Furthermore, when the meso-structure is very complex the heightfield might no longer be able to fully resolve the parallaxes, resulting in problems during the interpolation. Figure 10.8, shows such a case where we attempted to interpolate between two sponges which both exhibit a very complex meso-structure and considerable sub-surface scattering. Especially, the intermediate results are no longer capable of consistently reproducing the larger holes and structures. Currently, we also do not take interreflections into account. As this is also a geometry-dependent non-local effect, they should also be corrected prior to the patch assembly step. This might be possible with a technique similar to the one discussed in Chapter 5.

Furthermore, by blending the Eigen-textures linearly, a logarithmic interpolation between the ABRDFs is performed due to the dynamic range reduction. However, this approach would not give very intuitive results when blending two materials with very different lobe sizes. It might be possible to integrate an approach similar to the one we proposed in Chapter 8 into the patch merging steps of the texture synthesis. However, it would be necessary to reproject the result of this interpolation into the ABRDF basis as otherwise the synthesis could no longer be performed on the factorized representation. If this reprojection is not possible without acceptable losses in quality, suitable basis-extensions would have to be performed additionally.

A further aspect to be considered is the fact that the warping operation we apply to the Eigen-Textures is only an approximation. The ABRDFs of a BTF are relative to the local tangent frame and when the spatial arrangement is rotated or warped, it would be necessary to apply the same rotation also to the ABRDFs. Instead, we only apply the transformation to the distribution maps, discarding the transformation of the reflectance behavior. For strongly anisotropic materials, this would cause problems. Fortunately, for many materials this effect is negligible, since the amount of warping allowed during the patch blending is limited by the utilized regularization and we eliminate much of the anisotropy in a BTF by resampling it into local coordinate systems of the heightfield. In our experiments, we did not directly observe any serious problems, although the failure of the sponge dataset might partially be coming from this effect. One could consider precomputing rotated versions for each of the Eigen-ABRDFs, again represented in the factorized basis. This allows expressing the change under rotation as a matrix which can be multiplied to the spatial distribution coefficients during warping. However, when a truncated basis is used, this operation will eventually result in loss of information as the rotated Eigen-ABRDFs can only be approximated.

We restrict ourselves to the interpolation of BTFs, since it is possible to represent a large class of materials with this representation and, as we show, it is also well-suited for the proposed material interpolation. However, there is a wide range of different material representations which can be derived from measured data, such as spatially varying mixtures of analytical BRDFs, analytical BRDFs with spatially varying parameters, non-negative factorizations or mixture models (see Section 2.2 for an overview), each of them with applications of their own. It might be worth investigating whether similar interpolation techniques could also be applied there. However, in some cases a straightforward transfer might not be possible. For example, when using spatially varying linear combinations of analytical basis BRDFs as we did in Chapter 5, both input materials might have a different set of basis BRDFs and it is not clear a priori how to bring them into correspondence.

For the examples presented in this thesis, we used hand-drawn feature maps. As the identification of features usually requires a semantic understanding, the creation of these maps is a non-trivial problem and we doubt that there will be a fully automatic technique that will work for every material. However, since our approach builds on heightfields, which provide semantically more meaningful information than textures, BTFs or SVBRDFs alone, it might be possible to automatically derive such feature-maps for a wider range of materials.

Finally, with our current not fully optimized implementation the computation time is too high to allow the envisaged application of an interactive exploration of the material space. The runtime is mainly dominated by two aspects: The creation of interpolated patches and the texture synthesis step. Apart from low-level optimizations, the computational demand of the algorithm could further be reduced by algorithmic improvements. For the patch-creation, the patch-correspondences and warping transformations could be precomputed for a given set of input materials. For the texture synthesis, a texture optimization algorithm [KEBK05] has been chosen because of the high-quality it provides. Possibly a faster texture synthesis algorithm would also suffice.

10.5 Conclusion

We have presented a technique to interpolate between several measured BTFs. The interpolation is performed on an SVD factorized representation of the input BTFs. To cope with parallaxes and self-shadowing, we project the BTFs onto a heightfield, remove shadows and resample the reflectance behavior into the local coordinate systems. Using this approach, it is possible to interpolate between a wide range of different input materials and to create complex spatially varying materials.

Part III

Closure

CHAPTER 11

CONCLUSIONS AND FUTURE RESEARCH

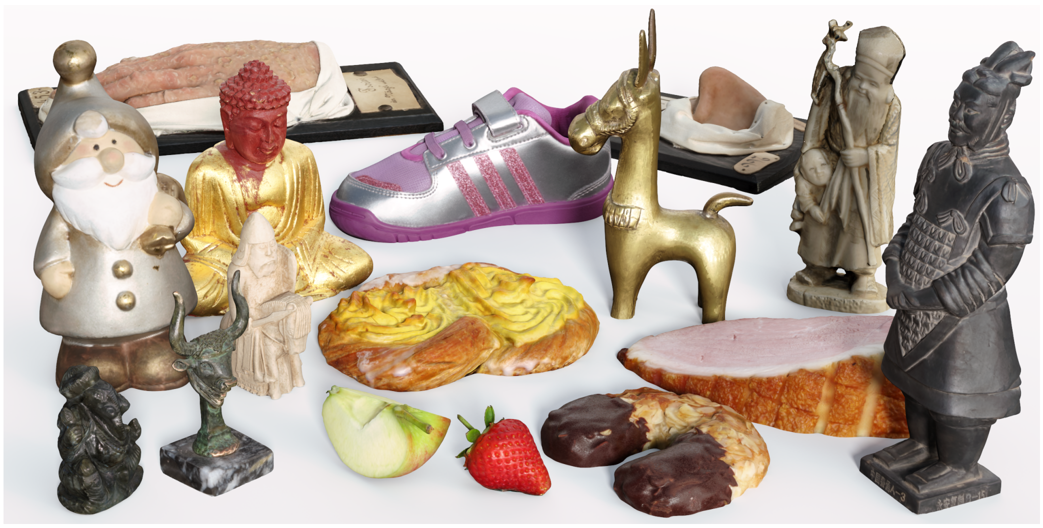


Figure 11.1: *Rendering of objects with measured BTFs*

During the last 15 years, the progress in data-driven techniques in computer graphics has enabled very faithful reproductions of a large range of different materials. For many materials and objects, it is possible to acquire their characteristic appearance from a real-world counterpart and to reproduce it faithfully in a high-quality rendering, which is hard to distinguish from an actual photograph (e.g. Figure 11.1). Still, these techniques have no yet found wide-spread use in practical applications. This is mainly due to two important reasons. On the one hand, the acquisition of the datasets remains challenging. Even though the measurement devices have improved considerably, as the devices built by our working group during the last 10 years described in Section 2.1 illustrate, the requirements in regard to quality and resolution have also risen. Thus, the acquisition is still too costly and time-consuming for many practical application, the size of the resulting datasets remains a challenge even for current computers and especially for specular materials the available angular resolutions are still not sufficient. On the other hand, the lack of suitable techniques to edit the resulting datasets has limited their application. Though the data-driven approaches are well suited for the exact reproduction of real-world exemplars, editing the resulting dataset is very challenging due to their size and complexity. Still in many applications an exact reproduction is not sufficient and techniques for the creative editing of the datasets are desired. This ranges from the generation of tileable materials without clearly visible repetitions, over the creation of spatially varying materials to the actual process of designing completely new materials based on the available measurements.

In this thesis, we have addressed these questions by developing techniques which allow for the more efficient compression of BTFs (Chapter 3), a more compact representation of reflectance datasets (Chapters 4, 5, 6, 7), the reconstruction of reflectance from sparser measurements (Chapters 5, 7) and interpolation techniques for BRDFs, textures and BTFs (Chapters 8,9,10). However, as always, a large number of open questions and possibilities for future research and improvement remain.

Parts of this conclusion have been published in the position paper "BTF based Material Representations: Current Challenges" by Roland Ruiters, and Reinhard Klein, at the *Eurographics Workshop on Material Appearance Modeling: Issues and Acquisition*, pages 17–20, June 2013.

11.1 Material Representations

Representations for materials can be roughly classified along the spectrum from model based approaches, which describe reality with an elaborate model and (ideally) a small number of parameters, to data-driven approaches, which make nearly no prior assumptions but instead use a comprehensive measurement. It is possible to derive both types of representations from real-world samples of the material. For model based approaches, the challenge is mainly in the development of suitable models and the fitting of the model, whereas for data-driven techniques the cost of the measurement and the compression of the resulting datasets are the main challenges. The representations used in this thesis span a wide range on this spectrum. They range from a nearly completely model based approach which represents a material as a heightfield and analytical BRDFs, over mostly data-driven approaches with additional model assumptions, such as the self-similarity, half/diff parameterization and regularization utilized for the tensor based SVBRDFs or the additional heightfield necessary for the BTF interpolation, to nearly completely data-driven compression techniques such as the compression based on a Sparse Tensor Decomposition.

Each of these techniques has its own distinctive set of advantages and disadvantages and makes different trade-offs. Therefore, it is not to be expected that a single approach can solve all problems. However, even the question where the "local optima" along this spectrum are has not yet been finally answered and provides a lot of space for future research. Due to the enormous complexity of many real-world materials, highly accurate simulations would require sophisticated models which are difficult to fit against measurements. On the other hand, there is a large demand for cheaper and simpler measurements, which stands in contrast to the high angular samplings that would be necessary to fully resolve high-frequent effects, such as specular highlights. Therefore, purely data-driven approaches are probably also not the ideal solution in many cases.

In Figure 11.2, a comparison between a photograph and a rendering of an object with a measured BTF is shown, which demonstrates the limitations of the purely data-driven acquisition. Most of the object is reproduced quite accurately, but the angular resolution is not sufficient to resolve the specular highlights correctly. As described in Chapter 7, this is due to limitations of both the utilized measurement setup, and the selected representation. It happens even though a large number (198) of illumination directions has been acquired and the resulting dataset requires already 165 GB of storage. Further increasing the angular resolution both of the dataset and measurement-device would therefore be very costly. Acquiring the reflectance of challenging objects, such as this example, in a way that at the same time is inexpensive and allows for a faithful reproduction, will thus require both progress with regard to the utilized measurement approaches as well as the applied representation.

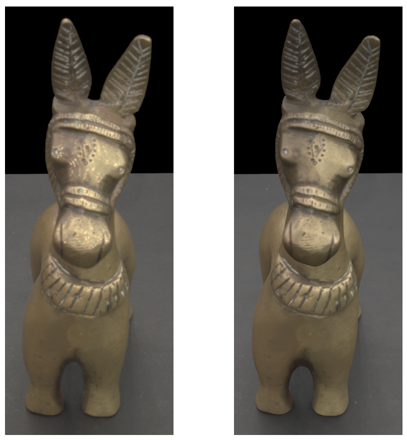


Figure 11.2: *Comparison of a photograph (left, background masked out) with a rendering (right) of an object with an acquired BTF.*

Cost-efficient measurements of complex objects or materials will probably necessitate the utilization of some kind of prior. However, an open question is whether typical assumptions such as analytical BRDF models, smoothness, low-rank etc. provide the best priors or whether data-driven priors are a superior approach. Given a suitably large set of high-quality measurements, these could provide the necessary prior to reconstruct new materials successfully from a much sparser measurement without explicitly making model assumptions. This idea has already been applied to reconstruct BRDFs [MPBM03b] or SVBRDFs [WWHL07] from a small number of samples and a database of isotropic BRDF measurements. This principle could be extended to a database containing spatially varying reflectance data in the form of BTFs as this would provide additional information in the form of local neighborhoods. In the area of image processing, this approach has already been successfully applied. In a recent example [SH12], that is very similar to our case, super-resolution is performed by first identifying images showing a similar context in a large database and then synthesizing a high-resolution image from them using the captured low-resolution image as a constraint. Similarly, it might be possible to identify measurements of similar materials in a database and then use constrained synthesis algorithms to create a plausible reconstruction of the presented sample.

In many cases, the selected representation serves at the same time as a prior during the reconstruction, for example when using analytical BRDF models, a heightfield or a low-rank representation. When using a data-driven prior, these two aspects should be decoupled. However, it is not be possible to separate the reconstruction process completely from the utilized representation. Due to the necessary resolution to resolve specular highlights, using a tabulated representation of the full dataset is infeasible. Therefore, the common approach of first performing a measurement or reconstruction and then a subsequent compression step does not scale well. Instead, a representation which is at the same time sufficiently compact, capable of reproducing the appearance of a wide range of materials faithfully, allows for the integration of a suitable prior and is directly applicable during the reconstruction process is required. Our work on the reconstruction of a material's reflectance behavior directly in a tensor based SVBRDF representation is a first step into this direction. However, currently the range of materials that can be represented this way is limited to isotropic SVBRDFs. It is still lacking the ability of a BTF to reproduce the characteristic appearance of more complex materials with a complex meso-structure.

Though the actual measurement process was not the focus of this thesis, as existing measurement devices were utilized to provide the input data, it remains an important question whether alternative measurement approaches should be utilized instead. In most of the current approaches, the continuous reflectance function is sampled at discrete points (or more precisely in small regions, if considering that a pixel is actually an integral in the spatial domain and the entrance pupil of the camera and the extent of the light source result in an integral in the angular domain). However, it is very difficult to scale this approach to the angular resolutions necessary to resolve highly specular objects, as these exhibit a very high-frequent reflectance behavior and one is thus fundamentally limited by the Nyquist criterion. The approach we utilized in Chapter 7 is limited to objects which exhibit a sufficient degree of self-similarity to provide an adequate sampling of all materials the object is composed of. One alternative might be to measure weighted integrals of the reflectance function. Area light sources or pattern-based or gradient illumination can be used to perform those measurements. Again, given a suitable prior, it should be possible to reconstruct the reflectance behavior from these measurements. First techniques reconstructing analytical SVBRDFs utilizing spherical gradient illumination [GCP*09], Gray-Codes [FCMB09] or Spherical Harmonics [TFG*13] have already been proposed. Similar approaches have received much attention lately in the context of compressive sensing, where the measurement is specified in the form of a random measurement basis and the prior as a second basis in which the signal can be represented sparsely. Currently, it is not clear whether the mathematical framework of compressive sensing can be directly applied to the measurement of BTFs. However, even if not, the general idea of performing measurements in a different basis than the currently employed Dirac delta functions and then utilizing a suitable prior to reconstruct the reflectance behavior might be an interesting avenue of future research.

11.2 Material Interpolation

The work in this thesis shows that it is possible to compute plausibly looking interpolations between measured materials. This is possible even for materials with complex feature topology, spatially varying reflectance behavior and a meso-structure resulting in strong parallaxes. This provides the necessary foundation for and demonstrates the general possibility of a data-driven material design approach. However, for a practical application of this design paradigm a lot of further questions have to be resolved. On the one hand, a lot of technical challenges remain, such as the necessity for a considerable speed-up to allow for interactive design, the question whether the feature maps can be generated automatically, the synthesis of large and repetition-free materials and the creation of seamless and distortion-free materials on the surface of an object. On the other hand, the question how the currently available material interpolation technology is best utilized to enable an intuitive and efficient material design process is still open.

Especially an efficient synthesis is of considerable practical importance. Our contact to practitioners has shown that this is currently one of the limiting factors. A texture artist, we were in contact with, would have been very interested in the texture interpolation technique, but only if he could use it interactively without having to wait hours for a result. Similarly, designers from the automobile industry would have liked to use synthesis techniques to create surface textures for car interiors but the required resolutions and sample sizes could not yet be realized with our approach. It remains an open question whether existing more efficient texture synthesis approaches could be combined with our patch interpolation technique or whether the fact that the resulting patches are less consistent than uninterpolated ones results in unsatisfactory results. Furthermore, either more efficient techniques to compute the interpolated patches or a suitable precomputation approach should be developed.

The question how the material interpolation technology can be best utilized for a practical material design also strongly depends on the availability of interactive synthesis. It is still unclear, which type of user interface and which design paradigm allows for the most intuitive and efficient design process. It might be that an approach such as the one we presented in Chapter 8 for BRDFs, where the designer is given an intuitive interface to directly interpolate between the measured materials, is sufficient. On the other hand, it might be that a more indirect approach, such as the one proposed by Matusik [MPBM03a] for BRDFs, in which the actually measured samples are abstracted and the user instead navigates a material space along perceptual traits, is more intuitive.

A more fundamental but nonetheless very important question for material design is the possibility of a perceptual metric for materials. Is it possible to quantify the perceptual distance between two materials? This would allow for a perceptually uniform movement in the material space and thus also for the creation of perceptually uniform interpolation sequences. However, this question is far more general as such a metric would enable practical material retrieval and material recognition. This in turn could provide a connection between the virtual material design and reality. It might become possible to find a material in the databases from suppliers, to provide quality control or even to guide production processes based on a material description resulting from the material design process.

[AB00] ANDERSSON C. A., BRO R.: The N-way Toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems 52*, 1 (August 2000), 1 – 4. 85

[ABD*90] ANDERSON E., BAI Z., DONGARRA J., GREENBAUM A., MCKENNEY A., CROZ J. D., HAMMERLING S., DEMMEL J., BISCHOF C., SORENSEN D.: LAPACK: a portable linear algebra library for high-performance computers. In *Proceedings of the Conference on Supercomputing* (October 1990), pp. 2–11. http://www.netlib.org/lapack/. 34

[AEB06] AHARON M., ELAD M., BRUCKSTEIN A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing 54*, 11 (November 2006), 4311–4322. 20, 42, 44, 47

[AP08] AN X., PELLACINI F.: AppProp: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 27* (August 2008), 40:1–40:9. 101, 102

[APS00] ASHIKMIN M., PREMOŽE S., SHIRLEY P.: A microfacet-based BRDF generator. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 2000), pp. 65–74. 18

[AZK08] ALLDRIN N. G., ZICKLER T., KRIEGMAN D.: Photometric stereo with non-parametric and spatially-varying reflectance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008). 21

[BAOR06] BEN-ARTZI A., OVERBECK R., RAMAMOORTHI R.: Real-time BRDF editing in complex lighting. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 25*, 3 (July 2006), 945–954. 117

[BGM09] BEYLKIN G., GARCKE J., MOHLENKAMP M. J.: Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing 31*, 3 (March 2009), 1840–1857. 82, 92, 95

[BJEYLW01] BAR-JOSEPH Z., EL-YANIV R., LISCHINSKI D., WERMAN M.: Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics (VCG) 7*, 2 (April 2001), 120–135. 132

[BK07] BADER B. W., KOLDA T. G.: Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing 30*, 1 (December 2007), 205–231. 50

[BM92] BESL P., MCKAY N.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14*, 2 (February 1992), 239–256. 138

[BÖK11] BILGILI A., ÖZTÜRK A., KURT M.: A general BRDF representation based on tensor decomposition. *Computer Graphics Forum (CGF) 30*, 8 (December 2011), 2427–2439. 20, 79, 83, 88

[Boo02] BOOTHE R.: *Perception of the visual environment*. Springer Verlag, November 2002. 84

[BP03] BARSKY S., PETROU M.: The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 25*, 10 (October 2003), 1239–1252. 72

[Bra02] BRAND M.: Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the European Conference On Computer Vision (ECCV) - Part I* (May 2002), pp. 707–720. 27, 30, 31

[CDS98] CHEN S. S., DONOHO D. L., SAUNDERS M. A.: Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing 20*, 1 (August 1998), 33–61. 42

[CGS06] CHEN T., GOESELE M., SEIDEL H.-P.: Mesostructure from specularity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 2, pp. 1825–1832. 59

[CLFS07] CHEN T. B., LENSCH H. P. A., FUCHS C., SEIDEL H. P.: Polarization and phase-shifting for 3D scanning of translucent objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2007). 61

[CLL07] CHANG J. Y., LEE K. M., LEE S. U.: Multiview normal field integration using level set methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2007). 60

[CM02] COMANICIU D., MEER P.: Mean Shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 24*, 5 (May 2002), 603–619. 141

[CPK06] COLBERT M., PATTANAIK S., KŘIVÁNEK J.: BRDF-Shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Applications 26*, 1 (January 2006), 30–36. 117

[CR00] CHUI H., RANGARAJAN A.: A new algorithm for non-rigid point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2000), vol. 2, pp. 2044–2051. 137

[CSL08] CHEN T. B., SEIDEL H. P., LENSCH H. P. A.: Modulated phase-shifting for 3D scanning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008). 61

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG) 1*, 1 (January 1982), 7–24. 63, 87

[DL97] DE LATHAUWER L.: *Signal processing based on multilinear algebra*. PhD thesis, K.U. Leuven, E.E. Department (ESAT), September 1997. 21

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 31*, 4 (July 2012), 82:1–82:10. 149, 151

[Duc77] DUCHON J.: Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Lecture Notes in Mathematics*, vol. 571. Springer Berlin / Heidelberg, April 1977, pp. 85–100. 137

[DvGNK97] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 1997), pp. 151–157. 10

[DWT*10] DONG Y., WANG J., TONG X., SNYDER J., LAN Y., BEN-EZRA M., GUO B.: Manifold bootstrapping for SVBRDF capture. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 29*, 4 (July 2010), 98:1–98:10. 19

[EAH00] ENGAN K., AASE S. O., HUSØY J. H.: Multi-frame compression: theory and design. *Signal Processing 80*, 10 (October 2000), 2121–2140. 42

[EF01] Efros A. A., Freeman W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 2001), pp. 341–346. 133

[EL99] Efros A. A., Leung T. K.: Texture synthesis by non-parametric sampling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (September 1999), vol. 2, pp. 1033–1038. 133

[EVC08] Esteban C. H., Vogiatzis G., Cipolla R.: Multiview photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 30*, 3 (March 2008), 548–554. 60

[FCMB09] Francken Y., Cuypers T., Mertens T., Bekaert P.: Gloss and normal map acquisition of mesostructures using Gray Codes. In *Proceedings of the International Symposium on Advances in Visual Computing (ISVC): Part II* (December 2009), pp. 788–798. 18, 181

[FH07] Fang H., Hart J. C.: Detail preserving shape deformation in image editing. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 26*, 3 (July 2007), 12. 132

[FH09] Filip J., Haindl M.: Bidirectional texture function modeling: A state of the art survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 31*, 11 (November 2009), 1921–1940. 12

[FKIS02] Furukawa R., Kawasaki H., Ikeuchi K., Sakauchi M.: Appearance based object modeling using texture database: acquisition, compression and rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques (EGWR)* (June 2002), pp. 257–266. 20, 41, 92

[Fou95] Fournier A.: Separating reflection functions for linear radiosity. In *Proceedings of the Eurographics Workshop on Rendering Techniques (EGWR)* (June 1995), pp. 296–305. 19, 79

[Gar10] Garcke J.: Classification with sums of separable functions. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD): Part I* (September 2010), pp. 458–473. 95

[GCHS05] Goldman D. B., Curless B., Hertzmann A., Seitz S. M.: Shape and spatially-varying BRDFs from photometric stereo. In *Proceedings of the IEEE International Conference on Computer*

Vision (ICCV) (October 2005), vol. 1, pp. 341–348. 17, 57, 58, 59, 61, 64, 65, 72

[GCP*09] GHOSH A., CHEN T., PEERS P., WILSON C. A., DEBEVEC P.: Estimating specular roughness and anisotropy from second order spherical gradient illumination. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2009), pp. 1161–1170. 18, 181

[Geo03] GEORGHIADES A. S.: Recovering 3-D shape and reflectance from a small number of photographs. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2003), pp. 230–240. 17, 18, 59

[GL96] GOLUB G. H., LOAN C. F. V.: *Matrix computations*, 3rd ed. Johns Hopkins University Press, October 1996. 27

[GMSK09] GUTHE M., MÜLLER G., SCHNEIDER M., KLEIN R.: BTF-CIELab: A perceptual difference measure for quality assessment and compression of BTFs. *Computer Graphics Forum (CGF) 28*, 1 (March 2009), 101–113. 19

[GR97] GORODNITSKY I., RAO B.: Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing 45*, 3 (March 1997), 600–616. 42

[GTHD03] GARDNER A., TCHOU C., HAWKINS T., DEBEVEC P.: Linear light source reflectometry. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 22*, 3 (July 2003), 749–758. 17, 18

[Hag89] HAGER W. W.: Updating the inverse of a matrix. *SIAM Review 31*, 2 (June 1989), 221–239. 102

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 1995), pp. 229–238. 132, 133

[HB08] HABER T., BEKAERT P.: Image-based acquisition of shape and spatially varying reflectance. In *Proceedings of the British Machine Vision Conference (BMVC)* (September 2008), pp. 81.1–81.10. 59

[HDKS00] HEIDRICH W., DAUBERT K., KAUTZ J., SEIDEL H.-P.: Illuminating micro geometry based on precomputed visibility. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 2000), pp. 455–464. 70

[HF11] HAINDL M., FILIP J.: Advanced textural representation of materials appearance. In *SIGGRAPH Asia, Courses* (December 2011), pp. 1:1–1:84. 16, 156

[HH05] HAINDL M., HATKA M.: BTF Roller. In *Proceedings of the International Workshop on Texture Analysis and Synthesis (TEXTURE)* (October 2005), pp. 89–94. 156

[HLZ10] HOLROYD M., LAWRENCE J., ZICKLER T.: A coaxial optical scanner for synchronous acquisition of 3D geometry and surface reflectance. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 29*, 4 (July 2010), 99:1–99:12. 17, 18

[HMM00] HALL P., MARSHALL D., MARTIN R.: Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 22*, 9 (September 2000), 1042–1049. 30

[HS03] HERTZMANN A., SEITZ S. M.: Shape and materials by example: A photometric stereo approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2003), vol. 1, pp. 533–540. 59

[HS05] HERTZMANN A., SEITZ S. M.: Example-based photometric stereo: Shape reconstruction with general, varying BRDFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27*, 8 (August 2005), 1254–1264. 17, 18

[HTSG91] HE X. D., TORRANCE K. E., SILLION F. X., GREENBERG D. P.: A comprehensive physical model for light reflection. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 1991), pp. 175–186. 17

[HZW*06] HAN J., ZHOU K., WEI L.-Y., GONG M., BAO H., ZHANG X., GUO B.: Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer: International Journal of Computer Graphics 22*, 9 (September 2006), 918–925. 133

[JCYS04] JIN H., CREMERS D., YEZZI A. J., SOATTO S.: Shedding light on stereoscopic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2004), vol. 1, pp. 36–42. 60

[KB09] KOLDA T. G., BADER B. W.: Tensor decompositions and applications. *SIAM Review 51*, 3 (September 2009), 455–500. 20

[KBD07] KAUTZ J., BOULOS S., DURAND F.: Interactive editing and modeling of bidirectional texture functions. *ACM Transactions on*

Graphics (TOG) - Proceedings of the ACM SIGGRAPH 26, 3 (July 2007), 53. 155

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)* (2006), pp. 61–70. 73

[KDMR*03] KREUTZ-DELGADO K., MURRAY J. F., RAO B. D., ENGAN K., LEE T.-W., SEJNOWSKI T. J.: Dictionary learning algorithms for sparse representation. *Neural Computation 15*, 2 (February 2003), 349–396. 42

[KDS96] KOENDERINK J. J., DOORN A. J. V., STAVRIDI M.: Bidirectional reflection distribution function expressed in terms of surface scattering modes. In *Proceedings of the European Conference On Computer Vision (ECCV)* (April 1996), vol. 2, pp. 28–39. 18

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 24*, 3 (July 2005), 795–802. 131, 133, 135, 139, 141, 173

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2D exemplars. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 26*, 3 (July 2007), 2:1–2:9. 133, 141

[KM99] KAUTZ J., MCCOOL M. D.: Interactive rendering with arbitrary BRDFs using separable approximations. In *Proceedings of the Eurographics Workshop on Rendering Techniques (EGWR)* (June 1999), pp. 281–292. 19, 79

[KM03] KOUDELKA M. L., MAGDA S.: Acquisition, compression, and synthesis of bidirectional texture functions. In *Proceedings of the International Workshop on Texture Analysis and Synthesis (TEXTURE)* (October 2003), pp. 59–64. 19, 41, 50, 156, 157, 166

[KPRN11] KABUL I., PIZER S. M., ROSENMAN J., NIETHAMMER M.: An optimal control approach for texture metamorphosis. *Computer Graphics Forum (CGF) 30*, 8 (December 2011), 2341–2353. 149, 151

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 22*, 3 (July 2003), 277–286. 133

[KSOF05] Kawasaki H., Seo K.-D., Ohsawa Y., Furukawa R.: Patch-based BTF synthesis for real-time rendering. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (September 2005), vol. 1, pp. I–393–I–396. 156

[LBAD*06] Lawrence J., Ben-Artzi A., DeCoro C., Matusik W., Pfister H., Ramamoorthi R., Rusinkiewicz S.: Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 25*, 3 (July 2006), 735–745. 20, 117, 155

[LF97] Lalonde P., Fournier A.: A wavelet representation of reflectance functions. *IEEE Transactions on Visualization and Computer Graphics (VCG) 3*, 4 (October 1997), 329–336. 18

[LFTG97] Lafortune E. P. F., Foo S.-C., Torrance K. E., Greenberg D. P.: Non-linear approximation of reflectance functions. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 1997), pp. 117–126. 17

[LH06] Lefebvre S., Hoppe H.: Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 25*, 3 (July 2006), 541–548. 136

[LHYK05] Lim J., Ho J., Yang M.-H., Kriegman D.: Passive photometric stereo from motion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (October 2005), vol. 2, pp. 1635–1642. 60

[LHZ*04] Liu X., Hu Y., Zhang J., Tong X., Guo B., Shum H.-Y.: Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics (VCG) 10*, 3 (May 2004), 278–289. 19, 41, 50, 156

[LKG*03] Lensch H. P. A., Kautz J., Goesele M., Heidrich W., Seidel H.-P.: Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics (TOG) 22*, 2 (April 2003), 234–257. 17, 18, 107

[LKK98] Lu R., Koenderink J. J., Kappers A. M. L.: Optical properties (bidirectional reflection distribution functions) of velvet. *Applied Optics 37*, 25 (September 1998), 5974–5984. 18

[LL11] Lepage D., Lawrence J.: Material matting. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH Asia 30*, 6 (December 2011), 144:1–144:10. 155

[LLSY02] LIU Z., LIU C., SHUM H.-Y., YU Y.: Pattern-based texture metamorphosis. In *Proceedings of the Pacific Conference on Computer Graphics and Applications (PG)* (October 2002), pp. 184–193. 132

[LLX*01] LIANG L., LIU C., XU Y.-Q., GUO B., SHUM H.-Y.: Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG) 20*, 3 (July 2001), 127–150. 133

[LM01] LEUNG T., MALIK J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision (IJCV) 43*, 1 (June 2001), 29–44. 156

[LMWY09] LIU J., MUSIALSKI P., WONKA P., YE J.: Tensor completion for estimating missing values in visual data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (September 2009), pp. 2114 –2121. 132

[LPF*07] LEUNG M.-K., PANG W.-M., FU C.-W., WONG T.-T., HENG P.-A.: Tileable BTF. *IEEE Transactions on Visualization and Computer Graphics (VCG) 13*, 5 (September 2007), 953–965. 156

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 23*, 3 (August 2004), 496–505. 19, 79, 80, 83, 87, 89

[LS00] LEWICKI M. S., SEJNOWSKI T. J.: Learning overcomplete representations. *Neural Computation 12*, 2 (February 2000), 337–365. 42

[LW07] LAI C.-H., WU J.-L.: Temporal texture synthesis by patch-based sampling and morphing interpolation. *Computer Animation and Virtual Worlds 18*, 4-5 (September 2007), 415–428. 132

[LWWT07] LIU L., WANG Y., WANG Q., TAN T.: Fast principal component analysis using eigenspace merging. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (September 2007), vol. 6, pp. VI–457–VI–460. 27

[LYS01] LIU X., YU Y., SHUM H.-Y.: Synthesizing bidirectional texture functions for real-world surfaces. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 2001), pp. 97–106. 156, 160

[MAA01] MCCOOL M. D., ANG J., AHMAD A.: Homomorphic factorization of BRDFs for high-performance rendering. In *Proceedings of* *the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 2001), pp. 171–178. 19, 79, 80, 87

[Mat03] MATUSIK W.: *A Data-Driven Reflectance Model*. PhD thesis, Massachusetts Institute of Technology, September 2003. 13

[MBK05] MÜLLER G., BENDELS G. H., KLEIN R.: Rapid synchronous acquisition of geometry and BTF for cultural heritage artefacts. In *Proceedings of the International Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST)* (November 2005), pp. 13–20. 14, 20, 92

[McA02] MCALLISTER D. K.: *A generalized surface appearance representation for computer graphics*. PhD thesis, The University of North Carolina at Chapel Hill, 2002. 17

[MG09] MENZEL N., GUTHE M.: g-BRDFs: An intuitive and editable btf representation. *Computer Graphics Forum (CGF) 28*, 8 (December 2009), 2189–2200. 18, 155, 161, 162

[MHP*07] MA W.-C., HAWKINS T., PEERS P., CHABERT C.-F., WEISS M., DEBEVEC P.: Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2007), pp. 183–194. 18

[MK06] MAGDA S., KRIEGMAN D. J.: Reconstruction of volumetric surface textures for real-time rendering. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2006), pp. 19–29. 17, 18

[MMK03] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time rendering of measured BTFs using local pca. In *Proceedings of the Conference on Vision, Modeling and Visualisation (VMV)* (November 2003), pp. 271–280. 19, 34, 35, 41, 50

[MMS*04] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis and rendering of bidirectional texture functions. *Computer Graphics Forum (CGF) - Eurographics, State of the Art Reports 24*, 1 (March 2004), 83–109. 12, 14, 16

[MOHW07] MEZA J. C., OLIVA R. A., HOUGH P. D., WILLIAMS P. J.: OPT++: An object-oriented toolkit for nonlinear optimization. *ACM Transactions on Mathematical Software 33*, 2 (June 2007), 12:1–12:27. 65

[MP99] MUNICH M. E., PERONA P.: Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (September 1999), vol. 1, pp. 108–115. 122

[MPBM03a] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 22*, 3 (July 2003), 759–769. 13, 18, 82, 83, 96, 118, 154, 159, 182

[MPBM03b] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: Efficient isotropic BRDF measurement. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2003), pp. 241–248. 13, 180

[MSK07] MÜLLER G., SARLETTE R., KLEIN R.: Procedural editing of bidirectional texture functions. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2007), pp. 219–230. 154, 156

[MSUA12] MONTES SOLDADO R., UREÑA ALMAGRO C.: *An Overview of BRDF Models*. Tech. Rep. LSI Technical Reports;2012-001, University of Granada, March 2012. 17

[Mül08] MÜLLER G.: *Data-Driven Methods for Compression and Editing of Spatially Varying Appearance*. PhD thesis, Universität Bonn, December 2008. 19, 20, 50

[MWLT00] MARSCHNER S. R., WESTIN S. H., LAFORTUNE E. P. F., TORRANCE K. E.: Image-based bidirectional reflectance distribution function measurement. *Applied Optics 39*, 16 (June 2000), 2592–2600. 13

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 24*, 3 (July 2005), 787–794. 129, 132, 133, 136, 154

[Nas84] NASH S. G.: Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis 21*, 4 (August 1984), 770–788. 66

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental analysis of BRDF models. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2005), pp. 117–126. 17, 18, 63, 65, 87, 104, 105, 107

[NDM06] NGAN A., DURAND F., MATUSIK W.: Image-driven navigation of analytical BRDF models. In *Proceedings of the Eurographics*

Symposium on Rendering (EGSR) (June 2006), pp. 399–407. 85, 117, 127

[Nic65] NICODEMUS F. E.: Directional reflectance and emissivity of an opaque surface. *Applied Optics 4*, 7 (July 1965), 767–773. 10

[NIK90] NAYAR S. K., IKEUCHI K., KANADE T.: *Shape from Interreflections*. Tech. Rep. CMU-RI-TR-90-14, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 1990. 60

[NKGR06] NAYAR S. K., KRISHNAN G., GROSSBERG M. D., RASKAR R.: Fast separation of direct and global components of a scene using high frequency illumination. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 25*, 3 (July 2006), 935–944. 61

[NRDR05] NEHAB D., RUSINKIEWICZ S., DAVIS J., RAMAMOORTHI R.: Efficiently combining positions and normals for precise 3D geometry. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 24*, 3 (July 2005), 536–543. 59

[NRH*77] NICODEMUS F., RICHMOND J., HSIA J., GINSBERG W., LIMPERIS T.: *Geometrical considerations and nomenclature for reflectance*. Tech. rep., US Dept. of Commerce, National Bureau of Standards, October 1977. 11

[NSRS13] NGUYEN C., SCHERZER D., RITSCHEL T., SEIDEL H.-P.: Material editing in complex scenes by surface light field manipulation and reflectance optimization. *Computer Graphics Forum (CGF) - Proceedings of the EG 32*, 2 (May 2013), 185–194. 117

[NVI08] NVIDIA: CUDA CUBLAS library, reference manual v2.0, March 2008. `http://developer.download.nvidia.com/compute/cuda/2_0/docs/CUBLAS_Library_2.0.pdf`. 33

[NZG04] NEUBECK A., ZALESNY A., GOOL L. V.: Viewpoint consistent texture synthesis. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)* (September 2004), pp. 388–395. 156

[NZG05] NEUBECK A., ZALESNY A., GOOL L. V.: 3d texture reconstruction from extensive BTF data. In *Proceedings of the International Workshop on Texture Analysis and Synthesis (TEXTURE)* (October 2005), pp. 13–19. 60, 162

[NZI01] NISHINO K., ZHANG Z., IKEUCHI K.: Determining reflectance parameters and illumination distribution from a sparse set of images for view-dependent image synthesis. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (July 2001), vol. 1, pp. 599–606. 17, 18

[OT09] OSELEDETS I., TYRTYSHNIKOV E.: Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing 31*, 5 (October 2009), 3744–3759. 54

[OT10] OSELEDETS I., TYRTYSHNIKOV E.: TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications 432*, 1 (January 2010), 70 – 88. 54

[PCDS12] PALMA G., CALLIERI M., DELLEPIANE M., SCOPIGNO R.: A statistical method for SVBRDF approximation from video sequences in general lighting conditions. *Computer Graphics Forum (CGF) - Proceedings of the EGSR 31*, 4 (June 2012), 1491–1500. 17, 18

[PCF05] PATERSON J. A., CLAUS D., FITZGIBBON A. W.: BRDF and geometry capture from extended inhomogeneous samples using flash photography. *Computer Graphics Forum (CGF) - Proceedings of the EG 24*, 3 (August 2005), 383–391. 17, 18, 58, 59, 60

[PFG00] PELLACINI F., FERWERDA J. A., GREENBERG D. P.: Toward a psychophysically-based light reflection model for image synthesis. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 2000), pp. 55–64. 117

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 2000), pp. 465–470. 133

[PRK93] PATI Y., REZAIIFAR R., KRISHNAPRASAD P.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers* (November 1993), pp. 40–44. 42

[PS00] PORTILLA J., SIMONCELLI E. P.: A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision (IJCV) 40*, 1 (October 2000), 49–70. 131, 132, 133, 135, 142, 159

[RLW*09] RAY N., LÉVY B., WANG H., TURK G., VALLET B.: Material space texturing. *Computer Graphics Forum (CGF) 28*, 6 (September 2009), 1659–1669. 132, 145, 146, 149

[RMS*08] RUMP M., MÜLLER G., SARLETTE R., KOCH D., KLEIN R.: Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum (CGF) - Proceedings of the EG 27*, 2 (April 2008), 527–536. 14

[Row97] ROWEIS S.: EM algorithms for PCA and SPCA. In *Proceedings of the Conference on Advances in neural information processing systems (NIPS)* (December 1997), pp. 626–632. 26, 27, 29

[RSK10b] RUMP M., SARLETTE R., KLEIN R.: Groundtruth data for multispectral bidirectional texture functions. In *Proceedings of the Conference on Color in Graphics, Image and Vision (CGIV)* (June 2010), pp. 326–330. 13

[RT99] RUZON M. A., TOMASI C.: Color edge detection with the compass operator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 1999), vol. 2, pp. 2160–2166. 136

[Rus98] RUSINKIEWICZ S.: A new change of variables for efficient BRDF representation. In *Proceedings of the Eurographics Workshop on Rendering Techniques (EGWR)* (June 1998), pp. 11–22. 14, 81, 94, 95

[RZE08] RUBINSTEIN R., ZIBULEVSKY M., ELAD M.: *Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit*. Tech. Rep. CS-2008-08, Technion - Comp. Science Department, April 2008. 53

[SBLD03] SUYKENS F., BERGE K. V., LAGAE A., DUTRÉ P.: Interactive rendering with bidirectional texture functions. *Computer Graphics Forum (CGF) - Proceedings of the EG 22*, 3 (September 2003), 463–472. 19, 79

[SC78] SAKOE H., CHIBA S.: Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing 26*, 1 (February 1978), 43–49. 120

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 1, pp. 519–528. 59

[Sch94] SCHLICK C.: An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum (CGF) - Proceedings of the EG 13*, 3 (March 1994), 233–246. 63, 87

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *Proceedings of* *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008). 136

[SH12] SUN L., HAYS J.: Super-resolution from internet-scale scene matching. In *Proceedings of the IEEE International Conference on Computational Photography (ICCP)* (April 2012). 180

[Ska03] SKARBEK W.: Merging subspace models for face recognition. In *Proceedings of the International Conference on Computer Analysis of Images and Patterns* (August 2003), pp. 606–613. 27

[SKB10] SCHWENK K., KUIJPER A., BOCKHOLT U.: Modeling wavelength-dependent BRDFs as factored tensors for real-time spectral rendering. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP)* (May 2010), pp. 165–172. 20, 79

[SS95] SCHRÖDER P., SWELDENS W.: Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 1995), pp. 161–172. 18

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2003), pp. 167–178. 12, 19, 41

[SWI97] SATO Y., WHEELER M. D., IKEUCHI K.: Object shape and reflectance modeling from observation. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 1997), pp. 379–387. 17, 18

[SWRK11] SCHWARTZ C., WEINMANN M., RUITERS R., KLEIN R.: 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)* (October 2011), pp. 25–32. 15, 20, 92, 106, 107, 161

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic BRDFs. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 26*, 3 (July 2007), 27. 20, 79, 83

[TFG*13] TUNWATTANAPONG B., FYFFE G., GRAHAM P., BUSCH J., YU X., GHOSH A., DEBEVEC P.: Acquiring reflectance and shape from continuous spherical harmonic illumination. *ACM Transac-*

tions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 32, 4 (July 2013). 181

[TS12] TSAI Y.-T., SHIH Z.-C.: K-clustered tensor approximation: A sparse multilinear model for real-time rendering. *ACM Transactions on Graphics (TOG) 31*, 3 (June 2012), 19:1–19:17. 20, 53

[Tsa09] TSAI Y.-T.: *Parametric Representations and Tensor Approximation Algorithms for Real-Time Data-Driven Rendering*. PhD thesis, National Chiao Tung University, May 2009. 20, 53, 56

[TW04] TONIETTO L., WALTER M.: Morphing textures with texton masks. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)* (October 2004), pp. 348–353. 132

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 21*, 3 (July 2002), 665–672. 156

[VHC06] VOGIATZIS G., HERNANDEZ C., CIPOLLA R.: Reconstruction in the round using photometric normals and silhouettes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 2, pp. 1847–1854. 60

[VT04] VASILESCU M. A. O., TERZOPOULOS D.: TensorTextures: multilinear image-based rendering. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 23*, 3 (August 2004), 336–342. 20, 41, 47

[VTC05] VOGIATZIS G., TORR P. H. S., CIPOLLA R.: Multi-view stereo via volumetric graph-cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2005), pp. 391–398. 111

[WAKB09] WILLS J., AGARWAL S., KRIEGMAN D., BELONGIE S.: Toward a perceptual space for gloss. *ACM Transactions on Graphics (TOG) 28*, 4 (August 2009), 1–15. 117, 159

[Wal96] WALL M.: *GAlib: A C++ Library of Genetic Algorithm Components*, 2.4 ed. Mechanical Engineering Department, MIT, August 1996. `http://lancet.mit.edu/ga/`. 65

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 1992), pp. 265–272. 17

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 1992), pp. 255–264. 18

[WBS*04] WANG Z., BOVIK A. C., SHEIKH H. R., MEMBER S., SIMONCELLI E. P., MEMBER S.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. 85

[WDR11] WU H., DORSEY J., RUSHMEIER H.: A sparse parametric mixture model for BTF compression, editing and rendering. *Computer Graphics Forum (CGF) - Proceedings of the EG 30*, 2 (April 2011), 465–473. 17, 18, 155

[Wei02] WEI L.-Y.: *Texture Synthesis by Fixed Neighborhood Searching*. PhD thesis, Stanford University, November 2002. 132

[WHZ*08] WEI L.-Y., HAN J., ZHOU K., BAO H., GUO B., SHUM H.-Y.: Inverse texture synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 27*, 3 (August 2008), 52:1–52:9. 136

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (July 2000), pp. 479–488. 133

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics, State of the Art Reports* (April 2009), pp. 93–117. 133

[WLL*09] WEYRICH T., LAWRENCE J., LENSCH H. P. A., RUSINKIEWICZ S., ZICKLER T.: Principles of appearance acquisition and representation. *Foundation and Trends in Computer Graphics and Vision 4*, 2 (February 2009), 75–191. 12

[Woo80] WOODHAM R. J.: Photometric method for determining surface orientation from multiple images. *Optical Engineering 19*, 1 (February 1980), 139–144. 59

[WRO*12] WEINMANN M., RUITERS R., OSEP A., SCHWARTZ C., KLEIN R.: Fusing structured light consistency and Helmholtz normals for 3D reconstruction. In *Proceedings of the British Machine Vision Conference (BMVC)* (September 2012), pp. 108.1–108.12. 16, 57

[WSL04] WU E., SUN Q., LIU X.: Recovery of material under complex illumination conditions. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE)* (June 2004), pp. 39–45. 60

[WSRK11] WEINMANN M., SCHWARTZ C., RUITERS R., KLEIN R.: A multi-camera, multi-projector super-resolution framework for structured light. In *Proceedings of International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)* (May 2011), pp. 397–404. 15, 57, 73, 76, 162, 165

[WTL*06] WANG J., TONG X., LIN S., PAN M., WANG C., BAO H., GUO B., SHUM H.-Y.: Appearance manifolds for modeling time-variant appearance of materials. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 25*, 3 (July 2006), 754–761. 155

[WW07] WEIDLICH A., WILKIE A.: Arbitrarily layered micro-facet surfaces. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE)* (December 2007), pp. 171–178. 17

[WWHL07] WEISTROFFER R. P., WALCOTT K. R., HUMPHREYS G., LAWRENCE J.: Efficient basis decomposition for scattered reflectance data. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2007), pp. 207–218. 18, 180

[WWS*05] WANG H., WU Q., SHI L., YU Y., AHUJA N.: Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 24*, 3 (July 2005), 527–535. 20, 41, 45, 50, 52

[WXC*08] WU Q., XIA T., CHEN C., LIN H.-Y. S., WANG H., YU Y.: Hierarchical tensor approximation of multi-dimensional visual data. *IEEE Transactions on Visualization and Computer Graphics (VCG) 14*, 1 (January 2008), 186–199. 20, 41

[WY04] WU Q., YU Y.: Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 23*, 3 (August 2004), 364–367. 133

[WZT*08] WANG J., ZHAO S., TONG X., SNYDER J., GUO B.: Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 27*, 3 (August 2008), 41:1–9. 18

[XWT*09] Xu K., Wang J., Tong X., Hu S.-M., Guo B.: Edit propagation on bidirectional texture functions. *Computer Graphics Forum (CGF) - Proceedings of the Pacific Graphics 28*, 7 (October 2009), 1871–1877. 155

[YDMH99] Yu Y., Debevec P., Malik J., Hawkins T.: Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (August 1999), pp. 215–224. 17, 18, 60

[YXA04] Yu T., Xu N., Ahuja N.: Recovering shape and reflectance model of non-lambertian objects from multiple views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2004), vol. 2, pp. 226–233. 60

[ZDW*05] Zhou K., Du P., Wang L., Matsushita Y., Shi J., Guo B., Shum H.-Y.: Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics (VCG) 11*, 5 (September 2005), 519–528. 156

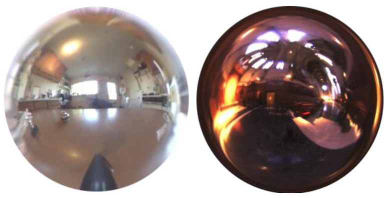
[ZERB05] Zickler T., Enrique S., Ramamoorthi R., Belhumeur P. N.: Reflectance sharing: Image-based rendering from a sparse set of images. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)* (June 2005), pp. 253–264. 21

[ZZV*03] Zhang J., Zhou K., Velho L., Guo B., Shum H.-Y.: Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics (TOG) - Proceedings of the ACM SIGGRAPH 22*, 3 (July 2003), 295–302. 132

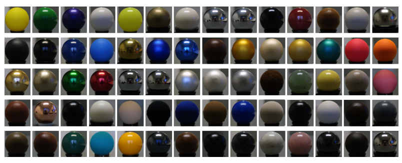
Data Sources

The following data sets used in this thesis have been taken from external sources. We would like to thank the authors for making them available to us.

Dragon Model
© 2007 UTIA, Academy of Sciences of the Czech Republic, and CGG, Czech Technical University in Prague
http://dcgi.felk.cvut.cz/cgg/eg07/index.php?page=dragon



Kitchen and Grace Cathedral Light Probes
© 1999 Paul Debevec
http://www.debevec.org/Probes/



MERL BRDF Database
© 2007 Mitsubishi Electric Research Laboratories
http://www.merl.com/brdf/

Texture Samples
© CGTextures
http://www.cgtextures.com