Compression and real-time Rendering of measured BTFs using local PCA

Gero Müller, Jan Meseth, Reinhard Klein

University of Bonn
Institute of Computer Science II
Römerstrasse 164, 53117 Bonn, Germany
Email: {gero, meseth, rk}@cs.uni-bonn.de

Abstract

The Bidirectional Texture Function (BTF) is a suitable representation for the appearance of highly detailed surface structures under varying illumination and viewing conditions. In most current approaches the BTF is aquired by a series of images of a flat probe taken under different light and camera positions. Due to its huge size, real-time rendering of objects textured with this six-dimensional data requires suitable approximations.

In this paper we present a new approach for BTF-compression and real-time rendering using much less memory than other comparable approaches whilst achieving similar quality. Our method exploits a BRDF-wise arrangement of the data and employs a flexible generalization of the well-known Principal Component Analysis (PCA) named *local PCA* for the data compression.

1 Introduction

The visual quality of computer generated images depends on exact geometric modeling and realistic simulation of light and reflection. Because full and physical correct simulation of the interaction between light and matter is infeasible in practice, considerable effort has been spent on developing reflection models, which describe the reflection properties of a possibly wide range of materials with a small set of parameters.

Most of these approaches model the Bidirectional Reflectance Distribution Function (BRDF) of surface points. Since they describe reflectance on a micro-scale only (the underlying geometry is too small to be visible to the viewer), they cannot capture the full complexity of reflection behavior on inhomogenous materials.

Materials with spatially varying reflectance properties can be described by the BTF, which was introduced by Dana et al [4]. Since the complex effects like self-occlusions and inter-reflections arising from the arbitrary mesostructure captured by measured BTFs seem to be intractable by a general and simple mathematical model, the BTF is usually represented by a series of images taken under different light and camera poses. Unfortunately, typical materials contain high-frequency content both in the spatial and angular domain, requiring a high-density sampling consisting of thousands of images to represent the BTF with sufficient quality [18]. Rendering directly from this data via linear interpolation is impractical and obviously not real-time even on todays sophisticated hardware. Therefore some sort of approximation is required. Such renderable BTF representations should simultaneously achieve good approximation quality, high compression rates and provide the possibility for real-time rendering.

In this paper, we analyze PCA-based BTF approximation and present the following key improvements over existing methods:

- Instead of applying PCA to sets of images, we utilize a BRDF-arrangement of the data.
- We applied the *local PCA* [10] to the data, a method combining Vector Quantization (VQ) and PCA exploiting the local linearity of high-dimensional datasets.

We show experimentally, that these enhancements significantly reduce the approximation error. As a result the number of PCA components can safely be reduced, which makes real-time rendering possible and tremendously increases the compression rate compared to similar methods. Finally, we present a real-time rendering algorithm for our model exploiting the abilities of current graphics boards.

2 Related Work

Our work is naturally related to any work done in the field of image-based rendering, especially if new images are generated from a huge set of pre-acquired images and thus data compression has to be applied to deal with the large amount of data.

In their pioneering work Levoy and Hanrahan [14] introduced light-field rendering and they used VQ and Lempel-Ziv entropy encoding for the compression. Miller et al. [21] parameterized light fields over surfaces (introducing Surface Light Fields) and encoded the images JPEG-like with a discrete cosine transform. In following work surface light fields were compressed using tailored versions of either VQ or PCA in order to handle the irregular sampled data [29] and by PCA applied to subsets (around a vertex of the parameterized mesh) of the resampled data [3][22].

While light fields allow renderings from novel viewpoints, reflectance fields enable rendering under arbitrary lighting conditions. Debevec et al. [8] acquired and rendered surface reflectance fields from human faces. Generation of novel viewpoints was made possible by exploiting a human skin reflectance model. Matusik et al. [16] compressed reflectance fields by block-wise PCA. The Polynomial Texture Map by Malzbender et al. [15] approximates nearly planar and diffuse surface reflectance fields efficiently by fitting a polynomial to each texel. Ashikhmin and Shirley [1] used basis textures lit by a steerable light basis for relighting.

Furukawa et al. [9] extended surface light field rendering by (sparsely) sampling the whole angular space. To handle the increased amount of data, they used a generalization of PCA called Tensor Product Expansion. The full angular space is also covered by the method of Lensch et al. [12] who used conventional BRDF-modeling in combination with an iterative clustering procedure to cope with insufficient sample density.

Our work is more closely related to methods that consider material independently from geometry, which can roughly be grouped in two categories.

The first group interprets the BTF as a set of spatially varying BRDFs and fits simple analytic functions to the discrete BRDFs achieving impressive compression rates. Real-time rendering is possible, since the BRDF representations require few parameters and can be evaluated efficiently in hardware. McAllister et al. [19] fitted Lafortune lobes to the BRDFs that they defined per discrete surface point (i.e. texel). Daubert et al. [7] additionally incorporated a view-dependent look-up table dealing with the flattening effect of the simple Lafortune-model[1]. This effect was also noted in recent work of Meseth et al. [20] who proposed fitting the surface reflectance fields independently in order to preserve the depth structure of the surface. Still, this group of methods suffers from lack of rendering quality due to the simplicity of the fitted analytic functions.

The second group of methods was developed in the context of pattern-recognition and they intended to model the statistical properties of BTFs. The extracted information has been used in classification and recognition of materials under arbitrary light and viewing conditions (see for example [6], [13], [26] and [28]) or for BTF-synthesis [5][27]. Recently, Sattler et al. [23] presented the first BTF-rendering method based on this kind of data-driven approach. They used PCA on subsets of the images that represent the sampled BTF and were able to decode and render the compressed data at high quality at interactive frame-rates. But their method is restricted to scenes containing very few materials only, since even one compressed BTF still consumes more than 200MB.

3 BTF-Compression

The BTF as originally introduced by Dana et al. [4] is a function of 2D-texture varying with 4D light and view direction (neglecting wavelength). In this section we describe our sampling of this function and how data compression can effectively be applied to the sampled data such that real-time rendering becomes possible.

3.1 Definitions

As depicted in figure 1, a sampled BTF can be interpreted as a collection of discrete textures

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

where $\mathcal{M}$ denotes the set of discrete measured view- and light-directions $(\mathbf{v},\mathbf{l})$, or as a set of tabu-

[1] Of course the precision of a Lafortune-representation can be increased by adding more lobes. Unfortunately, non-linear fitting is only practical up to 3-4 lobes.

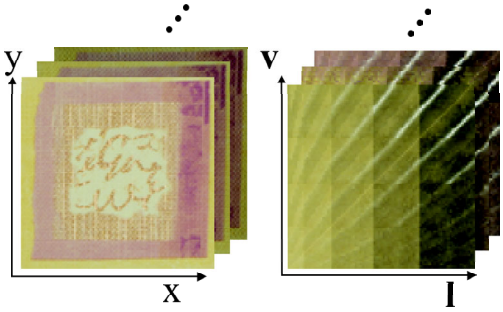

Figure 1: Two arrangements of the BTF data: As set of images (left) and as set of BRDFs (right).

lated BRDFs

$$\mathbf{BTF}_{Brdf} := \left\{B_{(\mathbf{x})}\right\}_{\mathbf{x}\in I\subset\mathbb{N}^2}.$$

Note, that these BRDFs do not fulfill physically demanded properties like reciprocity. Furthermore, they already contain a factor $(\mathbf{n}\cdot\mathbf{l})$ between incident direction and surface normal. This is nicely illustrated in figure 1.

The BTF-data employed in this work is a high-quality RGB-sampling with $|\mathcal{M}| = 81\times 81$ and $|I| = 256\times 256$ leading to more than 1.2GB of data (consider [23] for details on the measurement procedure). Real-time rendering of the raw data is impractical, considering that a typical scene will contain several materials. Therefore some sort of data compression has to be applied.

A very common technique is the PCA, which provides the in a least-squares sense optimal affine-linear approximation. Linear approximations have the advantage of fast fitting time and efficient evaluation on modern graphics hardware. But unfortunately, BTFs generally exhibit highly non-linear effects like specularities or self-occlusions due to viewpoint change which may require many coefficients in order to be reproduced in a visually satisfying quality. On the other hand Basri and Jacobs [2] have shown recently, that the space of images of a lambertian object under arbitrary illumination lies close to a linear 9D-subspace. At least PCA can provide some information about the complexity of the dataset.

3.2 Eigen-Textures vs. Eigen-BRDFs

To apply PCA to the sets $\mathbf{BTF}_{Tex}$ and $\mathbf{BTF}_{Brdf}$, we simply interpret the elements $T_{(\mathbf{v},\mathbf{l})}$ and $B_{(\mathbf{x})}$ as 3x256x256 and 3x81x81 elemental vectors respectively, subtract the mean (denoted by $\bar{T},\bar{B}$, respectively) from each vector and put them into a large matrix A[2]. The principal components are the eigenvectors of the symmetric matrix AA^T. We used the numerical package LAPACK to solve the resulting large eigen-problems. The principal components E_i^{Tex} of $\mathbf{BTF}_{Tex}$ are well known as Eigen-Textures, thus we simply call the components E_i^{Brdf} *Eigen-BRDFs*.

The reconstructions $\tilde{T}^c_{(\mathbf{v},\mathbf{l})}$ and $\tilde{B}^c_{(\mathbf{x})}$ are given as follows:

$$\tilde{T}^c_{(\mathbf{v},\mathbf{l})} = \bar{T} + \sum_{i=1}^{c}\langle T_{(\mathbf{v},\mathbf{l})} - \bar{T}, E_i^{Tex}\rangle * E_i^{Tex}$$

$$\tilde{B}^c_{(\mathbf{x})} = \bar{B} + \sum_{i=1}^{c}\langle B_{(\mathbf{x})} - \bar{B}, E_i^{Brdf}\rangle * E_i^{Brdf}$$

Since the rendering algorithm presented in section 4 can efficiently handle $c \leq 8$ we are interested in the error remaining for a given c. We measure the average absolute reconstruction error

$$\epsilon(\mathbf{BTF}_{Tex}, c) = \sum_{(\mathbf{v},\mathbf{l})\in\mathcal{M}} \frac{||T_{(\mathbf{v},\mathbf{l})} - \tilde{T}^c_{(\mathbf{v},\mathbf{l})}||}{|I||\mathcal{M}|}$$

with a similar expression for $\epsilon(\mathbf{BTF}_{Brdf}, c)$.

According to our measurements, the BRDF-arrangement performs slightly better (see figure 2 for an example). This can be explained with the eigenvalue-plot in figure 2: The variance in $\mathbf{BTF}_{Brdf}$ depends only on the complexity of the surface i.e. on *spatial* variation of reflectance properties and shadowing- and masking-effects while in $\mathbf{BTF}_{Tex}$ additional variance is introduced by the measurement process i.e. registration errors and filtering. Furthermore, specularities introduce strong variance among the textures, while their variation across the BRDFs depends only on the material structure.

But nevertheless the average error of an 8-component Eigen-BRDF approximation is still 3.8% (4.2% for Eigen-Texture), which results in notable blurring in the reconstructions (figure 3). But since using more components is prohibited, another way to error reduction has to be found. In the next section we will demonstrate a method to significantly reduce the error by using a combination of VQ and PCA.

[2] In this case not the whole dataset was used as the training set, because it otherwise would exceed the maximum memory block size

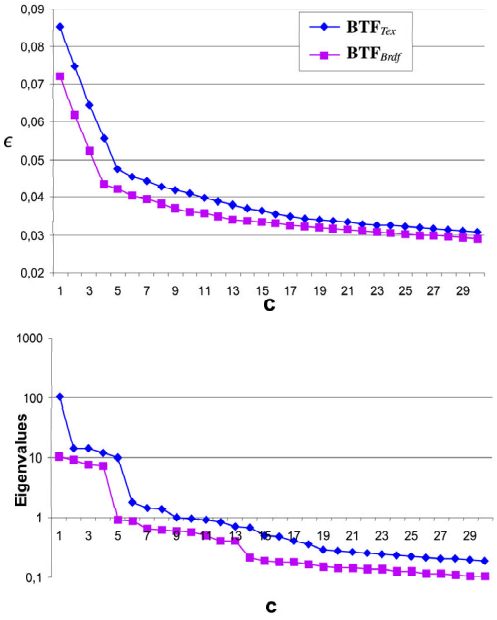

Figure 2: Average reconstruction error (top) and eigenvalues for the material *proposte*.

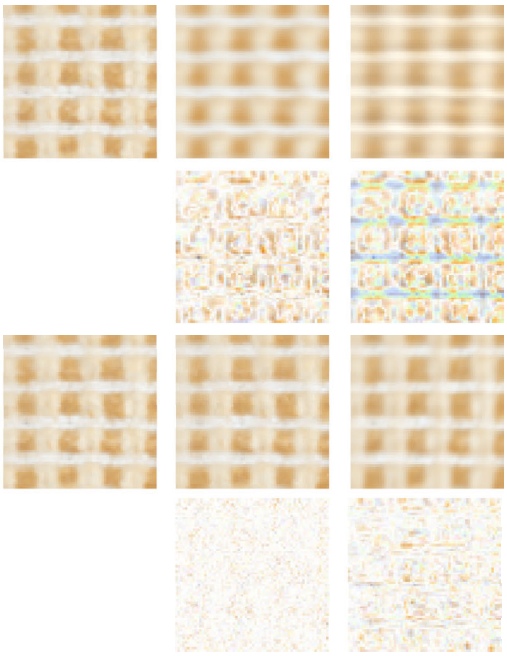

Figure 3: The original frontal view of the *proposte* texture (left), the reconstruction for $c = 8$ without clustering (top) and with $k = 32$ clusters (bottom) from **BTF**$_{Brdf}$ (middle) and from **BTF**$_{Tex}$ (right) with enhanced (multiplied by 4) and inverted difference images.

3.3 Error-reduction with local PCA

Although not globally linear, many high-dimensionally datasets show a local linear behavior. This basic observation leads to the local PCA method, which was introduced by Kambhatla and Leen[10] to the machine-learning community in competition to classical non-linear/neural-network learning algorithms[3]. In contrast to more sophisticated non-linear dimensionality reduction techniques as used for example recently in data-driven BRDF-modeling [17] this method introduces no additional run-time cost to the reconstruction apart from a simple cluster look-up.

The encoding stage of the algorithm can be summarized as follows:

1. Initialize k cluster-centers r_j randomly chosen from the dataset. Assign a collection of c unity basis-vectors $e_{i,j}$ to each cluster.
2. Partition the dataset into regions by assigning each data-vector to its closest center. The distance to a center r_j is given by squared reconstruction error:

$$||x-\tilde{x}_j||^2 = ||x-r_j - \sum_{i=1}^{c} \langle x-r_j, e_{i,j}\rangle e_{i,j}||^2$$

3. Compute new centers r_j as the mean of the data in the region j.
4. Compute a new set of basis-vectors $e_{i,j}$ per region, i.e. perform a PCA in each region.
5. Iterate steps 2.-4. until the change in average reconstruction error falls below a given threshold.

We applied the method to both **BTF**$_{Tex}$ and **BTF**$_{Brdf}$. Reconstructed samples are depicted in figure 3. The improvement in average reconstruction error depending on the number of clusters k is shown in figure 4. Note, that more than 30 PCA-components are needed to reach the quality achieved by using 16 clusters and only 8 components. Since the reconstruction cost is dominated by the number of

[3] Independently from our work this method was introduced into computer graphics in a recent paper on Precomputed Radiance Transfer[25].

components which have to be summed up during rendering, this enables fast rendering as shown in section 4.

Please note also the superior behavior of the BRDF-wise arrangement. This supports intuition, since most materials are made of resembling parts leading to natural clusterings. Another great advantage of the BRDF-wise arrangement arises from the fact, that the spatial sampling density is usually higher than the angular one ($|I| > |\mathcal{M}|$). The memory needed for an additional cluster is given by the number of the used principal components which have dimension $|I|$ or $|\mathcal{M}|$ respectively. In our case $|I| \approx 10 * |\mathcal{M}|$ what is also reflected in table 1. In a typical case with $k = 32, c = 8$ we achieve a compression ratio of about 1:100 against 1:10.

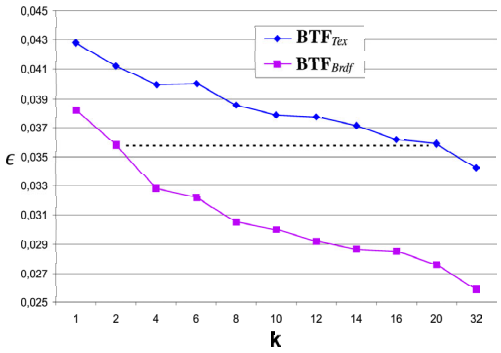

Figure 4: Average reconstruction error for increasing number of clusters k and fixed number of components $c = 8$ (*proposte*).

	$k\backslash c$	4	6	8	10
Tex	4	6.3	9.5	12.7	15.9
Brdf		1.15	1.73	2.3	2.9
	8	12.6	18.9	25.3	31.6
		1.78	2.68	3.57	4.46
	16	25.2	37.8	50.4	63
		3.04	4.6	6	7.6
	32	50.4	75.5	100.8	125.9
		5.6	8.4	11.1	13.9

Table 1: Size of the local PCA-compressed data in *million bytes* depending on the number of clusters k and components c. The data format is 16-bit float. The size of the original data is 1.2GB.

4 BTF-Rendering

4.1 Rendering Algorithm

Accurate rendering algorithms have to compute results approximating the rendering equation for every surface point $\mathbf{x}$ by computing outgoing radiance L_r as follows (neglecting emissive effects):

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

Here, $BRDF_{\mathbf{x}}$ denotes the BRDF for point $\mathbf{x}$, L_i denotes incoming radiance, n is the surface normal and Ω_i is the hemisphere over $\mathbf{x}$. Employing measured BTFs, the following approximation results:

$$L_r(\mathbf{x},\mathbf{v}) \approx \int_{\Omega_i} BTF(\mathbf{x},\mathbf{v},\mathbf{l}) L_i(\mathbf{x},\mathbf{l})\, d\mathbf{l}$$

The area foreshortening term is removed since this effect is represented in the measured BTFs already. In the presence of a finite number of point light sources only, the integral reduces to a sum. Approximating the BTF by the clustered Eigen-BRDFs, the above equation reduces to

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=1}^{n} \sum_{k=1}^{c} \alpha_{k,m}(\mathbf{x},\mathbf{v},\mathbf{l}_j) E_{k,m}^{Brdf}(\mathbf{v},\mathbf{l}_j) L_i(\mathbf{x},\mathbf{l}_j)$$

Here, n denotes the number of light sources, α denotes the projections on the respective basis vectors as in section 3.1, c is the number of components from the clustered Eigen-BRDFs $E_{i,m}^{Brdf}$ (m is the cluster or material index). Since the BTF was sampled for fixed light and view directions only, linear interpolation is employed to support arbitrary view and light directions, resulting in the final equation:

$$L_r(\mathbf{x},\mathbf{v}) \approx \sum_{j=1}^{n} \sum_{\substack{\tilde{\mathbf{v}} \in N(\mathbf{v}) \\ \tilde{\mathbf{l}} \in N(\mathbf{l}_j)}} w_{\tilde{\mathbf{v}},\tilde{\mathbf{l}}} \sum_{k=1}^{c} \alpha_{k,m}(\mathbf{x},\tilde{\mathbf{v}},\tilde{\mathbf{l}}) E_{k,m}^{Brdf}(\tilde{\mathbf{v}},\tilde{\mathbf{l}}) L_i(\mathbf{x},\mathbf{l}_j)$$

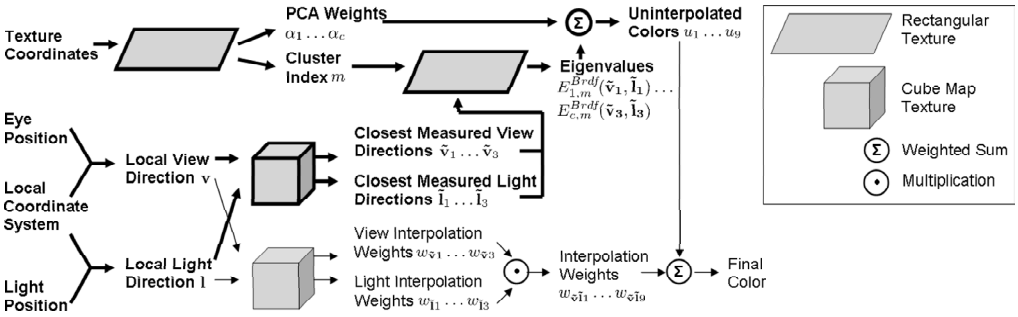

Figure 5: Data Flow during Rendering. The elements of the basic rendering algorithm are highlighted by bold font, thick arrows and thick borders. The other elements contribute to the view and light direction interpolation. The required data is stored in several textures. All processing takes place in the pixel shaders.

By $N(\mathbf{v})$ we denote the set of neighboring view directions of $\mathbf{v}$, for which data was measured ($N(\mathbf{l})$ respectively), while w denotes an interpolation weight.

The BTF data needed for rendering is stored in three textures as depicted in figure 5. A rectangular texture T_1 stores material cluster indices together with sets of floating-point PCA weights α which determine, how the components of the indexed Eigen-BRDFs are to be combined. Another rectangular texture T_2 holds the floating-point valued components of the Eigen-BRDFs for the various clusters.

We employ two cube-maps to determine the three closest measured view- and light-directions for the current view- and light-direction and the respective interpolation weights.

The rendering process is depicted in figure 5. The inputs are standard texture coordinates, the eye and light positions, and a per-pixel coordinate system, which is interpolated from the local coordinate systems at the vertices which are specified with the geometry and stored in display lists.

We first compute view and light directions and transform them into the pixel's coordinate system. Using cube maps, we lookup the three closest view and light-directions from the measurement process, together with their interpolation weights (please note that for the basic rendering algorithm only the nearest view and light direction is needed). The interpolation weights for each pair of closest view and light direction are multiplied to yield the final nine interpolation weights.

At the same time, the basic rendering algorithm is performed: using the texture coordinates, we lookup cluster indices and PCA weights for every pixel on the screen (just as we would lookup colors for texture mapping). The cluster index is used to select the appropriate 81x81x3 vectors representing the c Eigen-BRDFs of the current cluster from T_2. The view and light direction are used to select the appropriate RGB components from the large vectors. If interpolation is used, this lookup is repeated for every pair of closest view and light direction. Combining the PCA weights of the current pixel with the RGB components, the uninterpolated colors are computed (one for each pair of closest view and light direction). In a final step, these colors are multiplied with their respective interpolation weights and the results are summed to form the final color of the pixel.

4.2 Results

We implemented our rendering algorithm on a Geforce FX graphics board, since it supports the Pixel Shader 2.0+ extension, which permits fragment programs containing up to 1024 instructions. A single rendering pass with a single light source only and without interpolation requires about 50 instructions. Interpolation between the nine closest measured pairs of light and view directions increases the count to about 300 instructions - a quite challenging number even for the newest graphics boards with 8 pixel pipelines and GPU clock frequencies of up to 500 MHz. We achieved frame-rates of about 14 Hz for typical models. We expect better frame-rates for upcoming versions of the current Windows OpenGL driver, which appears not

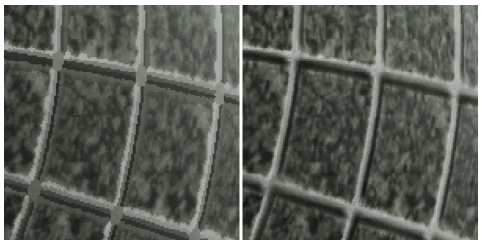

Figure 6: Magnified view of the stone covered sphere from figure 8. The images visualize the reduced quality if byte values are used to represent PCA components and PCA weights.

to be optimized to handle large amounts of texture memory - especially if floating point textures are employed.

Our approach was tested with several materials and models. Figure 7 shows the tremendous quality difference between simple lit textures and BTF rendering. Both pictures were taken under the same lighting conditions. As the samples from the measurement show, the appearance of the material changes drastically under varying light and view directions which is correctly modelled by the BTF rendering algorithm, leading to specular highlights, accurate dimming and realistic color changes. The lit textures (diffuse lighting was assumed) cannot (sufficiently) reproduce these effects. Please note that in this case even bump mapping would not produce substantially better results than the lit textures since the wallpaper is approximately flat.

In figure 6 the difference in quality using either 8 bit fixed point (byte) or 16 bit floating point (float) valued PCA components and weights is shown. In the left image, both components and weights were represented by byte values. In the right image float values were employed, leading to very smooth color changes. Using byte values produces renderings with alias effects, incorrectly spaced and shaded highlights and an overall lack of sharpness.

Figure 9 shows the astonishing depth impression achieved by our BTF rendering method. The proposte material makes the flat, triangular cloth model appear very bumpy. In figure 8, correct self-shadowing of the material and specular highlights are visible, while the knitted wool material in figure 10 correctly reproduces real-world effects like loss of structure for grazing viewing angles (i.e. the otherwise bumpy surface appears approximately flat).

5 Conclusions and Future Work

In this work, we presented a flexible framework for compression and real-time rendering of measured BTFs achieving better compression rates and lower error than other comparable methods. We applied the local PCA-algorithm to two arrangements of the dataset and found out experimentally, that the arrangement in terms of tabulated BRDFs performs much better than the classical per-image approach both in memory requirements and approximation error. This result can be expected for most real world materials, where surface-parts show resembling reflectance behavior.

In addition, we presented a rendering algorithm running completely on the GPU that achieves real-time frame rates for moderate resolutions, due to the large number of per-pixel operations. Additional hardware features allowing simultaneous lookup from several texture units would significantly speed up this process (e.g. by looking up all 8 PCA components at once, the number of instructions would be reduced to about a third).

For future work we will inspect, if the structure of the principal components gives rise to further modeling. One possibility would be for example the application of common BRDF-modeling techniques to the Eigen-BRDFs.

Furthermore, we will implement a MIP-mapping strategy for rendering. Generation of distinct MIP-levels will require recomputation of cluster indices and PCA weights for the pixels of the MIP-map. Linear interpolation between different MIP-levels will be possible as well but will require specifically designed shaders.

Additionally, we plan to extend our rendering algorithm to allow natural, image-based lighting following the approach described by Sloan et al. [24] and Kautz et al. [11].

Acknowledgements

This work was funded by the European Union under the project RealReflect (IST-2001-34744). The shirt model was kindly supplied by the Virtual Try-On project (BMBF 01IRA01A). Special thanks belong to Ralf Sarlette who provided the BTF measurements.

References

[1] M. Ashikhmin and P. Shirley, Steerable illumination textures. In *ACM Transactions on Graphics*, **21**(1), pp. 1–19, 2002

[2] R. Basri and D. W. Jacobs, Lambertian Reflectance and Linear Subspaces. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **25**(2), pp. 218–233, 2003

[3] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. In *SIGGRAPH 2002*, pp. 447–456, 2002

[4] K. J. Dana, B. van Ginneken, S. K. Nayra, and J. J. Koenderink. Reflectance and Texture of Real World Surfaces. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 151–157, 1997

[5] K. J. Dana and S. K. Nayar, Histogram model for 3d textures. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 618–24, June 1998

[6] K. J. Dana and S. K. Nayar. Correlation Model for 3D Texture. In *IEEE Int. Conf. on Computer Vision*, pp.1061–1066, 1999

[7] K. Daubert, H. Lensch, W. Heidrich and H. P. Seidel. Efficient Cloth Modeling and Rendering. In *12th Eurographics Workshop on Rendering*, pp. 63–70, 2001

[8] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH 2000*, pp. 145–156, 2000

[9] R. Furukawa, H. Kawasaki, K. Ikeuchi and M. Sakauchi. Appearance based object modeling using texture database: Acquisition compression and rendering In *13th Eurographics Workshop on Rendering*, June 2002

[10] N. Kambhatla and T. K. Leen. Dimension Reduction by Local PCA In *Neural Computation, 9*, pp. 1493–1516, 1997

[11] J. Kautz, P.-P. Sloan, J. Snyder Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics In *12th Eurographics Workshop on Rendering*, pp. 301–308, 2002

[12] H. Lensch, M. Goesele, J. Kautz, W. Heidrich, and H.-P. Seidel. Image-Based Reconstruction of Spatially Varying Materials. In *12th Eurographics Workshop on Rendering*, pp. 103–114, 2001

[13] T. K. Leung and J. Malik. Recognizing Surfaces using Three-Dimensional Textons. In *IEEE Int. Conf. on Computer Vision*, pp. 1010–1017, 1999

[14] M. Levoy and P. Hanrahan. Light Field Rendering. In *Computer Graphics, Volume 30*, pp. 31–42, 1996

[15] T. Malzbender, D. Gelb, and H. Wolters. Polynomial texture maps. In *SIGGRAPH 2001*, pp. 519–528, 2001

[16] W. Matusik, H.-P. Pfister, A. Ngan, P. Beardsley, R. Ziegler and L. McMillan. Image-Based 3D Photography using Opacity Hulls. In *ACM Transactions on Graphics*, **21**(3), pp. 427–437, 2002

[17] W. Matusik, H.-P. Pfister, M. Brand, and L. McMillan. A Data-Driven Reflectance Model. In *SIGGRAPH 2003*, 2003

[18] D. K. McAllister. A Generalized Representation of Surface Appearance. *PhD thesis, University of North Carolina*, 2002

[19] D. K. McAllister, A. Lastra, and W. Heidrich. Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. In *Graphics Hardware 2002*, pp. 78–88, 2002

[20] J. Meseth, G. Müller, R. Klein. Preserving Realism in real-time Rendering of Bidirectional Texture Functions. In *OpenSG Symposium 2003*, pp. 89-96, April 2003.

[21] G. Miller, S. Rubin and D. Ponceleon. Lazy Decompression of Surface Light Fields for Precomputed Global Illumination. In *9th Eurographics Workshop on Rendering*, pp. 281–292, June 1998

[22] K. Nishino, Y. Sato and K. Ikeuchi. Eigen-Texture Method: Appearance Compression and Synthesis Based on a 3D Model. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(11), pp.1257–1265, 2001

[23] M. Sattler, R. Sarlette, and R. Klein. Efficient and Realistic Visualization of Cloth. In *Eurographics Symposium on Rendering*, 2003

[24] P.-P. Sloan, J. Kautz, J. Snyder Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments In *SIGGRAPH 2002*, pp. 527–536, 2002

[25] P.-P. Sloan, J. Hall, J. Hart and J. Snyder Clustered Principal Components for Precomputed Radiance Transfer. In *SIGGRAPH 2003*, pp. 382–391, 2003

[26] P.-H. Suen and G. Healey. The Analysis and Recognition of Real-World Textures in Three Dimensions. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 5, pp.491–503, 2000

[27] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *SIGGRAPH 2002*, pp. 665–672, 2002

[28] M. Varma and A. Zisserman. Classifying Images of Materials: Achieving Viewpoint and Illumination Independence In *European Conference on Computer Vision*, pp. 255–271, 2002.

[29] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface Light Fields for 3D Photography. In *SIGGRAPH 2000*, pp. 287–296, 2000

Figure 7: Cloth covered with wallpaper. Comparison between BTF rendering (right) and lit textures (left). In the middle, two samples from the measurement process of the wallpaper material taken under varying view and light direction.

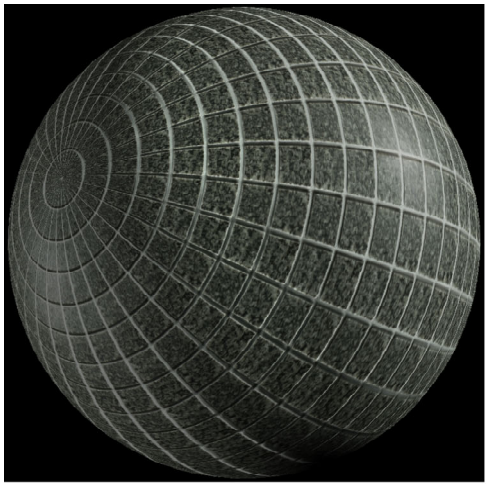

Figure 8: Stone covered Sphere.

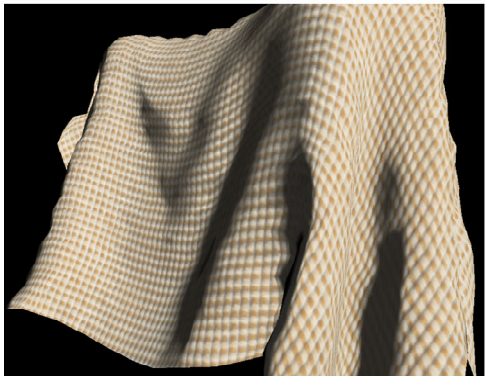

Figure 9: Proposte on Cloth

Figure 10: Woolen Shirt.