Acquisition, Synthesis and Rendering of Bidirectional Texture Functions

G. Müller, J. Meseth, M. Sattler, R. Sarlette and R. Klein

University of Bonn, Institute for Computer Science II

Abstract

One of the main challenges in computer graphics is still the realistic rendering of complex materials such as fabric or skin. The difficulty arises from the complex meso structure and reflectance behavior defining the unique look-and-feel of a material. A wide class of such realistic materials can be described as 2D-texture under varying light- and view direction namely the Bidirectional Texture Function (BTF). Since an easy and general method for modeling BTFs is not available, current research concentrates on image-based methods which rely on measured BTFs (acquired real-world data) in combination with appropriate synthesis methods. Recent results have shown that this approach greatly improves the visual quality of rendered surfaces and therefore the quality of applications such as virtual prototyping. This STAR will present in detail the state-of-the-art techniques for the main tasks involved in producing photo-realistic renderings using measured BTFs

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Digitizing and scanning I.4.1 [Digitization and Image Capture]: Reflectance I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture

1. Introduction

The realistic visualization of materials and object surfaces is one of the main challenges in todays computer graphics. Traditionally the geometry of a surface is modeled explicitly (e.g. with triangles) only up to a certain scale while the micro structure responsible for the reflectance behavior of a material is simulated using relatively simple analytical BRDF models. For special groups of materials such as metals or shiny plastics these models are able to generate a more or less correct approximation of real appearance, but they are neither suited nor designed for all materials out there in the real world. Furthermore these models are not easy to parameterize in order to achieve a desired effect, because the parameters may not correspond to some intuitive or physical meaning. Things become even more complicated, if one wants to simulate the so-called meso structure of a complex inhomogeneous material. These structures are in-between the macro-scale geometry modeled by triangles and the micro-scale geometry modeled by analytical BRDF models and play a very important role in defining and transporting the unique *Look & Feel* of a material. If they are not reproduced accurately the images will look artificial.

Driven by these limitations of traditional modeling and the enormous growth of memory capacity (256MB of random access memory are almost standard equipment of current off-the-shelf graphics hardware) image-based methods become increasingly important. However, applications of purely image-based techniques like Light Field Rendering (e.g. [LH96]) are still limited to special domains. This explains the fact that texture mapping as a combination of images and classical geometric modeling remains the most popular technique for rendering meso structure and is used in any field from real-time rendering to global illumination. Nevertheless, textures are able to represent only a fraction of the appearance of complex materials. In fact one has to assume that the material is flat and diffuse, and visually important light- and view-dependent effects like shadowing, masking, complex non-diffuse reflection etc. are not captured.

These effects are included in the Bidirectional Texture Function (BTF), a 6-dimensional texture representation introduced by Dana et al. [DvGNK99] which extends the common textures by dependence on light- and view-direction. Using densely sampled BTFs instead of ordinary 2D textures makes no real conceptual difference and thus consists

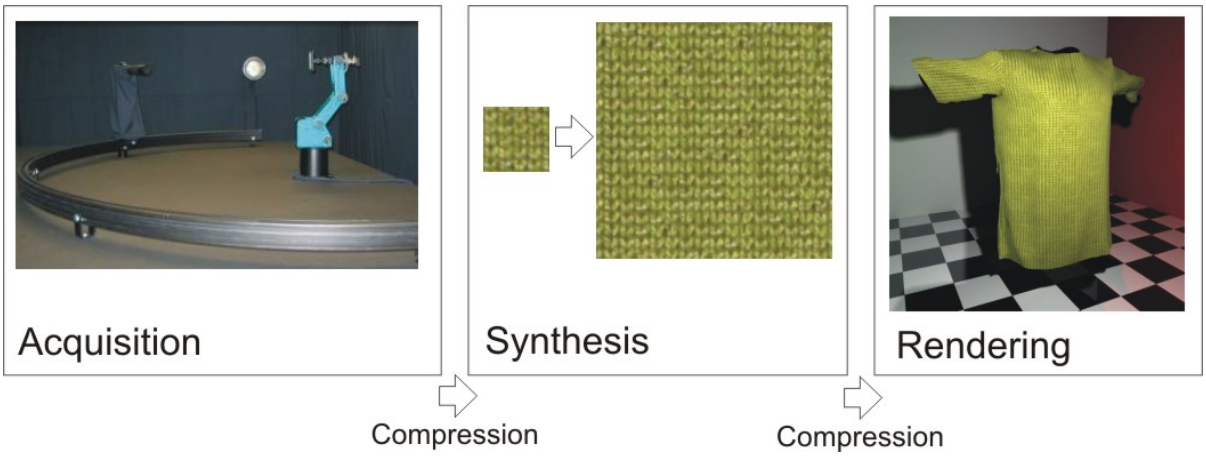

Figure 1: *In principle there is no conceptual difference between using BTFs and 2D-textures. The compression steps can be omitted, but in fact they are absolutely necessary to achieve acceptable frame-rates and processing times.*

of the following main steps that are also illustrated in figure 1:

- Texture acquisition
- Synthesis
- Rendering

Between these steps compression can be inserted to reduce the storage requirements of the texture. These compression steps are optional for 2D texturing, but as it will become clear in the following, they are absolutely necessary for BTFs to achieve acceptable frame-rates and processing times. As shown in figure 2, the visual impression of complex materials generated by rendering from a sampled BTF is improved significantly compared to simple diffuse texture mapping. So why is the usage of ordinary 2D textures still more widespread?

First, current acquisition systems are expensive and the measurement process is time consuming as the directional dependent parameters (light- and view-direction) have to be controlled very accurately. Otherwise the resulting data will be incorrect. Furthermore, the size of measured BTFs lies in a range from hundreds of megabytes to several gigabytes. This hampers both synthesis and rendering so that only effective compression techniques can provide a solution.

Due to these limitations BTF rendering is still not mature enough for industrial applications. Nevertheless, there is a growing demand for interactive photo-realistic material visualization in the industry. For special applications such as high-end virtual reality environments, BTF rendering can already satisfy these demands. Simple material representations like 2D texture or bump-maps sooner or later will be replaced by more complex representations that capture all the subtle effects of general light-material interaction. This STAR intends to give an overview over to what extent current BTF research can contribute to this process.

The paper is structured as follows: Sections 2–5 will cover in detail the before mentioned main steps of using image-based material representations, i.e. acquisition, compression, synthesis and rendering of BTFs. In section 6 current and future applications of BTFs are discussed and the last section gives a summary and concludes the STAR.

2. Acquisition

The acquisition of 2D textures is a very simple process which can be performed using a standard 2D scanner or an off-the-shelf digital camera and an image-processing software. On the contrary, the acquisition of BTFs requires a complex and controlled measurement environment. As BTF acquisition is physical measurement of real-world reflection, special attention has to be paid to the device calibration and image registration. Otherwise the measurements will contain inaccuracies which may generate visible rendering artifacts. In this section current techniques for accurate measurements of BTFs will be presented. As an introduction the following subsection will give some theoretical background and a short overview of related image based techniques for physical reflectance measurement.

2.1. Principles of Reflectance Measurement

To measure the reflectance behavior of a surface, a set of parameters which describe the process of light scattering on a surface has to be defined. For this purpose let us imagine the path of a photon through a surface as illustrated in figure 3: It hits the surface at time t_i, position x_i and with an associated wavelength λ_i. Given a fixed local coordinate frame at every surface point, the incoming direction of the photon motion can be determined as (θ_i, ϕ_i). The photon travels through the material and leaves the surface at position x_r, time t_r, with possibly changed wavelength λ_r in the direction (θ_r, ϕ_r).

Figure 2: *Comparing simple texture mapping and rendering from a sampled BTF*

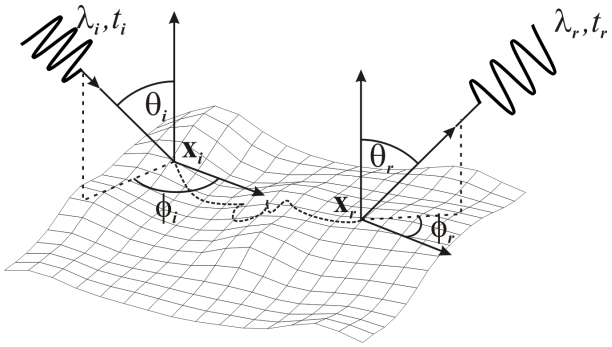

Figure 3: *The parameters of general light-material interaction*

According to this description, the reflection of light is a process involving 12 parameters! Since the measurement of a 12-dimensional function is currently not practical, additional simplifying assumptions have to be made. Typically these are the following:

- light transport takes zero time ($t_i = t_r$)
- reflectance behavior of the surface is time invariant ($t_0 = t_i = t_r$)
- interaction does not change wavelength ($\lambda_i = \lambda_r$)
- wavelength is discretized into the three color bands red, green and blue (consider only $\lambda_{r,g,b}$)

We now have arrived at an 8-dimensional function, which is called the bidirectional surface scattering distribution function (BSSRDF) [NRH*77]. It describes the light transport between every point on the surface for any incoming and outgoing direction. Figure 4 shows further simplifications of the BSSRDF by fixing some of the parameters. For these simpler models measurement systems have been proposed and since this work is related to BTF-measurement, a short overview over some of this work is given in the following.

Inclusion of Subsurface Scattering Assuming a homogeneous material yields the BSSDF for which Jensen et al. [JMLH01] designed a practical model based on a dipole approximation. They derived the parameters of this model (absorption cross section σ_a and reduced scattering cross section σ_s') from a single HDR-image. They achieved good results for materials like marble and even fluids like milk. Recently Goesele et al. [GLL*04] presented a measurement setup for translucent inhomogeneous objects which takes multiple images of an object illuminated with a narrow laser beam. They had to assume diffuse surface reflection and strong subsurface scattering because they did not measure the angular dependency.

Uniform and Opaque Materials For many applications it is convenient to drop spatial dependence and consider reflection taking place on an infinitesimal surface element. This process is described by the 4-dimensional BRDF and the classical measurement device for this quantity is the *gonioreflectometer* which samples the angular dependency sequentially by positioning a light source and a detector at various directions from the sample [NRH*77]. Several methods attempted to reduce measurement times by exploiting CCD-chips for taking several BRDF-samples at once. Ward [War92] used a hemispherical half-silvered mirror and

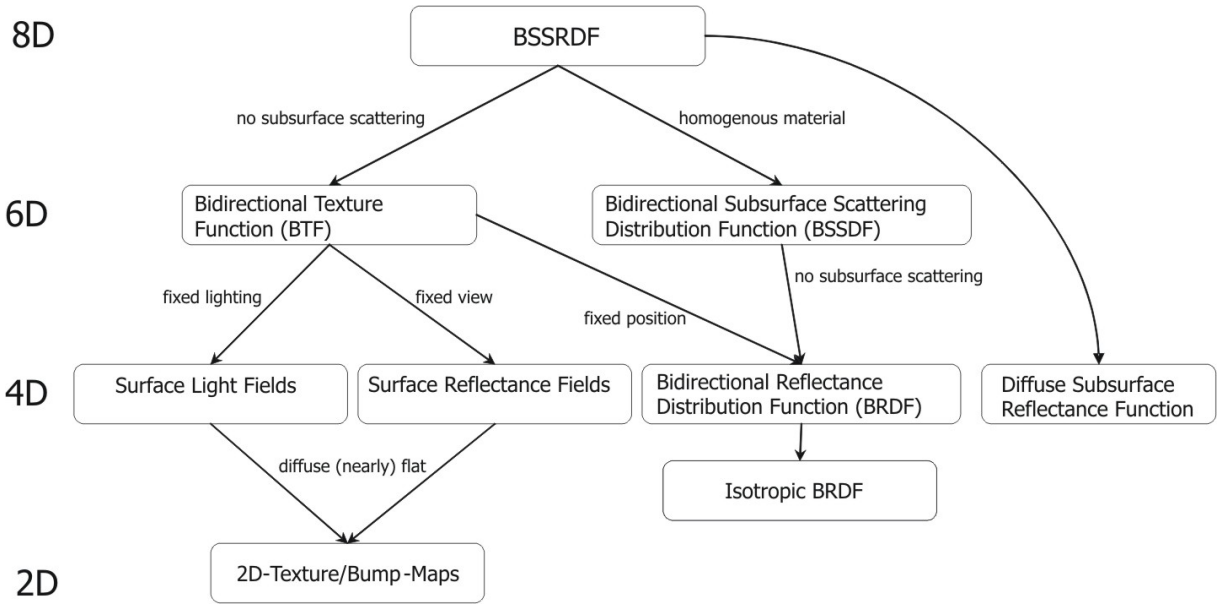

Figure 4: *A hierarchy of reflectance functions*

a camera with a fish-eye lens to acquire the whole exitant hemisphere of the flat probe at once. Alternatively one could take images from a curved sample as it was done by Marschner et al. [MWL*99] and Matusik et al. [MPBM03b]. The latter work was tailored to measuring isotropic BRDFs. It demonstrates also how measurement times can be significantly reduced by using sophisticated learning algorithms and a database of densely acquired BRDFs.

Surface Light-Fields While simple texture-mapping could also be interpreted as image-based rendering, the term became really popular with the works on light fields [LH96] and lumigraphs [GGSC96]. They sample the pencil of light rays flowing through a point in space using an array of cameras. The corresponding quantity parameterized over surfaces is the surface light field [MRP98, WAA*00]. It records spatially varying view dependence neglecting subsurface scattering and fixing illumination. The surface light field is measured using many images from different viewpoints of an object with known geometry.

Surface Reflectance-Fields Another very popular variant of image-based rendering is image-based relighting. In this case one takes many images of a surface lit by a set of basis lights. The linear nature of light reflection now allows relighting the surface with arbitrary lighting projected onto the light basis. Debevec et al. [DHT*00] built a so called *light-stage* which records the appearance of human face while a light source is rotating around the face. For nearly planar objects Malzbender et al. [MGW01] constructed a hemispherical gantry attached with 50 strobe light sources. They also introduced *Polynomial Texture Maps* (PTM), a compact representation for the acquired data that is especially suited for diffuse materials. A very complex acquisition setup was built by Matusik et al. [MPN*02]. It captures the object from varying view-points and handles objects with complex silhouettes using multi-background matting techniques and thus actually samples a 6D slice of the BSSRDF. Masselus et al. [MPDW03] fixed the viewpoint again but instead used a spatially located light basis which enabled the relighting of parts of the surface.

Bump-Maps Bump mapping, first introduced by Blinn [Bli77], is a very popular extension to simple texture mapping. Bump maps consist of an array of normals that are used instead of the normals of the base geometry and can be used to simulate a bumpy surface with small height variations. The normals of a bumpy and not too specular surface can be reliably measured using photometric stereo. This technique captures three or more images from a single viewpoint of a surface illuminated from different directions. Rushmeier et al. [RTG97] for example built a setup which takes five images lit from different directions. Even low-cost graphics boards are now able to render bump maps in real-time and they are used extensively for example in video games. Note however, that due to its inherent limitations many visually important effects are not reproduced by bump-mapping.

2.2. BTF-Measurement

After shortly introducing general reflectance measurement we will now concentrate on the BTF as a measured six-dimensional slice of the general light scattering function of

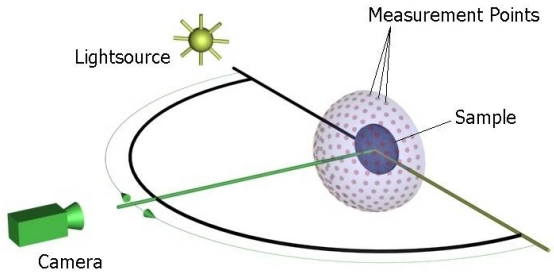

Figure 5: *Capturing the BTF of planar sample using gonioreflectometer-like setup with a fixed light source, sampleholder and a moving camera.*

a surface S:

$$\mathbf{BTF}_{rgb}(\mathbf{x},\theta_i,\phi_i,\theta_r,\phi_r) := \int_S \mathbf{BSSRDF}_{rgb}(\mathbf{x}_i,\mathbf{x},\theta_i,\phi_i,\theta_r,\phi_r)d\mathbf{x}_i$$

It this sense the BTF integrates subsurface scattering from neighboring surface locations, as it is done by existing measurement setups. Nevertheless this definition allows also for the mathematical interpretation of the BTF as spatially varying BRDF. The corresponding previous work can be grouped roughly into two categories.

The first group captures the geometry of a small object and its reflection properties parameterized over the surface i.e. the spatially varying BRDF. The works of Furukawa et al. [FKIS02] and Lensch et al. [LKG*01] fall into this category. They capture the geometry of the object using laser scanners and take several images under varying light and viewing conditions. The methods differ in their data representation. While Furukawa et al. map the images onto the triangles of the model and compress the appropriately reparameterized data using tensor product expansion, Lensch et al. fitted the data to a parametric reflection model. In order to cope with insufficient sample density they used an iterative clustering procedure.

The second group aims at capturing the appearance of an opaque material independently from geometry. These methods have in common, that they capture the BTF of a planar material sample. The acquired data can be used instead of simple 2D-textures and mapped onto arbitrary geometry. Since this STAR is concerned with BTFs as a generalized texture representation, in the following these methods are described in detail.

2.2.1. Gonioreflectometer-like Setup with CCD-Chips

The most common approaches use a gonioreflectometer-like setup with a CCD-chip instead of a spectrometer in order to capture the spatial variation of reflection. This approach has proved to be reliable and several variations of it have been published. However its drawback are the long measurement times.

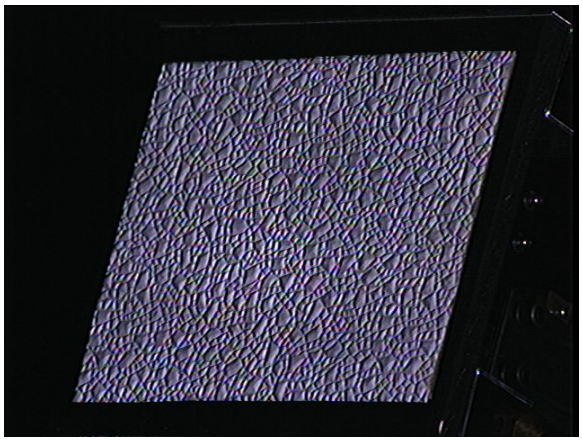

Figure 6: *One image of the sample ROUGH PLASTIC from the CUReT database.*

The first measurement system that used such a gonioreflectometer-like setup as depicted in figure 5 was presented in the pioneering work of Dana et al. [DvGNK99]. Their system takes 205 images of isotropic materials which is a too sparse sampling for high-quality rendering, in particular for rough surfaces and materials with strong specular pikes. Even though they mentioned the possibility of using the data for rendering, the original intent of the work was building up a material database for computer vision related tasks such as texture recognition, texture segmentation and shape-from-texture. They measured 61 real-world surfaces and made them available through the CUReT [Col] database. Figure 6 shows a sample from the database. Similar, but improved version of the measuring system were described in [MLH02] and [HEE*02]. Some measurements of the latter system are now also publicly available through the BTF database Bonn [BTF]. We will describe this system in greater detail in the following.

Measurement Setup The measurement setup is designed to conduct an automatic measurement of a BTF that also allows the automatic alignment and postprocessing of the captured data. A high-end digital still camera is used as image sensor. The complete setup, especially all metallic parts of the robot, are covered with black cloth or matte paint, with strong diffuse scattering characteristics.

The system uses planar samples with the maximum size of 10×10 cm. In spite of these restrictions, measurement of a lot of different material types, e.g. fabrics, wallpapers, tiles and even car interior materials is possible. As shown in figure 8, the laboratory consists of a HMI (Hydrargyrum Medium Arc Length Iodide) bulb, a robot holding the sample and a rail-mounted CCD camera (Kodak DCS Pro 14N). Table 1 shows the sampling density of the upper hemisphere for light and view direction which results in $n = 81$ unique

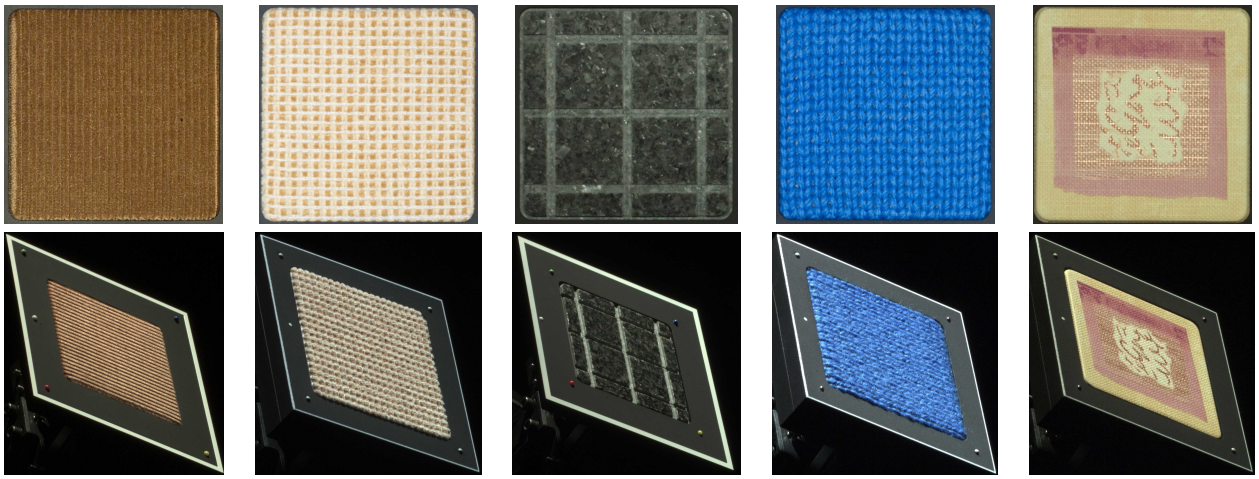

Figure 7: *Measured BTF samples; from left to right (top row): CORDUROY, PROPOSTE, STONE, WOOL and WALLPAPER. Bottom row: perspective views* $(\theta = 60, \phi = 144)$ *of the material and sample holder with illumination from* $(\theta = 60, \phi = 18)$*. Note the mesostructure and changing colors.*

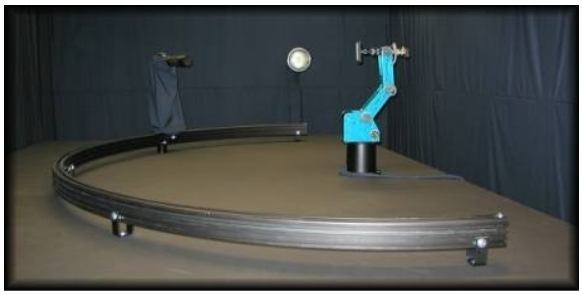

Figure 8: *Measurement setup of the Bonn-System consisting out of an HMI lamp, a CCD camera and a robot with a sample holder.*

θ [°]	$\Delta\phi$ [°]	No. of images
0	−*	1
15	60	6
30	30	12
45	20	18
60	18	20
75	15	24

Table 1: *Sampling of viewing and illumination angles of the BTF database Bonn.* * *= only one image taken at* $\phi = 0°$

directions for camera and light position. Hence, 6561 pictures of a sample are taken.

Figure 7 shows several measured samples. The top row shows frontal views of the different samples, whereas the bottom row shows oblique views. In the latter case especially the mesostructure of the samples becomes visible. Each raw image is 12 megabytes in size (lossless compression) with a resolution of 4500×3000 pixels (Kodak DCR 12-bit RGB format). With this setup, the measuring time is about 14 hours, where most of the time is needed for the data transfer from the camera to the host computer.

Calibration To achieve high-quality measurements, the equipment has to be calibrated.

- To compensate the positioning error due to the robot and the rail system, one has to track the sample holder mounted on the robot arm using the camera. Experiments determined that these errors are small in the described setup. Therefore, marker points, which are placed on the sample holder, are detected only during the post-processing phase, allowing a software jitter correction.
- A geometric calibration has to be applied to the camera to reduce geometric distortion, caused by the optical system of the camera.
- For each sample to be measured, the aperture of the camera is adjusted in such a way that the number of saturated or dark pixels in the pictures is minimized given a fixed aperture during the measurement process.
- To achieve the best possible color reproduction, the combination of the camera and the light source has to be color calibrated. For the measurement of the camera color profile a special CCD-Camera standard card (Gretag Macbeth - Color Checker DC) is used.

Data Postprocessing After the measurement the raw image data is converted into a set of rectified, registered images capturing the appearance of the material for varying

light and view directions. Now a complete set of discrete reflectance values for all measured light and viewing directions can be assigned to each texel of a 2D texture.

Registration is done by projecting all sample images onto the plane which is defined by the frontal view ($\theta = 0, \phi = 0$). To be able to conduct an automatic registration, borderline markers were attached to the sample holder plate, see figure 7. After converting a copy of the raw data to a binary image, standard image processing tools are used to detect the markers. In following steps the mapping (which maps these markers to the position of the markers in the frontal view) is computed and utilized to fill the registered image with appropriate colors.

To convert the 12-bit RGB images stored in the proprietary format of the camera manufacturer to standard 8 bit RGB file formats, the standard color profiles provided with the Camera SDK (look and output profile) and camera (tone curve profile) are applied to the image. The most appropriate 8 bit color range is extracted after applying an exposure gain to the converted data.

After this postprocessing step, the final textures are cut out of the raw reprojected images and resized appropriately (256×256 pixels in size for probes in the database, up to about 800×800 in principle). A final dataset with 256×256 pixels spatial resolution has a data amount of 1.2GB.

2.2.2. Using Video Cameras

Koudelka et al. [KMBK03] presented a system resembling the before mentioned ones, but it fixes the image sensor (a Canon XL-1 digital video camera) and moves the light source (a white LED mounted on a robot arm). The employed hemisphere sampling results in a dataset of about 10.000 images. Due to the use of a video camera with relatively low resolution compared to a high-end still camera, a measurement takes about 10 hours. Samples from this system are publicly available for research purposes and include interesting natural materials like lichen or moss and man-made materials like carpet or even lego bricks.

2.2.3. Using Mirrors

Inspired by BRDF-measurement techniques, it has also been proposed to use mirrors for BTF measurement in order to make several measurements in parallel. The advantages in measurement time and registration precision (no moving parts) are accompanied by a number of disadvantages. Multiple reflection on mirrors (not perfect reflectors) cause low image quality and lead to a difficult color calibration. Angular sampling and spatial resolution are often coupled, i.e. a higher angular resolution leads to a lower spatial resolution. Han et al. [HP03] presented a kaleidoscope-based measurement system. Since results are not publicly available the measurements cannot be judged reliably.

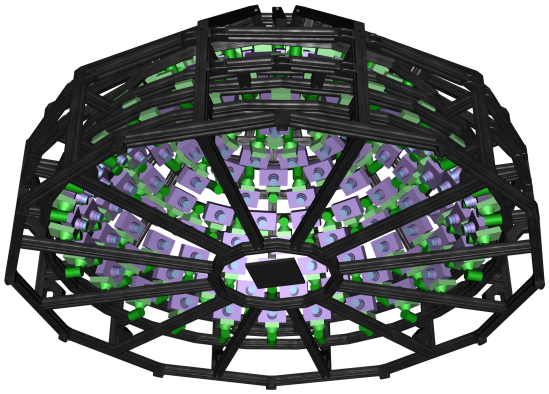

Figure 9: *Sketch of the proposed camera array. 151 digital cameras are mounted on a gantry, focusing on the sample, which is placed in the center of the hemisphere.*

2.2.4. Using a camera array

For a fast high quality acquisition of BTFs in our recent approach we propose an array of 151 digital still cameras mounted on a hemispherical gantry (see Figure 9 for a sketch of the setup). A similar gantry with mounted light sources was used by Malzbender et al. [MGW01] to capture PTMs. Although the setup is costly to build, a camera array is capable of measuring many samples in a short time. Due to the parallel structure of the acquisition, the example setup would be capable of capturing a BTF dataset of $151^2 = 22801$ images in less than one hour. No moving parts are needed. Therefore, the region of interest (ROI) is known for every camera and can be extracted at subpixel precision. Hence, there is no need for a time-consuming detection of the ROI, the post-processing (reprojection, geometric correction, color correction) is fast enough to be done in parallel to the measurement. The angular resolution depends on the number of cameras and the spatial resolution on the imaging chips. This will result in a high angular resolution; every measured direction represents an average solid angle of only 0.04161 steradians. The spatial resolution would be up to 280DPI for a resulting BTF texture size of 1024x1024. As light sources, the built-in flash lights of the cameras will be used.

2.2.5. Discussion

Currently only the standard gonioreflectometer-like measurement setups have proven that they can be used to capture high-quality BTFs reliably. Their drawback is the speed - several hours is too long and makes measured BTFs an expensive resource. Using mirrors may be a promising approach in the future, but the quality of the measurements of current systems remains dubious. Using a camera array will

greatly reduce measurement times at the expense of the costs for a large number of cameras.

3. Compression

Due to its huge size the pure image-based representation of a BTF consisting of the thousands of images taken during the measurement process is neither suitable for rendering nor for synthesis. In order to achieve real-time frame rates and acceptable synthesis times, some sort of data-compression has to be applied. Such a method should of course preserve as much of the relevant features of the BTF as possible, but should also exploit the redundancy in the data in an efficient way and provide a fast, preferably real-time decompression stage. An optimal method would achieve high compression rates with low error and real-time decompression. For integration into current real-time rendering systems an implementation of the decompression stage on modern GPUs would be also of great value.

Most existing compression techniques interpret the BTF as shown in figure 10. As a collection of discrete textures

$$\left\{T_{(\mathbf{v},\mathbf{l})}\right\}_{(\mathbf{v},\mathbf{l})\in\mathcal{M}},$$

where $\mathcal{M}$ denotes the discrete set of measured view- and light-directions, or as a set of spatially varying *apparent* BRDFs (ABRDF, the term was introduced in a paper of Wong et al. [WHON97]):

$$\{B_{\mathbf{x}}\}_{\mathbf{x}\in I\subset\mathbf{N}^2}$$

Note, that ABRDFs do not fulfill physically demanded properties like reciprocity, since they include scattering effects from other parts of the surface. As illustrated in figure 11 they can also contain a factor $(\mathbf{n}\cdot\mathbf{l})$ between incident direction and surface normal and strong effects from meso-scale shadowing and masking.

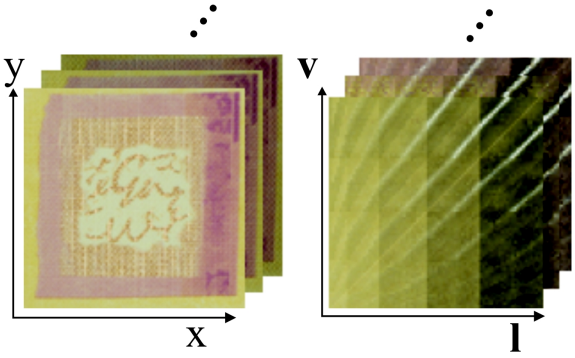

Figure 10: *Two arrangements of the BTF data: As set of images (left) and as set of ABRDFs (right).*

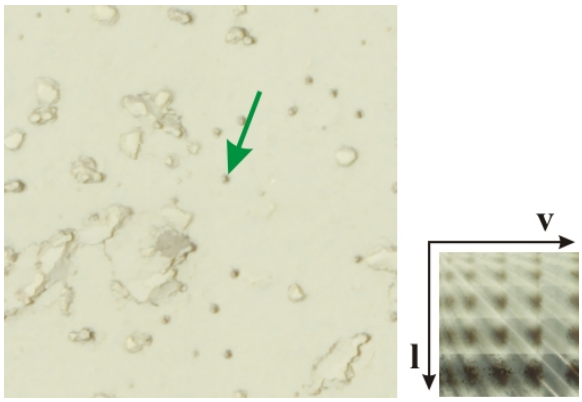

Figure 11: *An ABRDF (right) from the PLASTERSTONE BTF (left). While the reflectance of the white plaster alone is quite regular, the holes introduce strong meso-scale shadowing and masking.*

3.1. Fitting Analytical BRDF-Models

As mentioned in the introduction to section 3 already, BTFs can be understood as spatially varying ABRDFs. Therefore, a natural approach for compressing BTFs is via a pixel-wise representation using BRDF models which are fitted to the synthetic or measured BTF data. Candidate BRDF models need to be efficiently computable to achieve real-time capabilities. Therefore, fitting either the widely used Phong [Pho75] model, the Blinn [Bli77] model, the model of Ward [War92], the Cosine-Lobe model of Lewis [Lew93] or the Generalized Cosine-Lobe model of Lafortune et al. [LFTG97] to the measured data leads to straightforward extensions from BRDF to BTF representations.

3.1.1. Lafortune Lobes

The simplest BTF model based on analytic function fitting was published by McAllister et al. [MLH02] and is directly based on the Lafortune [LFTG97] model. Lafortune et al. propose to approximate the BRDF by a sum of lobes

$$s(\mathbf{v},\mathbf{l}) = \left(\mathbf{v}^t\cdot\mathbf{M}\cdot\mathbf{l}\right)^n \qquad (1)$$

with $\mathbf{v}$ and $\mathbf{l}$ denoting local view and light direction respectively, while the general 3×3 matrix $\mathbf{M}$ and the exponent n define the lobe.

To fit these parameters to the reflectance properties of a synthetic or measured BRDF, non-linear fitting methods like the Levenberg-Marquardt algorithm [PFTV92] are employed. Fitting the complete matrix allows for very general BRDFs but is very time consuming. Therefore, McAllister et al. decided to employ a more restricted, diagonal matrix $\mathbf{D}$, since fitting and rendering efforts are significantly reduced without a major loss in rendering quality. Thus, they use the following, spatially varying lobes:

$$s_{\mathbf{x}}(\mathbf{v},\mathbf{l}) = \left(\mathbf{v}^t\cdot\mathbf{D}_{\mathbf{x}}\cdot\mathbf{l}\right)^{n_{\mathbf{x}}}. \qquad (2)$$

This results in the following BTF approximation:

$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \rho_{d,\mathbf{x}} + \sum_{j=1}^{k} \rho_{s,\mathbf{x},j} \cdot s_{\mathbf{x},j}(\mathbf{v},\mathbf{l}) \qquad (3)$$

where ρ_d and ρ_s denote diffuse and specular albedo, specified as RGB values, and k is the number of lobes.

The model requires only a few parameters to be stored per pixel resulting in a very compact material representation (about 2 MB per material depending on the spatial resolution and number of lobes). Due to the expensive non-linear minimization, the number of Lafortune lobes is practically limited to about 4 lobes. Therefore the method achieves pleasing results only for a very limited range of materials with minor surface height variation.

3.1.2. Scaled Lafortune Lobes

BRDF models were not designed for the spatially varying ABRDFs which can contain strong effects from meso-scale shadowing, masking (figure 11). Therefore, more specialized models for ABRDFs were developed which also try to model some of these effects.

Daubert et al. [DLHS01] proposed a material representation, which is also based on the Lafortune model but includes an additional, multiplicative term $T_{\mathbf{x}}(\mathbf{v})$ modelling occlusion. Following their proposal, the BTF is evaluated as follows:

$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx T_{\mathbf{x}}(\mathbf{v}) \cdot \left(\rho_{d,\mathbf{x}} + \sum_{j=1}^{k} s_{\mathbf{x},i}(\mathbf{v},\mathbf{l}) \right). \qquad (4)$$

The view-dependent lookup-table T is defined per pixel and therefore the model requires significantly more parameters to be stored. It is thus necessary to combine this method with quantization approaches when handling materials that require significant spatial resolution. The model, as presented originally, was intended to independently represent the three channels of the RGB model by fitting individual Lafortune lobes and lookup-tables for each color channel.

3.1.3. Reflectance Fields

As a qualitative improvement over the previous method, Meseth et al. [MMK03a, MMK04a] published an approach to BTF rendering based on fitting a set of functions to the reflectance fields of the BTF only and performing a simple linear interpolation for view directions not contained in the measured set. Following this proposal, the BTF is evaluated as follows:

$$BTF(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{v \in N(\mathbf{v})} w_{\mathbf{x},v} \cdot RF_{\mathbf{x},v}(\mathbf{l}) \qquad (5)$$

Here, $N(\mathbf{v})$ denotes the set of closest view directions (a subset of the measured view directions), $w_{\mathbf{x},v}$ denotes the spatially varying interpolation weight, and $RF_{\mathbf{x},v}$ is the spatially varying reflectance field for view direction v which is approximated either by a biquadratic polynomial following the Polynomial Texture Map approach of Malzbender et al. [MGW01] or adopting Lafortune lobes as follows:

$$RF_{\mathbf{x},v}(\mathbf{l}) \approx \rho_{d,\mathbf{x}} + \rho_{s,\mathbf{x},v}(\mathbf{l}) \cdot \sum_{i=1}^{k} s_{\mathbf{x},v}(\mathbf{l}) \qquad (6)$$

with $s_{\mathbf{x},v}$ similar to a spatially varying Lafortune lobe but for fixed view direction and k being the number of lobes.

Since the reflectance fields are fitted per pixel and measured view direction, the amount of parameters necessary to evaluate the model is higher than for the scaled Lafortune lobes model. Like the model of McAllister et al. [MLH02], the approach is designed for efficient rendering and therefore the lobes are intended to compute luminance values that scale the RGB color albedo instead of fitting individual lobes for each color channel. Unlike previous methods based on function fitting, the approach requires interpolation between view directions, since the reflection fields are defined for fixed view directions.

3.1.4. Reflectance Field Polynomials

Recently, Filip and Haindl [FH04] suggested an even more accurate model based on the idea of Lafortune lobes: instead of computing reflectance fields by summing lobe contributions like Meseth et al. [MMK03a], they interpolate view and light-dependent polynomials:

$$RF_{\mathbf{x},v}(\mathbf{l}) \approx \sum_{l \in N(\mathbf{l})} w_l \sum_{i=1}^{k} a_{i,\mathbf{x},v,l} \left(\rho_{s,v} \cdot s_{\mathbf{x},v}(\mathbf{l}) \right)^{i-1}. \qquad (7)$$

Here, a denotes the coefficients of the polynomial, $s_{\mathbf{x},v}$ is defined as in equation 6 and w_l denotes interpolation weights for the contributions of the nearest light directions $N(\mathbf{l})$ which is a subset of the measured light directions.

Although approximation quality is superior to the previously mentioned approaches based on analytic function fitting and the data requirements are comparable to those of Meseth et al. [MMK03a], the evaluation of the BTF requires substantially more computation time due to the necessary interpolation of both view and light direction. Especially if applied to each color channel individually, as intended by the authors, this drawback severely limits use in real-time applications. Other applications areas, like texture synthesis - for which the model was intended - or offline rendering, might still find this method useful.

3.2. Linear Basis Decomposition

Using parametric BRDF-models fitted to the measured data per pixel has some drawbacks concerning realism. Many models were originally designed to model only a particular class of uniform materials and all models are only an

approximation of real reflectance using some simplifying assumptions about the underlying physical process (refer to the recent work of Matusik et al. [MPBM03a] on data-driven reflectance modeling for a more detailed discussion of this topic). The situation becomes even worse for the apparent BRDFs since they contain additional complex effects resulting from the surrounding meso structure.

One way to get rid of this problem would be the relaxation of the restricting assumptions of BRDF modeling and the interpretation of the measured data as a multi-dimensional signal. Then general signal-processing techniques such as Principal Component Analysis (PCA) [Jol86, PTVF92] can be applied. PCA minimizes the variance in the residual signal and provides the in a least-squares sense optimal affine-linear approximation of the input signal. We use the terms PCA and Singular Value Decomposition (SVD) synonymously during the rest of this section, since the principal components of the centered data matrix X are the columns of the matrix V with $X = U\Lambda V^T$ being the SVD of X.

PCA has been widely used in the field of image-based rendering to compress the image data. For example Nishino et al. [NSI99] applied PCA to the reparameterized images of an object viewed from different poses and obtained so-called *eigen-textures*. Matusik et al. [MPN*02] compressed the pixels of the captured reflectance field applying PCA to 8 by 8 image blocks.

The several BTF-compression methods that use PCA differ mainly in two points: (i) the slices of the data to which PCA is applied independently and (ii) how these slices are parameterized.

3.2.1. Per-Texel Matrix Factorization

One approach especially suited for real-time rendering applies PCA to the per-texel ABRDFs. Such methods were developed in the context of real-time rendering of arbitrary BRDFs at the time when the Phong-model was the state of the art in real-time rendering. The original idea as introduced by Kautz and McCool [KM99] can be stated as follows: Given the 4-dimensional BRDF, find a factorization into a set of 2-dimensional functions:

$$B_{\mathbf{x}}(\mathbf{v},\mathbf{l}) \approx \sum_{j}^{c} g_{\mathbf{x},j}(\pi_1(\mathbf{v},\mathbf{l}))h_{\mathbf{x},j}(\pi_2(\mathbf{v},\mathbf{l})) \tag{8}$$

The functions π_1 and π_2 are projection functions which map the 4D-dimensional BRDF parameters $(\mathbf{v},\mathbf{l})$ to a 2D-space. These projection functions have to be chosen carefully, because the parameterization significantly affects the quality of low-term factorizations. Given such a factorization real-time reconstruction of the BRDF using graphics hardware becomes easy, since the functions $g_{\mathbf{x},j}, h_{\mathbf{x},j}$ can be stored in texture maps and combined during rendering. A trade-off between quality and speed is possible by controlling the number of terms c.

Several methods to find such factorizations have been proposed. Given the sampled values of the BRDF arranged in a 2D-matrix X the SVD of X provides the solution with the lowest RMS-error. But the resulting functions contain negative values which may be problematic for a GPU implementation and the RMS-error may not be the perceptually optimal error metric. As an alternative McCool et al. [MAA01] presented a technique called Homomorphic Factorization (HF). Instead of using a sum of products they approximate the BRDF by an arbitrary number of positive factors:

$$B_{\mathbf{x}}(\mathbf{v},\mathbf{l}) \approx \prod_{j}^{c} p_{\mathbf{x},j}(\pi_j(\mathbf{v},\mathbf{l})) \tag{9}$$

A solution is computed by minimizing RMS-error in the logarithmic space which in fact minimizes *relative* RMS-error in the original minimization problem which was found to be perceptually more desirable. Furthermore, the resulting factors are positive and an arbitrary number of projection functions π_i can be used which allows for highly customized factors that capture certain BRDF features. This is of special importance for the ABRDFs from a measured BTF. They contain horizontal and vertical features like shadowing and masking and also diagonal features like specular peaks. Depending on the parameterization a simple single term expansion can capture only the one or the other.

Recently Suykens et al. [SvBLD03] presented a method called Chained Matrix Factorization (CMF) which encompasses both previous factorization methods by accommodating the following general factorization form:

$$B_{\mathbf{x}}(\mathbf{v},\mathbf{l}) \approx \prod_{j}^{d} \sum_{k}^{c_j} P_{\mathbf{x},j,k}(\pi_{j,1}(\mathbf{v},\mathbf{l}))Q_{\mathbf{x},j,k}(\pi_{j,2}(\mathbf{v},\mathbf{l})). \tag{10}$$

Such a chained factorization is computed using a sequence of simple factorizations using for example SVD, but each time in a different parameterization. As a comparison the authors approximated the ABRDFs of synthetic BTFs using CMF and HF. In floating precision they reported similar approximation errors but the factors computed from CMF had a much smaller dynamic range and thus could be safely discretized into 8-bit textures used for rendering on graphics hardware. Furthermore, they stated CMF to be easier to compute and implement.

The compression ratio of per-texel matrix factorization depends on the size of the matrix X, i.e. the sampling of the angular space, and the number of factors. Please note, that these techniques where originally designed for BRDF rendering and for scenes containing a few (maybe some hundred) BRDFs. A single BTF with 256^2 texels contains 64k ABRDFs! Hence to reduce the memory requirements for real-time rendering a clustering of the factors may be necessary.

3.2.2. Full BTF-Matrix Factorization

The per-texel factorization methods from the previous section have the disadvantage, that they do not exploit inter-texel coherence. This can be accomplished by applying a PCA to the complete BTF-data arranged in a $|\mathcal{M}| \cdot |I|$ matrix

$$X = \left(B_{\mathbf{x}_0}, B_{\mathbf{x}_1}, \ldots, B_{\mathbf{x}_{|I|}}\right).$$

Keeping only the first c eigenvalues results in the following BTF-approximation:

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{j}^{c} g_j(\mathbf{x}) h_j(\mathbf{v},\mathbf{l}) \tag{11}$$

This approach was used in the works of Liu et al. [LHZ*04] and Koudelka et al. [KMBK03].

The remaining issue is, how to choose c. Ravi Ramamoorthi [Ram02] showed by an analytic PCA construction, that using about five components is sufficient to reconstruct lighting variability in images of a convex object with Lambertian reflectance. Therefore, for nearly diffuse and relatively flat samples a good reconstruction quality can be expected for low c. However, as illustrated in figure 12, this will not be sufficient for complex BTFs containing considerable self-shadowing, masking and obviously for non-diffuse reflectance. Therefore, Koudelka et al. chose $c = 150$ to represent all significant effects with enough fidelity. To reduce the size of the resulting dataset even further they stored the basis-vectors as JPG-images resulting in very high compression rates. But of course real-time reconstruction from this representation is not possible.

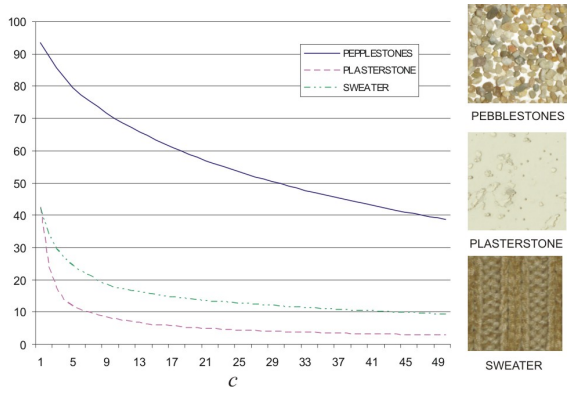

Figure 12: *RMS-error of the full BTF-matrix factorization depending on the the number of terms c. There is a direct correspondence between the magnitude and decrease of error and the BTF-complexity.*

An alternative approach for full BTF-matrix factorization was sketched by Vasilescu and Terzopoulos [VT03]. They arranged the BTF-data in a 3-mode tensor and applied multilinear analysis (3-mode SVD), which corresponds to the application of standard SVD to different arrangements of the data. The resulting reconstruction formula provides greater control of dimensionality reduction but a quality comparison to the standard linear analysis is not yet available.

A serious problem of the full BTF-matrix factorization methods is the size of the matrix X which easily could reach several gigabytes in float-precision. In this case, even the computation of the covariance matrix XX^T would be a very lengthy operation. Hence the method can be applied only to small samples or a subset of the sample.

3.2.3. Per-View Factorization

The full BTF-matrix factorization suffers from memory problems during computation and the reconstruction is only fast *and* correct for relatively simple materials. Sattler et al. [HEE*02, SSK03] published a method that deals with these problems. Because their original intention was to visualize cloth BTFs that exhibit a significant amount of depth variation and hence highly non-linear view-dependence, they use an approach similar to the work of Meseth et al. [MMK03a] in the sense that they applied PCA to slices of the BTF with fixed view-direction. This leads to the following set of data-matrices:

$$X_{\mathbf{v}_i} := \left(T_{(\mathbf{v}_i,\mathbf{l}_0)}, T_{(\mathbf{v}_i,\mathbf{l}_1)}, \ldots, T_{(\mathbf{v}_i,\mathbf{l}_{M_{\mathbf{v}_i}})}\right)$$

with $M_{\mathbf{v}_i}$ denoting the number of sampled light directions for the given view direction $\mathbf{v}_i$. The PCA is applied to all matrices $X_{\mathbf{v}_i}$ independently which poses no computational problems compared to the full BTF matrix. Then keeping only the first c eigenvalues gives the following BTF approximation:

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{j=1}^{c} g_{\mathbf{v},j}(\mathbf{l}) h_{\mathbf{v},j}(\mathbf{x}) \tag{12}$$

Compared to equation 11, the value of c can be set much lower (the authors chose c between 4 and 16) which enables interactive or even real-time rendering frame rates with good visual quality. However, the memory requirements are much higher. For example, $c = 16$ and $M_{\mathbf{v}_i} = 81$ lead to more than 1200 terms that have to be stored.

3.2.4. Per-Cluster Factorization

As already mentioned, a complex BTF contains highly non-linear effects like self-shadowing, self-occlusion and non-diffuse reflectance. Nevertheless, many high-dimensional data sets exhibit a local linear behavior. Applying per-texel or per-view factorization implicitly exploit this observation by selecting fixed subsets of the data and approximating these subsets with an affine-linear subspace. A more general approach would choose these subsets depending on the data. This is the idea behind the local PCA method, which was introduced by Kambhatla and Leen [KL97] to the machine-learning community in competition to classical non-linear/neural-network learning algorithms. It combines

clustering and PCA using the reconstruction error as metric for choosing the best cluster.

Recently, Mueller et al. [MMK03b] applied this method to BTF-compression and proposed the following approximation:

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{j}^{c} g_{k(\mathbf{x}),j}(\mathbf{x}) h_{k(\mathbf{x}),j}(\mathbf{v},\mathbf{l}) \qquad (13)$$

The operator $k(\mathbf{x})$ is the cluster index look-up given a position $\mathbf{x}$. In this case clustering is performed in the space of ABRDFs which was found being better suited for real-time rendering than clustering in the space of images. Now the number of clusters can be chosen according to computational resources and quality requirements. Figure 13 compares per-cluster factorization to full matrix factorization with the same number of terms c. Good results were obtained for cluster counts between 16 and 32, which is much smaller than the fixed cluster number (e.g. $M_{\mathbf{v}_i} = 81$) used in per-view factorization.

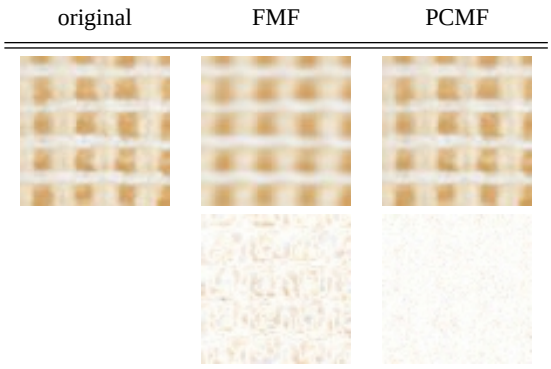

Figure 13: *Comparing full matrix factorization and per-cluster matrix factorization. From left to right: original frontal view of PROPOSTE BTF, reconstruction from full matrix factorization (FMF) with $c = 8$ terms, reconstruction from per-cluster matrix factorization (PCMF) with 32 clusters and 8 terms per cluster. Second row: enhanced and inverted difference images.*

3.3. Discussion

The following discussion is based on figure 14.

Obviously, the quality of the reconstruction from linear basis decompositions is better than from parametric reflectance models even if additional parameters like scaling values per view or even a full fit per view are used. Furthermore, the increased quality achieved by using this additional complexity does not legitimate the increased memory requirements. The qualitatively best result is achieved using per-view factorization but unfortunately the memory requirements are very high, since for every measured view direction a set of textures and weights has to be stored. Using per-texel matrix factorization does not exploit spatial coherence and thus the quality is not as good as per-view or per-cluster factorization even for considerable memory requirements. Furthermore, the chained resampling steps can introduce additional resampling error. Suykens et al. [SvBLD03] propose k-means clustering of the factors across the spatial dimension to reduce memory requirements. This obviously could be applied to every per-texel BTF compression method, but for complex materials that do not contain uniform areas this will introduce cluster artifacts. These cluster artifacts are reduced using PCA in each cluster as done in the per-cluster factorization method. Hence, this method seems to offer a good compromise between reconstruction cost, visual quality and memory requirements.

4. Synthesis

Current acquisition systems as presented in section 2 can capture only small material samples up to a few centimeters in extent. In practical applications such a sample should cover a much larger geometry, e.g. a fabric BTF covering a seat. The simplest way to accomplish this would be a simple tiling of the sample, but not all materials are suitable for tiling and typically the resulting repetitive patterns are not visually pleasing. A more sophisticated approach would generate a new larger texture that locally resembles, i.e. *looks like* the original texture but contains no obvious repetition artifacts.

Such an approach is called texture synthesis by example and has been subject of ongoing research for the last ten years.

4.1. 2D-Texture Synthesis by Example

The first attempts in using texture synthesis for computer graphics tried to approximate the underlying stochastic process and sample from it. For example, the methods of Heeger and Bergen [HB95] and DeBonet [De 97] are of this kind. Unfortunately, only a limited kind of textures could be reproduced adequately using this technique, due to the potential complexity of the stochastic process.

Therefore recently methods based on heuristic sampling became popular because they are faster and much simpler to implement. They do not try to model the global texture statistics but instead attempt to enforce the statistics locally for a single texel or small patch.

In practice, the conditional distribution for a texel or patch to synthesize given its already synthesized neighborhood is approximated by finding texels or patches with similar neighborhoods in the sample texture and choosing from this candidate set. This approach has been popularized by the work of Efros et al. [EL99] and Wei et al. [WL00].

These methods synthesize the new texture texel-by-texel and are well suited for textures with small and not too regular

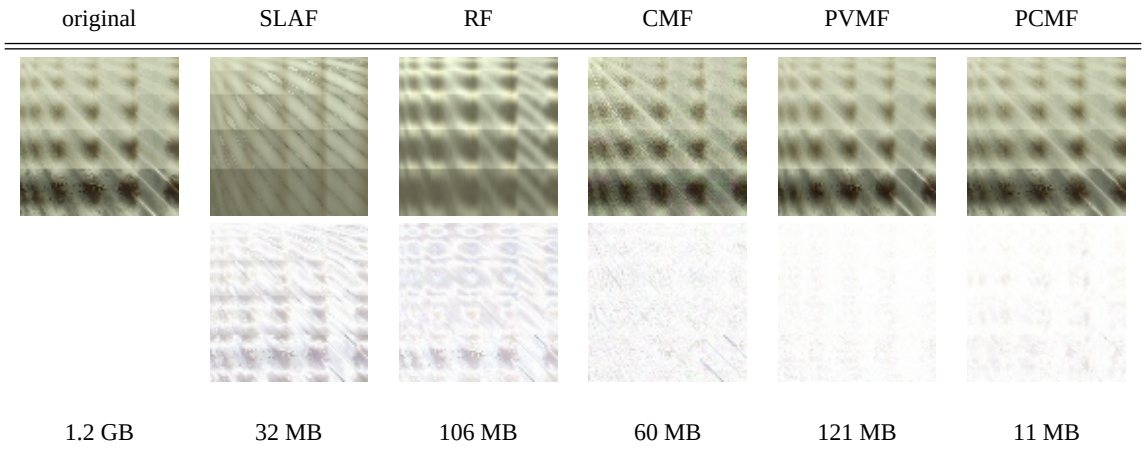

Figure 14: *Comparison of selected BTF-compression methods (see section 3 for details). Top row: original and reconstructed ABRDFs. Second row: inverted difference images. Bottom row: storage requirements for the compressed representations using parameters suitable for interactive rendering. From left to right: Original ABRDF of PLASTERSTONE BTF, Scaled Lafortune Lobes with 2 lobes (SLAF), Reflectance Fields with per-view polynomial (RF), Per-Texel Chained Matrix Factorization with 4 factors (CMF), Per-View Matrix Factorization with 8 terms (PVMF), Per-Cluster Matrix Factorization with 32 clusters and 8 terms (PCMF). The original dataset consists of 6561 8-bit RGB-images with* 256^2 *pixels in size. Numerical precision is 16 bit for PCMF and 8 bit for all other methods.*

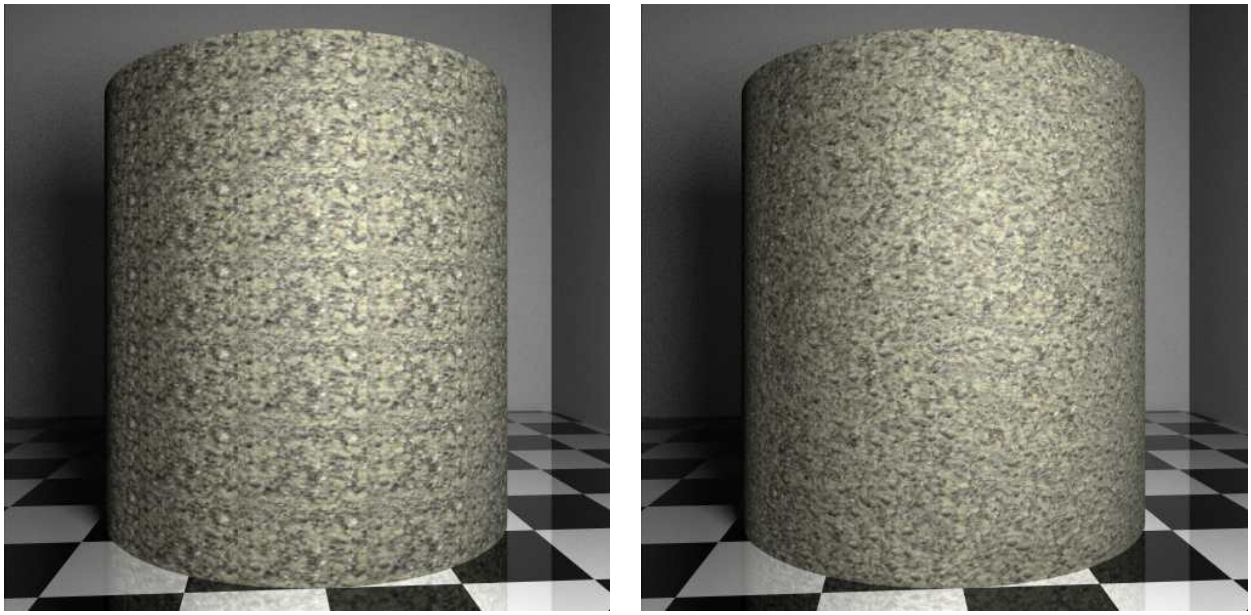

Figure 15: *In the left image the stone BTF is tiled resulting in visible repetitive patterns. On the right pixel-based synthesis was applied.*

features. A natural extension to this texel-based textures synthesis is patch-based texture synthesis which was introduced by Efros and Freeman [EF01] and copies texture patches at once into the synthesized texture while trying to minimize the visible seams between the patches.

The before mentioned methods work surprisingly well for many types of textures and can easily be applied to higher dimensional textures such as BTFs. In order to achieve feasible running times dimensionality reduction techniques such as Principal Component Analysis (PCA) have to be applied.

Figure 15 shows a result of applying pixel-based synthesis to a stone BTF whose dimension was reduced using PCA, in comparison to simple tiling.

4.2. Application to BTF-Synthesis

A first approach to BTF synthesis was made by Liu et al. [LYS01]. They recovered an approximate height field of the measured sample by shape-from-shading methods. After synthesizing a new height field with similar statistical properties using standard 2D texture synthesis methods, they render a gray image given a desired view and light direction. With the help of a reference image from the BTF measurement with the same view and light direction, they generate a new image by combining the rendered gray image and the reference image. A serious problem of the method is its inability to synthesize consistent views of the material.

A following publication of Tong et al. [TZL*02] targeted this drawback by synthesizing the whole BTF at once. To reduce the huge computational costs related to the high dimensionality of BTFs, they computed textons [LM99], i.e. texels with representative features obtained by clustering. Synthesis was afterwards done in a per-pixel manner based on a precomputated dot-product matrix which was used to efficiently compute distances between textons and linear combinations of textons.

In a most recent publication Koudelka et al. [KMBK03] applied the image quilting algorithm of Efros and Freeman [EF01] to dimension reduced BTFs.

5. Rendering

Generally accurate and physically plausible rendering algorithms have to compute a solution of the rendering equation at every surface point $\mathbf{x}$ (neglecting emissive effects):

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} \rho_{\mathbf{x}}(\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})(n_{\mathbf{x}}\cdot\mathbf{l})\, d\mathbf{l} \qquad (14)$$

Here, $\rho_{\mathbf{x}}$ denotes the BRDF, L_i denotes incoming radiance, $n_{\mathbf{x}}$ is the surface normal and Ω_i is the hemisphere over $\mathbf{x}$.

Including measured BTFs into the rendering equation 14 is very simple:

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} BTF(\mathbf{x},\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})(n_{\mathbf{x}}\cdot\mathbf{l})\, d\mathbf{l} \qquad (15)$$

Now the measured BRDF at point $\mathbf{x}$ is simply looked up from the BTF. Here we assume that a mapping from the 3D-surface to the 2D spatial texture domain exists already. Please note that the BTF also models meso-scale geometry. However, since this information is projected into the BTF the rendering will not be correct for example at object silhouettes.

5.1. Solving the Rendering Equation

Currently there are two popular approaches that are primarily used to solve the rendering equation.

5.1.1. Monte-Carlo Sampling

The first approach tries to solve equation 14 accurately using Monte-Carlo sampling methods. Many variants of this approach such as path tracing [Kaj86], distribution ray tracing [CPC84] and photon mapping [Jen96], to mention a few, have been developed over the years. Despite recent advances towards interactive global illumination (e.g. [WPS*03]) accurate solutions of arbitrary complex scenes can still take hours or even days to compute.

Obviuosly equation 15 can be solved using Monte-Carlo sampling as well. In this case the renderer simply has to be extended to support a particular BTF compression scheme which in fact corresponds to the implementation of the decompression stage on the CPU. This is possible for any compression method introduced in section 3. Figure 16 shows a path-traced image of a scene containing objects covered with compressed BTFs.

Figure 16: *Path-traced image of a max-planck bust covered with a LEATHER BTF compressed using per-cluster matrix factorization. The walls are covered with PLASTERSTONE and the floor with FLOOR-COVERING.*

5.1.2. Approximate Solutions for Real-Time Rendering

The second approach makes a priori several simplifying assumptions so that the integral can be solved more quickly and is amenable to hardware implementation. The goal is to reduce the integral in equation 14 to a finite sum containing only very few terms.

Point Lights The most popular simplification is using only a set $\Lambda = \{\mathbf{l}_j\}$ of point or directional light-sources and discarding general interreflections (i.e. the recursion in equation 14). For notational simplicity the term $\mathbf{l}_j$ encodes both intensity and direction or location of the light source. Since these lights are given in global coordinates and the BRDF is usually given in the local frame at $\mathbf{x}$ we also need the local coordinates $\tilde{\mathbf{l}}$:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_j^{|\Lambda|} \rho_{\mathbf{x}}(\mathbf{v},\tilde{\mathbf{l}}_j) G(\mathbf{x},\mathbf{l}_j)(n_{\mathbf{x}} \cdot \mathbf{l}_j)\mathbf{l}_j \qquad (16)$$

The geometry term $G(\mathbf{x},\mathbf{l}_j)$ contains the visibility function and an attenuation factor depending on the distance. For the rest of this STAR we will neglect the visibility function in the geometry term, because interactive large-scale shadowing is an independent research topic in its own right (for a survey refer to e.g. [HLHS03]). Using equation 16 a scene can be rendered real-time using todays consumer graphics hardware. Arbitrary BRDFs can be implemented using the programmable vertex- and fragment-shader capabilities. In the case of BTF-rendering the following sum has to be computed:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_j^{|\Lambda|} BTF(\mathbf{x},\mathbf{v},\tilde{\mathbf{l}}_j) G(\mathbf{x},\mathbf{l}_j)(n_{\mathbf{x}} \cdot \mathbf{l}_j)\mathbf{l}_j \qquad (17)$$

In fact, the challenge of developing a fast BTF-rendering algorithm for point lights is reduced to the implementation of a fragment program that evaluates the BTF for given parameters. For several of the compression schemes presented in section 3 such an implementation has been proposed and we will go into detail in section 5.4.

Infinitely Distant Illumination Another widely used simplification assumes infinitely distant incident illumination and no interreflections. In this case the dependency of incident lighting on surface position $\mathbf{x}$ can be removed:

$$L_r(\mathbf{x},\mathbf{v}) \approx \int_{\Omega_i} \rho_{\mathbf{x}}(\mathbf{v},\mathbf{l}) L_i(\mathbf{l})(n_{\mathbf{x}} \cdot \mathbf{l})\, d\mathbf{l} \qquad (18)$$

For special types of BRDFs this integral can be precomputed (e.g. [KVHS00]) and the results can be displayed in real-time. Another approach is to reduce the integral to a finite sum by projecting light and BRDF onto a basis like Spherical Harmonics (SH) [SKS02] or wavelets [NRH03] and keeping only a finite number of terms. Using this approach even transport effects like shadows and interreflections can be precomputed and projected onto the basis.

Special care has to be taken while transferring such an approach to BTF-rendering. The methods were originally designed only for special types of BRDFs or the results are only computed per vertex. Hence, only few algorithms for BTF-rendering using distant illumination have been published so far. The details will be discussed in section 5.5.

5.2. BTF-Rendering using Real-Time Raytracing

The recent advances in computation power and improved algorithms allow for interactive ray-tracing even on a single desktop PC [WPS*03]. Real-time performance can be achieved using PC clusters. As in section 5.1.1 the integration of a particular BTF compression scheme into such a system corresponds to the implementation of the decompression stage on the CPU. Figure 20 shows a car interior covered with measured BTFs and rendered by the OpenRT real-time ray-tracing engine [WBS02].

5.3. BTF-Rendering using Graphics Hardware

To incorporate BTFs into current real-time rendering systems, the evaluation of the BTF should be done on the graphics processing unit (GPU) i.e. integrated into the fragment-shading process. To achieve this, the compressed representation of the BTF must be stored in textures and the reconstruction is performed using the programmable units of the graphics board. The parameters for BTF evaluation are the standard 2D-texture coordinates $\mathbf{x}$, the local view direction $\mathbf{v}$ and in the case of point light sources the local light direction $\mathbf{l}$.

A crucial point in all BTF-rendering methods is interpolation. Since a measured BTF is discrete in all 6 dimensions, smooth renderings require interpolation in each dimension. To achieve high frame-rates it is indispensable, that at least some of this dimensions are interpolated using built-in hardware interpolation. For the other dimensions either the nearest neighbor must be chosen or the interpolation of the nearest neighbors has to be executed explicitly in the fragment shader. In both cases there has to be an operator $N(\cdot)$ that supplies the nearest samplings. Such a look up can be performed on the GPU using dependent texture look-ups. To perform explicit interpolation in the fragment shader, the corresponding interpolation weights τ_i should also be precomputed and stored in textures.

In the following sections we will present and discuss the existing BTF-rendering methods that achieve interactive or even real-time frame-rates exploiting the capabilities of current graphics hardware.

5.4. Interactive Rendering of BTFs with Point Lights

5.4.1. Parametric Reflectance Models

Efficient implementations of the parametric reflectance models from section 3.1 have been presented by various publications. McAllister et al. [MLH02] describe a real-time evaluation scheme for equation 3. Coefficients of the spatially varying Lafortune lobes are stored in 2D textures. Evaluation can efficiently be done using vertex and fragment shaders. Since Lafortune lobes are continuous in the angular domain, no interpolation is required in the angular domain. Spatial interpolation has to be done explicitly in the fragment shader

(magnification) or is left to the multisampling and built-in MIP-mapping capabilities of graphics hardware, although MIP-maps have to be built manually e.g. by fitting new sets of Lafortune lobes for each BTF resolution.

As an extension to the model of McAllister et al. the BTF model of Daubert et al. [DLHS01] only requires an additional evaluation of the view-dependent visibility term. Although significant numbers of parameters are required to store this term, it can easily be encoded in a texture. Interpolation of the view-direction can be achieved using graphics hardware. Spatial interpolation is done like in the previous approach.

The even more complex model of Meseth et al. [MMK03a] is evaluated very similarly to the previous two ones with the significant difference that the discretization of the view direction requires an additional manual interpolation (as denoted in equation 5). Therefore, two cube maps are utilized which store for each texel in the cube map (representing a certain view direction) the three closest view directions and respective interpolation weights. Interpolation is achieved by evaluating the reflectance fields for the three closest view directions and interpolating the results based on the interpolation factors stored in the cube map. Spatial interpolation can be done exactly like in the previous approaches.

Efficiently evaluating the BTF model of Filip and Haindl [FH04] constitutes an even more complex problem since it requires interpolation of both view and light direction, effectively requiring evaluation of the basic model nine times. Since the model was not intended for real-time rendering, no such algorithm was proposed yet and it is questionable if the improved rendering quality can compensate the significantly increased rendering costs.

5.4.2. Per-Texel Matrix Factorization

Generally, the rendering algorithms for BRDF-factorizations can be used [KM99, MAA01] with the difference, that the factors now change in every fragment. Suykens et al. [SvBLD03] detailed how this can be accomplished:

Every factor is reparameterized and stored into a parabolic map [Hei98]. Then all these factors are stacked into 3D-textures and normalized to a range between 0 and 1. The resulting scale factors are stored in a scale map. A particular factor is selected in its 3D-texture using transformed 2D texture coordinates or, if clustered factors are employed, by the values from an index map. To avoid mixing of neighboring factors in the 3D-texture the z-coordinate has to be chosen carefully. A value within a particular factor is indexed using the appropriately reparameterized local view and light directions. While interpolation inside a factor is supported by the hardware, spatial interpolation between neighboring texels is not implemented. Equation 10 now can be executed in a fragment shader.

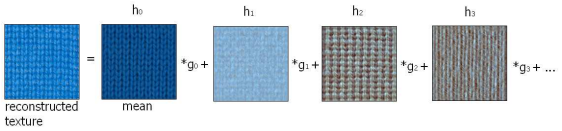

Figure 17: *Schematic rendering using the per-view matrix factorization.*

5.4.3. Per-View Matrix Factorization

Sattler et al. [SSK03] demonstrated how equation 12 can be implemented using a combination of CPU and GPU computations.

Figure 17 shows the basic approach: Combination of the eigen-textures $h_{\mathbf{v},j}(\mathbf{x})$ with the appropriate weights $g_{\mathbf{v},j}(\mathbf{l})$ using the multi-texturing capabilities of modern graphics boards. This combination is done for every triangle-vertex. A smooth transition is ensured by blending the resulting three textures over the triangle using a fragment program [CBCG02].

While the interpolation in the spatial domain is done by the graphics hardware, light and view direction are interpolated explicitly. To interpolate between the nearest measured light directions the term

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{\tilde{l}\in N(\mathbf{l})} \tau_{\tilde{l}} \sum_{j=1}^{c} g_{\mathbf{v},j}(\tilde{l}) h_{\mathbf{v},j}(\mathbf{x}) \qquad (19)$$

has to be evaluated. In order to speed up this computation the weights $\lambda_{\mathbf{v},j}(\mathbf{l}) = \sum_{\tilde{l}\in N(\mathbf{l})} \tau_{\tilde{l}} \cdot g_{\mathbf{v},j}(\tilde{l})$ are computed on the CPU per frame and sent to the GPU resulting in the term

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{j=1}^{c} \lambda_{\mathbf{v},j}(\mathbf{l}) h_{\mathbf{v},j}(\mathbf{x}). \qquad (20)$$

To perform view-interpolation different sets of eigen-textures have to be combined resulting in an expensive combination:

$$\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) \approx \sum_{\tilde{v}\in N(\mathbf{v})} \tau_{\tilde{v}} \sum_{j=1}^{c} \lambda_{\tilde{v},j}(\mathbf{l}) h_{\tilde{v},j}(\mathbf{x}). \qquad (21)$$

5.4.4. Full BTF-Matrix Factorization

To evaluate equation 11 the factors $g_j(\mathbf{x})$ and $h_j(\mathbf{v},\mathbf{l})$ have to be evaluated, whereas the factors $g_j(\mathbf{x})$ can be stored as simple 2D-textures and the factors $h_j(\mathbf{v},\mathbf{l})$ as 4D-textures. But unfortunately neither 4D-textures nor their full interpolation is currently supported by graphics hardware.

Therefore, Liu et al. [LHZ*04] store the factors $h_j(\mathbf{v},\mathbf{l})$ into stacks of 3D-textures. The tri-linear filtering capabilities of graphics hardware now can be exploited to interpolate the view direction and the polar angle of the light direction. The final step for 4D filtering is performed manually by blending two, tri-linearly filtered values with closest azimuth angles in

light direction. As usually, the values of $h_j(\mathbf{v},\mathbf{l})$ parameterized over the hemispheres of view and light directions have to be resampled and reparameterized in order to be stored in textures. In order to avoid the fragment shader's online effort of calculating the reparameterized local light and view directions, which are necessary for accessing the 3D-textures and interpolation weights, the mapping is precomputed and stored in a cube map.

5.4.5. Per-Cluster Matrix Factorization

Apart from the additional cluster look-up, evaluating equation 13 is essentially the same as evaluating 11. Hence the real-time rendering algorithm for equation 13 presented by Schneider [Sch04] is similar in style to the approach presented in the previous section. He also stores the factors $h_{k(\mathbf{x}),j}(\mathbf{v},\mathbf{l})$ in stacks of 3D-textures and accesses them through reparameterized local light and view directions. The cluster index introduces an additional dependent texture look-up. Since existing graphics boards only support hardware supported interpolation of fixed point values, every factor is quantized to 8-bit separately yielding scaling factors $s_{j,k}$. As compensation the factors $g_j(\mathbf{x})$, which can be stored as floating-point values since they require no interpolation, have to be divided by the corresponding scaling factor $s_{j,k}$.

Mipmapping the BTF can simply be implemented by executing the shader instructions twice (once for the currently best mipmap level and once for the next best mipmap level) and interpolating the resulting colors. Bilinear, spatial interpolation is currently not supported, since the additional overhead is prohibitive. Fortunately, hardware supported full-screen antialiasing can reduce potential artifacts significantly.

5.5. Interactive Rendering of BTFs with Distant Illumination

We consider relighting of BTFs with distant illumination according to equation 18. Typically this distant illumination is represented by an environment map. Such an environment map can either be computed by the graphics hardware or captured from a natural environment by taking pictures (e.g. of a metallic sphere) [DM97, HDR].

5.5.1. Parametric Reflectance Models

Combining parametric reflectance models for BTFs with image-based lighting relies on the concept of prefiltered environment maps, which was first applied to the diffuse reflection model by Miller and Hoffman [MH84] and Greene [Gre86]. For a diffuse BTF with spatially varying reflection coefficients ($\mathbf{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) = \rho_d(\mathbf{x})$), equation 18 reduces to:

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} \rho_d(\mathbf{x}) L_i(\mathbf{l})(\mathbf{n}_x \cdot \mathbf{l})\, d\mathbf{l}$$

$$= \rho_d(\mathbf{x}) \int_{\Omega_i} L_i(\mathbf{l})(\mathbf{n}_x \cdot \mathbf{l})\, d\mathbf{l}$$

$$= \rho_d(\mathbf{x}) D(\mathbf{n}_x). \quad (22)$$

The prefiltered environment map $D(\mathbf{x},\mathbf{n}_x)$ can be precomputed on the CPU, stored in a cube texture map and used during rendering. Kautz and McCool [KM00] extended the concept to isotropic BRDFs. Since for non-diffuse cases the prefiltered result becomes view-dependent, they approximated equation 18 as follows:

$$L_r(\mathbf{x},\mathbf{v}) \approx (\mathbf{n}_x \cdot p(\mathbf{x},\mathbf{v})) \int_{\Omega_i} BTF(\mathbf{x},\mathbf{v},\mathbf{l}) L_i(\mathbf{l})\, d\mathbf{l} \quad (23)$$

$$\approx (\mathbf{n}_x \cdot p(\mathbf{x},\mathbf{v}))\, S(\mathbf{x},\mathbf{v}) \quad (24)$$

where $p(\mathbf{x},\mathbf{v})$ denotes the peak direction, i.e. the light direction with maximum influence on $L_r(\mathbf{x},\mathbf{v})$. The accuracy of this approximation increases with the specularity of the spatially varying ABRDFs. With this assumption, the specular prefiltered environment $S(\mathbf{x},\mathbf{v})$ can be computed analogously to the diffuse case.

McAllister et al. [MLH02] applied this concept to equation 3. The diffuse and specular terms are considered individually resulting in two prefiltered environment maps: the diffuse $D_{\mathbf{x}}(\mathbf{n}_x)$ and a specular one, which is computed based on the ideas of Kautz and McCool [KM00]. Note that whenever writing $\mathbf{x}$ as index we refer to instances discretely sampled in the spatial domain. First, the peak direction

$$p_{\mathbf{x}}(\mathbf{v}) = \mathbf{v}^t \cdot \mathbf{D}_{\mathbf{x}}$$

of each lobe is computed. Then the specular illumination part (for ease of notation, a 1-lobe approximation is assumed) is rewritten as follows:

$$L_{r,s}(\mathbf{x},\mathbf{v}) \approx \int_{\Omega_i} \rho_{s,\mathbf{x}} \cdot (p_{\mathbf{x}}(\mathbf{v}) \cdot \mathbf{l})^{n_{\mathbf{x}}} L_i(\mathbf{l})(\mathbf{n}_x \cdot \mathbf{l})\, d\mathbf{l}$$

$$\approx \rho_{s,\mathbf{x}} \cdot (\mathbf{n}_x \cdot p_{\mathbf{x}}(\mathbf{v})) \int_{\Omega_i} (p_{\mathbf{x}}(\mathbf{v}) \cdot \mathbf{l})^{n_{\mathbf{x}}} L_i(\mathbf{l})\, d\mathbf{l}$$

$$\approx \rho_{s,\mathbf{x}} \cdot (\mathbf{n}_x \cdot p_{\mathbf{x}}(\mathbf{v}))\, ||p_{\mathbf{x}}(\mathbf{v})||^{n_{\mathbf{x}}} S(p_{\mathbf{x}}(\mathbf{v}), n_{\mathbf{x}}) \quad (25)$$

with

$$S(p_{\mathbf{x}}(\mathbf{v}), n_{\mathbf{x}}) = \int_{\Omega_i} \left(\frac{p_{\mathbf{x}}(\mathbf{v})}{||p_{\mathbf{x}}(\mathbf{v})||} \cdot \mathbf{l}\right)^{n_{\mathbf{x}}} L_i(\mathbf{l})\, d\mathbf{l} \quad (26)$$

$S(\mathbf{p},n)$ denotes the specular prefiltered environment map.

Evaluating approximation 25 can efficiently be done using fragment shaders by first computing $p_{\mathbf{x}}(\mathbf{v})$, looking up the respective specular prefiltered value from a cube map, evaluating the specular part and adding the diffuse contribution. Special care has to be taken only concerning the specular exponent, which is represented as discrete versions in

$S(\mathbf{p},n)$ only. One can either choose the closest exponent or interpolate from the two closest ones.

Like for point-like sources, the continuity of the Lafortune lobes in the angular domain requires additional interpolation in the spatial domain only. Again, graphics hardware features like multisampling and MIP-mapping are employed for this task.

Whereas the extension of this approach to the model of Daubert et al. [DLHS01] requires an additional evaluation of the visibility term only, extending it to the reflectance field based model of Meseth et al. [MMK04a] requires interpolation from the results of the reflectance fields corresponding to the closest measured view directions. Nevertheless, all three approaches allow for real-time rendering.

5.5.2. Bi-Scale Radiance Transfer

Precomputed Radiance Transfer (PRT), as originally introduced by Sloan et al. [SKS02], is evaluated per vertex only and interpolated across the triangle. In a follow up work, Sloan et al. [SLSS03] extended PRT to support also spatially varying reflectance across the triangle. They projected the BTF per pixel and fixed view direction onto the SH basis generating a Radiance Transfer Texture (RTT) which now represents the per-view response of the material to the spherical harmonics basis. To cover large geometry with the memory intensive texture they used the synthesis algorithm of Tong et al. [TZL*02] to generate an ID map over the mesh which references into the RTT.

Generation of the RTT and the ID Map The generation of the RTT $B(\mathbf{x},\mathbf{v})$ is accomplished by projecting the BTF onto the first c elements of the SH basis $\{Y_j\}_{j\in\mathbf{N}}$:

$$B(\mathbf{x},\mathbf{v})_j = \int_{\Omega_i} BTF(\mathbf{x},\mathbf{v},\mathbf{l})Y_j(\mathbf{l})\,d\mathbf{l} \qquad (27)$$

Thus, the RTT is a 4D-array of c Spherical Harmonics coefficients (typically $c = 25$).

The ID Map is generated using BTF synthesis over densely resampled geometry called a *meso-mesh*. This step assigns every vertex an ID into the RTT. Then a texture atlas for the original coarse mesh is generated and for every texel in this atlas the nearest vertex in the meso-mesh is found and the corresponding ID is assigned to the texel.

Real-Time Rendering Standard PRT-rendering is performed per-vertex. That means computation of a matrix-vector multiplication between the transfer matrix and the incident lighting SH-coefficients. The resulting transferred lighting vector is interpolated across the triangle. To compute exitant radiance per-fragment the interpolated transferred lighting vector is dot multiplicated with the corresponding vector in the RTT which is accessed by the ID map and the local view direction. This is done in a fragment program. Each texel of the RTT is stored in an 8x8 texture block

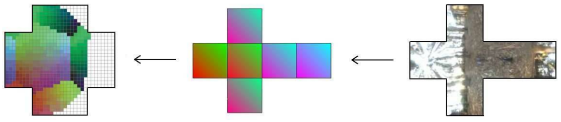

Figure 18: *Hemicube computation. Visibility map (left) with rendered color-coded lookup environment map (middle). White color in the visibility map stands for occlusion caused by the mesh. On the right side a HDR environment is shown, which is mapped onto the color-coded one.*

encoding the view dependence. This allows smooth bilinear interpolation across views using built-in hardware interpolation. Interpolation between spatial samples is not performed.

Since the RTTs are not compressed, the method supports only sparse samplings. They used 8x8 view-samples and 64x64 spatial samples which results in $64^3 * 25$ SH-coefficients that have to be stored per color band.

5.5.3. Per-View Matrix Factorization

Sattler et al. [SSK03] also proposed a method to relight the per-view factorized BTFs could by environment maps. The main idea is to discretize the integral in equation 18 using a hemicube [CG85], which leads to

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{\hat{l}\in H_i} \rho_{\mathbf{x}}(\mathbf{v},\hat{l})H_{i,\mathbf{x}}(\hat{l})(n_{\mathbf{x}}\cdot\hat{l}) \qquad (28)$$

where $H_{i,\mathbf{x}}$ denotes the discretized hemicube. $H_{i,\mathbf{x}}(\hat{l})$ returns the color in the hemicube over $\mathbf{x}$ at direction $\hat{l}$ or zero if the direction is occluded. The hemicube is precomputed and stored in a visibility map as follows:

Hemicube Precomputation The hemicube $H_{i,\mathbf{x}}$ stores a discretization of the hemisphere at the vertex $\mathbf{x}$. Figure 18 (left) shows an unfolded hemicube. Using a color-coded environment map (figure 18 middle) a look-up table into a high dynamic range map (figure 18 right) is created. This allows easy exchange of the environment map. By rendering the geometry in white color into the hemicube the visibility function is computed and self-shadowing can be supported. Since the directions in the visibility map not necessarily correspond to the measured light directions in the BTF, the map furthermore stores the four nearest measured light directions and the corresponding interpolation weights.

A hemicube pixel now stores the following information:

- visibility of a pixel of the environment map and if it is visible, the position of this pixel in the map
- four nearest measured directions with respect to the direction represented by this pixel
- corresponding interpolation weights

Rendering algorithm Given the nearest measured view direction $\mathbf{v}$ at vertex $\mathbf{x}$ and substituting the BRDF in equation 28 by the per-view factored BTF representation including the foreshortening term yields the following sum:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{\mathbf{l}\in H_i} \left(\sum_{j=1}^{c} \lambda_{\mathbf{v},j}(\mathbf{l}) h_{\mathbf{v},j}(\mathbf{x}) \right) H_{i,\mathbf{x}}(\mathbf{l}) \qquad (29)$$

The factors $\lambda_{\mathbf{v},j}(\mathbf{l})$ are from equation 20. As in equation 19 the factors $\gamma_{\mathbf{v},j} = \sum_{\mathbf{l}\in H_i} \lambda_{\mathbf{v},j}(\mathbf{l}) H_{i,\mathbf{x}}(\mathbf{l})$ are precomputed per vertex and sent to the GPU where the final expression is evaluated:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=1}^{c} \gamma_{\mathbf{v},j} h_{\mathbf{v},j}(\mathbf{x}) \qquad (30)$$

which again is only a linear combination of basis textures. For view interpolation the same calculations as in equation 21 have to be applied.

Since $\gamma_{\mathbf{v},j}$ is computed for all vertices $\mathbf{x}$ of the geometry a vector U holding all $\gamma_{\mathbf{x},\mathbf{v},\mathbf{j}}$ can be introduced:

$$U = (\gamma_{\mathbf{x}_0,\mathbf{v}_0,1}, \ldots, \gamma_{\mathbf{x}_0,\mathbf{v}_0,c}, \ldots) \qquad (31)$$

This vector has do be calculated once per environment map and allows real-time changes of the viewing position. A drawback of this method is, that changing the environment map and even a simple rotation implies a complete recalculation of U. Depending on the visibility map resolution and the number of vertices, this computation can take too long for interactive change of lighting. Reducing the visibility map resolution adaptively to achieve interactive changing rates introduces under-sampling artifacts of the environment map during motion, which can be compensated if the change stops, by using an adaptively higher resolution for the visibility map.

5.5.4. Per-Cluster Matrix Factorization

In section 5.5.2 the Bi-Scale Radiance Transfer method for image-based lighting of models covered with uncompressed BTFs was reviewed. To remove the limitation on the BTF resolution – both in the angular and spatial domain – Müller et al. [MMK04b] presented a combination of local PCA compressed BTFs with image based lighting.

Data Representation Similar to Sloan et al. [SLSS03] they compute Radiance Transfer Textures but encode them using the local PCA method [KL97]. In addition, similar to Sloan et al. [SHHS03], they apply local PCA to the transfer matrices of the mesh vertices. An approximate solution to equation 15 can now be computed by the weighted sum of dot products between the PCA-factors of the RTT and the transfer matrices.

Real-Time Rendering Since the dot products remain constant as long as only the camera is moving in the scene (i.e. neither the mesh nor the environment nor the BTF is changing), they can efficiently be precomputed on the CPU and afterwards be stored in a texture. Since precomputation and upload times of the dot products do not allow interactive rendering for very high quality settings, the number of RTT and transfer matrix components is reduced in dynamic situations. These reductions in quality do not significantly influence the perceived quality of the rendered results as long as high quality solutions are presented in static cases.

Like in Sloan et al. [SHHS03], rendering requires clusters of triangles to be rendered independently, where a triangle belongs to a certain cluster if at least one of its vertices belongs to the respective clusters. This increases the rendering time slightly but the overhead is negligible for smooth meshes.

Most rendering power is spent in the fragment program which computes the weighted sum. Interpolation of the angular domain of the BTF can be achieved by standard filtering features, spatial interpolation and filtering can be achieved using standard multisampling and MIP-mapping of the BTF.

5.6. Discussion

To our experience almost all real-time BTF rendering methods pose a great challenge to the current graphics boards and the performance differences vary greatly depending on the board and driver versions. Therefore, a rigorous comparison of the different rendering methods using for example frame-rates seems currently not possible and will be out of the scope of this STAR. Instead, we will give some general "hardware-indepent" properties of the algorithms which might help judging the strengths and weaknesses of the methods.

The method of McAllister et al. [MLH02] is mainly suited for nearly flat materials with complex and specular spatially varying reflection properties. Since it only uses slightly more memory than ordinary diffuse textures and can be rendered fast, it fits into current rendering systems. This is not true for the methods which use additional data to capture also depth-varying BTFs such as [DLHS01, MMK03a, FH04]. Their memory consumption and increased rendering cost restricts them to special domains and as pointed out in section 3 the visual quality of the used analytical models still remains questionable.

Far better visual quality is offered by the methods based on matrix factorization. But using PCA alone as done by Liu et al. [LHZ*04] is only real-time for very few terms, and thus only applicable for simple materials. Therefore, a segmentation of the data into subsets may be necessary. Using a fixed segmentation into reflectance fields as done by Sattler et al. [SSK03] provides excellent visual quality but requires too much texture memory to be used in complex scenes with many materials. Spatial clustering as in the method of Mueller et al. [MMK03b] or [SvBLD03] reduces the memory requirements drastically while retaining high-quality but

these methods face the not sufficiently solved problem of spatial interpolation and Mip-Mapping. This can result in decreased quality for texture magnification and minification. A problem of all matrix factorization based methods is the required random access to many textures, which can form a bottleneck on current graphics architectures.

6. Applications

Complex materials are part of everyone's day-to-day life. Since only few of them can adequately be modelled by a single BRDF, the applicability of BTFs in computer graphics related areas like entertainment and movie industry or industrial prototyping is very wide, although the requirements on BTF rendering vary drastically. In the following two subsections, application areas of realistic and predictive BTF rendering will be presented, special requirements and suitable existing or not yet discovered algorithms will be identified.

6.1. Realistic BTF Rendering

Although the word 'realistic' is known to everyone, very different understandings of the meaning exist. By 'realistic BTF rendering' we refer to rendering methods based on BTFs that produce believable images, ones that are very likely to be realistic. In this sense the term 'realistic rendering' is contrary to 'predictive rendering', which aims at producing images that reflect the result observed in reality. In the context of BTFs the degree of correctness of the prediction is evaluated e.g. based on the difference between the visualization of measured materials and the appearance of their actual counterparts from reality.

In accordance with this definition, realistic BTF rendering is suitable for applications that produce purely virtual results like the game or movie industry whose products are not expected to be ever seen but using a projection device.

Today, computer games feature elaborate graphics leading to astonishing effects already which is expected to improve even further in the future due to the continuous growth of computation power of graphics processing units (GPUs). Since the strictest requirement in computer games is real-time performance, games are just starting to employ true BRDF rendering due to the increased rendering efforts compared to standard texturing combined with simple, hardware accelerated light models. Nevertheless, upcoming games like 'Universal Combat: Edge to Edge' by 3000AD, Inc., will profit from enhanced rendering quality and realism and are very likely to achieve better sales records due to this feature. Obviously, replacing BRDF rendering by BTF rendering will increase image quality even further and therefore it appears to be just a question of time until the release of the first game utilizing BTF rendering.

Despite the ever-growing capabilities of CPUs and especially GPUs, BTF rendering algorithms for computer games need to fulfill two criteria: first and most important they need to be simple to achieve real-time frame rates. Second, the compression rate of the BTF model needs to be high since games typically utilize lots of materials simultaneously. Therefore, BTF rendering methods like the ones of McAllister et al. [MLH02] or intermediate methods between bump-mapping and BTF-rendering like the Polynomial Texture Maps [MGW01] or the approach of Kautz et al. [KSS*04] might be the first to be utilized at least for some kinds of materials for which their approximation quality turns out to be sufficient.

The movie industry, in contrast to game industry, is much more focused on high-quality images and less concerned about rendering time. Nevertheless, since movies often feature visually rich scenes of huge extent with a very large number of actors or objects, special requirements are put on the memory consumption of material representations. Therefore, rendering algorithms based on representations offering high compression rates, high quality reconstructions and easy adjustability of quality like the one of Müller et al. [MMK03b] will most likely be included in future movie production tools.

Unfortunately, existing movies do not feature BTF rendering yet, most probably due to missing acquisition systems built for industrial use and the lack of companies offering BTF acquisition or modelling services. An approximate impression of the realism achieved by employing BTFs is conveyed by movies featuring BRDF rendering, which has become a very common feature in special effects for movies. For the movie 'Matrix Reloaded' even measured BRDFs have been used [Bor03].

Examples of BTF applications targeting realistic rendering e.g. for movie production include rendering of trees with correct shading and shadowing [MNP01], rendering of feathers and birds [CXGS02] or textiles [DLHS01, SSK03, KSS*04]. Figure 19 shows an example of visualizing cloths in a HDR environment

6.2. Predictive BTF Rendering

In contrast to the above realistic rendering approach, the term predictive rendering denotes a much more difficult task: the results need not only be believable and possible, but they need to be closely related to a tangible counterpart. A prominent example application area of predictive BTF rendering is product design: designers want to be able to judge the appearance of the developed object already in early design phases before a real prototype was built. Renderings produced for these kinds of users need to faithfully predict the real appearance of the object in order to be useful.

Industrial products aiming in this direction already exist on the market. Maya from Alias System offers a combination of modelling tools and sophisticated rendering which creates highly realistic images for materials like plastic. Delcam plc

Figure 19: *Photorealistic textile visualization. The fabrics were rendered from measured BTFs and lit by the environment*

offers a complete CAD solution including the possibility to render images using a material library including highly reflective, transparent and bumpy materials. Lumiscaphe offers a software solution called Patchwork 3D for creating images from CAD models which are textured by the user using a provided selection of materials that approximate the BTF of real materials.

To fully achieve predictive results, many other requirements apart from correct material representations need to be met. In Klein et al. [KMM*03] an acquisition and rendering pipeline for realistic reflectance data in Virtual Reality (VR) is described. Among the necessary stages leading to predictive rendering are material and light source acquisition, texturing (since materials need to be applied to surface models as in reality), realistic lighting simulation including global illumination, realistic rendering and real-time tone mapping. The paper describes existing partial solutions but concludes that many issues are still to be solved to achieve the final goal. As an example, figure 20 shows a high-quality visualization of the interior of a Mercedes C class including measured BTFs. Although the image looks highly realistic in some parts, inadequate texturing, missing color and tone mapping and artificial point-light illumination make it appear far from similar to the real interior.

In terms of existing BTF rendering algorithms a comparison and evaluation of existing algorithms with respect to suitability for VR applications was published by Meseth et al. [MMSK03]. Unfortunately, due to the number of recent publications in this area the comparison is far from complete. Another problem is that the paper measures rendering quality as simple RGB color differences which does not comply with the human judgement of similarity. Due to the large number of factors influencing apparent color similarity and their complex dependencies (which are studied intensively by color scientists), more accurate psychophysical evaluations of such kind require time-intense experiments. Such experiments – although targeting only comparable questions – were e.g. conducted by McNamara et al. [MCTG00], Drago and Myszkowski [DM01] and Drago et al. [DMMS02].

Figure 20: *Visualization of the interior of a Mercedes C class using measured BTFs. This image was rendered using the OpenRT real-time ray-tracing engine [WBS02].*

To conclude, currently BTF rendering can improve the quality of rendered images significantly, representing an important yet not by itself sufficient steps towards predictive image generation. Despite the fast advances in computer graphics we do not expect achieving sufficient quality for predictive rendering in the nearest future due to missing methods, evaluation tests and integrated products. Nevertheless, the target does not seem to be fatuous and therefore will definitely be tackled in coming years due to the immense industrial needs.

7. Summary and Conclusion

In this STAR, we have presented research aiming at using BTFs as a generalized, image-based material representation for rendering. We have demonstrated that important steps towards this goal have already been made:

- The acquisition of BTFs using gonioreflectometer-like setups is reliable and results are now publicly available. But since the measurement times for one material lie in the range of several hours, BTFs are still a rare and expensive resource. Therefore, the development of fast and cheap methods for BTF measurement will be a very interesting field for future research.
- Numerous methods for BTF-compression are available. Depending on the desired application area, almost every method has its right to exist. Methods based on parametric reflectance models are suited for real-time rendering in large and complex scenes since they can be rendered fast and have minor memory requirements. For high-quality demands, methods based on more general compression techniques like linear basis decomposition should be used. Although even in this case real-time rendering is already possible, *the* optimal BTF-model for rendering has not been found yet.
- Synthesis of BTFs can be accomplished by applying well-known algorithms for 2D-texture synthesis.
- High-quality BTF rendering even with complex image-

based illumination is now possible in real-time. A problem remaining for future work is the fast interpolation of all six BTF-dimensions on graphics hardware.

Finally, we gave an overview of current and future application areas for BTFs. There is no doubt that simple material representations like 2D-texture or bump-maps sooner or later will be replaced by more complex representations that capture all the subtle effects of general light-material interaction. This STAR has shown, that the BTF has the potential be such a representation.

8. Acknowledgments

This work was partially funded by the European Union under the project RealReflect (IST-2001-34744). Some of the used high-dynamic range images were taken from Paul Debevecs webpage [HDR]. The textiles shown in this work were generated in scope of the BMBF project "Virtual Try-On" with the cloth simulation engine TüTex and provided by M. Wacker, M. Keckeisen, and S. Kimmerle from the WSI/GRIS at the University of Tübingen. For further details we refer to [EKS03, KKMW03, MKE03]. The car model was kindly provided by DaimlerChrysler. Special thanks belong to Roland Wahl, Marcin Novotni and Ferenc Kahlesz for proof reading.

References

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *Computer Graphics 11*, 3 (1977), 192–198. 72, 76

[Bor03] BORSHUKOV G.: Measured BRDF in Film Production - Realistic Cloth Appearance for 'The Matrix Reloaded'. In *SIGGRAPH 2003 Sketches* (2003). 88

[BTF] BTF DATABASE BONN: http://btf.cs.uni-bonn.de. 73

[CBCG02] CHEN W.-C., BOUGUET J.-Y., CHU M. H., GRZESZCZUK R.: Light field mapping: Efficient representation and hardware rendering of surface light fields. *Proceedings of ACM SIGGRAPH 2002* (2002). 84

[CG85] COHEN M. F., GREENBERG D. P.: The hemicube: A radiosity solution for complex environments. *Symposium on Computational Geometry 19*, 3 (July 1985), 31–40. 86

[Col] COLUMBIA-UTRECHT REFLECTANCE AND TEXTURE DATABASE: http://www1.cs.columbia.edu/ /cave/curet/. 73

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), ACM Press, pp. 137–145. 82

[CXGS02] CHEN Y., XU Y., GUO B., SHUM H.-Y.: Modeling and rendering of realistic feathers. In *Proceedings of the 29th annual Conference on Computer Graphics and Interactive Techniques* (2002), ACM Press, pp. 630–636. 88

[De 97] DE BONET J. S.: Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 361–368. 80

[DHT*00] DEBEVEC P., HAWKINS T., TCHOU C., DUIKER H.-P., SAROKIN W., SAGAR M.: Acquiring the reflectance field of a human face. *Proceedings of ACM SIGGRAPH 2000* (2000). 72

[DLHS01] DAUBERT K., LENSCH H., HEIDRICH W., SEIDEL H.-P.: Efficient cloth modeling and rendering. In *12th Eurographics Workshop on Rendering* (2001), pp. 63–70. 77, 84, 86, 87, 88

[DM97] DEBEVEC P. E., MALIK J.: Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), ACM SIGGRAPH, Addison Wesley, pp. 369–378. ISBN 0-89791-896-7. 85

[DM01] DRAGO F., MYSZKOWSKI K.: Validation Proposal for Global Illumination and Rendering Techniques. *Computers and Graphics 25*, 3 (2001), 511–518. 89

[DMMS02] DRAGO F., MARTENS W., MYSZKOWSKI K., SEIDEL H.-P.: *Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference*. Research Report MPI-I-2002-4-002, Max-Planck-Institut für Informatik, August 2002. 89

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (1999), 1–34. 69, 73

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 341–346. 81, 82

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. *Proceedings of Pacific Graphics* (2003). 91

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2* (1999), IEEE Computer Society, p. 1033. 80

[FH04] FILIP J., HAINDL M.: Non-linear reflectance model for bidirectional texture function synthesis. In *ICPR 2004* (2004). 77, 84, 87

[FKIS02] FURUKAWA R., KAWASAKI H., IKEUCHI K., SAKAUCHI M.: Appearance based object modeling using texture database: acquisition, compression and rendering. In *Proceedings of the 13th Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 257–266. 73

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. *Computer Graphics 30*, Annual Conference Series (1996), 43–54. 72

[GLL*04] GOESELE M., LENSCH H. P., LANG J., FUCHS C., SEIDEL H.-P.: Disco-acquisition of translucent objects. In *Proceedings of SIGGRAPH 2004* (August 2004), pp. 666–666. 71

[Gre86] GREENE N.: Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications 6*, 11 (1986), 21–29. 85

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 229–238. 80

[HDR] HDR: http://www.debevec.org/probes/. 85, 91

[HEE*02] HAUTH M., ETZMUSS O., EBERHARDT B., KLEIN R., SARLETTE R., SATTLER M., DAUBER K., KAUTZ J.: Cloth Animation and Rendering. In *Eurographics 2002 Tutorials* (2002). 73, 79

[Hei98] HEIDRICH W.: View-independent environment maps. In *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware '98* (1998). 84

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. In *Eurographics* (2003), Eurographics, Eurographics. State-of-the-Art Report. 83

[HP03] HAN J. Y., PERLIN K.: Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Trans. Graph. 22*, 3 (2003), 741–748. 75

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96* (1996), Springer-Verlag, pp. 21–30. 82

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001* (August 2001), pp. 511–518. 71

[Jol86] JOLLIFFE I.: *Principal Component Analysis*. Springer-Verlag, 1986. 78

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986), ACM Press, pp. 143–150. 82

[KKMW03] KIMMERLE S., KECKEISEN M., MEZGER J., WACKER M.: TüTex: A Cloth Modelling System for Virtual Humans. In *Proceedings 3D Modelling* (2003). 91

[KL97] KAMBHATLA N., LEEN T. K.: Dimension reduction by local principal component analysis. *Neural Comput. 9*, 7 (1997), 1493–1516. 79, 87

[KM99] KAUTZ J., MCCOOL M.: Interactive Rendering with Arbitrary BRDFs using Separable Approximations. In *Tenth Eurographics Workshop on Rendering* (1999), pp. 281–292. 78, 84

[KM00] KAUTZ J., MCCOOL M. D.: Approximation of Glossy Reflection with Prefiltered Environment Maps. In *Graphics Interface 2000* (2000), pp. 119–126. 85

[KMBK03] KOUDELKA M. L., MAGDA S., BELHUMEUR P. N., KRIEGMAN D. J.: Acquisition, compression and synthesis of bidirectional texture functions. In *3rd International Workshop on Texture Analysis and Synthesis* (2003). 75, 79, 82

[KMM*03] KLEIN R., MESETH J., MÜLLER G., SARLETTE R., GUTHE M., BALÁZS Á.: Realreflect: Real-time Visualization of Complex Reflectance Behaviour in Virtual Prototyping. In *Proceedings of Eurographics 2003 Industrial and Project Presentations* (2003). 89

[KSS*04] KAUTZ J., SATTLER M., SARLETTE R., KLEIN R., SEIDEL H.-P.: Decoupling BRDFs from surface mesostructures. To appear in proceedings of Graphics Interface 2004, 2004. 88

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: A Unified Approach to Prefiltered Environment Maps. In *11th Eurographics Workshop on Rendering* (2000), pp. 185–196. 83

[Lew93] LEWIS R. R.: Making shaders more physically plausible. In *Fourth Eurographics Workshop on Rendering* (Paris, France, 1993), pp. 47–62. 76

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 117–126. 76

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. *Computer Graphics 30*, Annual Conference Series (1996), 31–42. 69, 72

[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 10*, 3 (2004), 278–289. 79, 84, 87

[LKG*01] LENSCH H. P., KAUTZ J., GOESELE M., HEIDRICH2 W., SEIDEL H.-P.: Image-based reconstruction of spatially varying materials. *Proceedings of Eurographics Rendering Workshop 01* (2001). 73

[LM99] LEUNG T., MALIK J.: Recognizing surfaces using three-dimensional textons. In *Proceedings of the International Conference on Computer Vision-Volume 2* (1999), IEEE Computer Society, p. 1010. 82

[LYS01] LIU X., YU Y., SHUM H.-Y.: Synthesizing bidirectional texture functions for real-world surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 97–106. 82

[MAA01] McCool M. D., Ang J., Ahmad A.: Homomorphic Factorization of BRDFs for High-Performance Rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 171–178. 78, 84

[MCTG00] McNamara A., Chalmers A., Troscianko T., Gilchrist I.: Comparing real & synthetic scenes using human judgements of lightness. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (2000), Springer-Verlag, pp. 207–218. 89

[MGW01] Malzbender T., Gelb D., Wolters H.: Polynomial texture maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 519–528. 72, 75, 77, 88

[MH84] Miller G. S., Hoffman C. R.: Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments. In *SIGGRAPH '84 Advanced Computer Graphics Animation seminar notes* (1984). 85

[MKE03] Mezger J., Kimmerle S., Etzmuss O.: Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG 11*, 2 (2003), 322–329. 91

[MLH02] McAllister D., Lastra A., Heidrich W.: Efficient rendering of spatial bi-directional reflectance distribution functions. *Graphics Hardware 2002, Eurographics / SIGGRAPH Workshop Proceedings* (2002). 73, 76, 77, 83, 85, 87, 88

[MMK03a] Meseth J., Müller G., Klein R.: Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium 2003* (April 2003), Eurographics Association, Switzerland, pp. 89–96. 77, 79, 84, 87

[MMK03b] Müller G., Meseth J., Klein R.: Compression and real-time Rendering of Measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (November 2003), pp. 271–280. 80, 87, 88

[MMK04a] Meseth J., Müller G., Klein R.: Reflectance field based real-time, high-quality rendering of bidirectional texture functions. *Computers and Graphics 28*, 1 (February 2004), 103–112. 77, 86

[MMK04b] Müller G., Meseth J., Klein R.: Fast Environmental Lighting for Local-PCA Encoded BTFs. In *Computer Graphics International 2004 (CGI2004)* (June 2004). 87

[MMSK03] Meseth J., Müller G., Sattler M., Klein R.: Btf rendering for virtual environments. In *Virtual Concepts 2003* (November 2003), pp. 356–363. 89

[MNP01] Meyer A., Neyret F., Poulin P.: Interactive Rendering of Trees with Shading and Shadows. In *12th Eurographics Workshop on Rendering* (July 2001). 88

[MPBM03a] Matusik W., Pfister H., Brand M., McMillan L.: A data-driven reflectance model. *ACM Trans. Graph. 22*, 3 (2003), 759–769. 78

[MPBM03b] Matusik W., Pfister H., Brand M., McMillan L.: Efficient isotropic BRDF measurement. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), pp. 241–247. 72

[MPDW03] Masselus V., Peers P., Dutré P., Willems Y. D.: Relighting with 4d incident light fields. *ACM Trans. Graph. 22*, 3 (2003), 613–620. 72

[MPN*02] Matusik W., Pfister H., Ngan A., Beardsley P., Ziegler R., McMillan L.: Image-based 3d photography using opacity hulls. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 427–437. 72, 78

[MRP98] Miller G. S. P., Rubin S. M., Ponceleon D.: Lazy decompression of surface light fields for precomputed global illumination. In *Proceedings of the 9th Eurographics Workshop on Rendering* (1998), pp. 281–292. 72

[MWL*99] Marschner S. R., Westin S. H., Lafortune E. P. F., Torrance K. E., Greenberg D. P.: Image-based BRDF measurement including human skin. In *Proceedings of 10th Eurographics Workshop on Rendering* (1999), pp. 139–152. 72

[NRH*77] Nicodemus F., Richmond J., Hsia J., Ginsberg I., Limperis T.: Geometric considerations and nomenclature for reflectance. *Monograph 160* (October 1977). 71

[NRH03] Ng R., Ramamoorthi R., Hanrahan P.: All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics 22*, 3 (2003), 376–381. 83

[NSI99] Nishino K., Sato Y., Ikeuchi K.: Eigen-texture method: appearance compression based on 3d model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)* (June 1999), vol. 1, pp. 618 – 624. 78

[PFTV92] Press W. H., Flannery B. P., Teukolsky S. A., Vetterling W. T.: *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. 76

[Pho75] Phong B. T.: Illumination for computer generated pictures. *Communications of the ACM 18*, 6 (1975), 311–317. 76

[PTVF92] Press W., Teukolsky S., Vetterling W., Flannery B.: *Numerical recipes in C - The art of scientific computation*, 2nd ed. ISBN 0-521-43108-5. Cambridge University Press, 1992. 78

[Ram02] Ramamoorthi R.: Analytic pca construction for theoretical analysis of lighting variability in images of a lambertian object. *PAMI Oct 2002* (2002). 79

[RTG97] Rushmeier H. E., Taubin G., Guéziec

A.: Applying shape from lighting variation to bump map capture. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97* (1997), Springer-Verlag, pp. 35–44. 72

[Sch04] SCHNEIDER M.: Real-Time BTF Rendering. In *Proceedings of CESCG 2004* (2004). 85

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics 22*, 3 (2003), 382–391. 87

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 527–536. 83, 86

[SLSS03] SLOAN P.-P., LIU X., SHUM H.-Y., SNYDER J.: Bi-Scale Radiance Transfer. *ACM Transactions on Graphics 22*, 3 (2003), 370–375. 86, 87

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. *Proceedings of the Eurographics Symposium on Rendering 2003* (2003). 79, 84, 86, 87, 88

[SvBLD03] SUYKENS F., VOM BERGE K., LAGAE A., DUTRÉ P.: Interactive Rendering of Bidirectional Texture Functions. In *Eurographics 2003* (September 2003), pp. 463–472. 78, 80, 84, 87

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 665–672. 82, 86

[VT03] VASILESCU M. A. O., TERZOPOULOS D.: Tensortextures. In *Siggraph* (2003). 79

[WAA*00] WOOD D. N., AZUMA D. I., ALDINGER K., CURLESS B., DUCHAMP T., SALESIN D. H., STUETZLE W.: Surface light fields for 3D photography. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), pp. 287–296. 72

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), ACM Press, pp. 265–272. 71, 76

[WBS02] WALD I., BENTHIN C., SLUSALLEK P.: OpenRT – A Flexible and Scalable Rendering Engine for Interactive 3D Graphics. submitted for publication, meanwhile available as a Technical Report, TR-2002-01, Saarland University, 2002. 83, 90

[WHON97] WONG T.-T., HENG P.-A., OR S.-H., NG W.-Y.: Image-based rendering with controllable illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97* (1997), Springer-Verlag, pp. 13–22. 76

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 479–488. 80

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Realtime Ray Tracing and its use for Interactive Global Illumination. In *Eurographics State of the Art Reports* (2003). 82, 83