

Fast Environmental Lighting for Local-PCA Encoded BTFs

Gero Müller, Jan Meseth, Reinhard Klein

University of Bonn

Institute of Computer Science II

Römerstraße 164

D-53177, Bonn, Germany

{gero,meseth,rk}@cs.uni-bonn.de

Abstract

Rendering geometric models with complex surface materials in arbitrary lighting environments is a challenging problem. In order to relight and render geometries covered with complex, measured BTFs two problems have to be addressed: The memory problem resulting from the large size of the measured BTF data and the light integration problem resulting from summing up the contributions from all measured light-sources.

In this paper we describe how highly efficient BTF compression methods like Local-PCA and suitable representations of environmental light based on Spherical Harmonics can be combined leading to fast environmental lighting for efficiently encoded BTFs. As a side effect the method supports Precomputed Radiance Transfer.

1. Introduction

Realistic and efficient rendering of complex meso-structure is still a challenging task in today's computer graphics. Even for relatively flat materials like the wallpaper depicted in Figure 1 the changes in appearance for different light and view directions can be enormous. A full synthetic simulation of such a material with the classic rendering approach, i.e. explicit geometric modeling on a macro-scale and reflectance modeling on a micro-scale, seems impossible because of its complexity in modeling and computation.

This is the place where image-based rendering fits in: Instead of explicitly modeling reality, data captured from reality is used and new images are generated from that data. While applications of purely image-based techniques like Light Field Rendering (e.g. [13]) are still limited to special domains, combinations with geometry based approaches have made their way even into of-the-shelf consumer graphics hardware. And in the case of rendering complex meso-structure, texture-

mapping is nowadays the technique of choice and has introduced a new level of reality into computer graphics.

But of course simple textures do not capture light- and view-dependent effects like shadowing, masking, subsurface-scattering etc. These effects are included in the Bidirectional Texture Function (BTF), a 6-dimensional texture representation introduced by Dana et al. [3] which also depends on light- and view-direction.

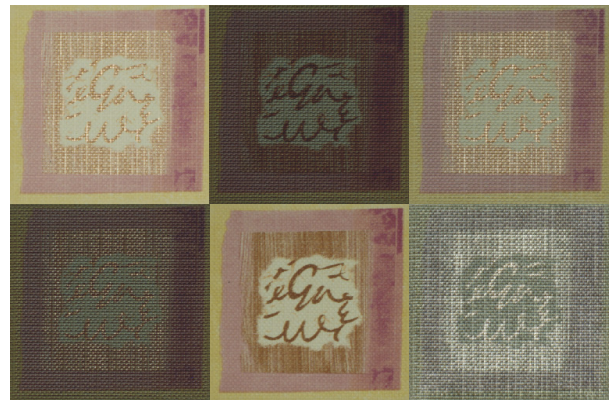


Figure 1. Six views of a wallpaper from various view and light directions. The appearance of the material changes drastically which cannot be reproduced by simple material representations like textures. The BTF correctly represents and reproduces these effects.

In most current approaches the BTF is acquired by a series of images of a flat material probe taken under different light- and camera configurations. Renderings containing view- and light-dependent effects can simply be obtained by choosing the images corresponding to the current view- and light-directions. In order to achieve smooth images linear interpolation between neighboring measurements should be done and the contributions from all light sources have to be com-

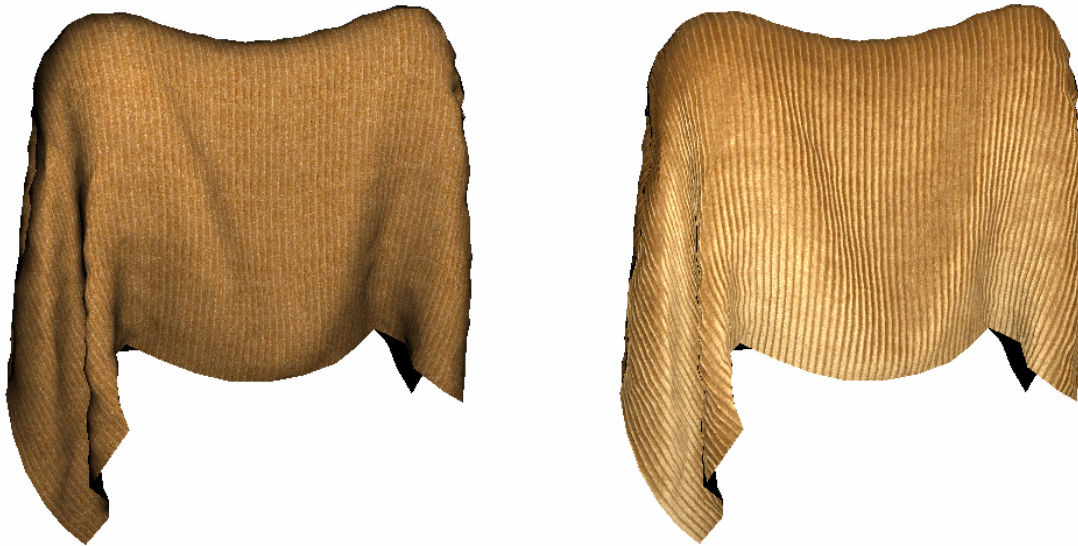


Figure 2. Comparison of simple texturing and BTF-rendering: On the left only one photo of a corduroy material was wrapped around the cloth. On the right a corduroy BTF was used.

bined. Figure 2 shows an example. Obviously there remain two big problems with this kind of approach:

1. Memory requirements – Typical BTF-samplings contain thousands of images. This restricts the number of materials in a scene and hampers real-time rendering.
2. Relighting the BTF with complex and large light sources is too expensive for real-time applications since the contributions from all measured light directions have to be combined.

Recently several papers have been published that address the first problem by providing efficient compression schemes that allow for fast and even real-time BTF-rendering with acceptable error on programmable graphics hardware (e.g. [4][15][16][26][19]). Unfortunately, they either omit environmental lighting or achieve insufficient rendering quality. Complex relighting of BTFs was addressed using Spherical Harmonics but published approaches either targeted pure bi-directional reflectance distribution (BRDF) rendering (e.g. [21][23]) or restricted the BTF to patches of small extent since no BTF-compression was applied [25]. This prohibits materials that require high spatial resolution due to both high- and low-frequency surface structure like the wallpaper in figure 1.

In this paper we propose compression with Local Principal Component Analysis (LPCA) and relighting with Spherical Harmonics addressing both the memory and the relighting problem. As a side effect we can combine BTF-rendering with precomputed radiance transfer enabling also large-scale shadows and inter-reflections.

2. Related Work

2.1 Image Based Rendering

Research in the area of image-based rendering nearly always deals with the two problems that we face as well: finding a way to compress the huge amounts of image data without introducing too much error and generating new images as fast as possible.

In their pioneering work Levoy and Hanrahan [13] introduced light-field rendering: an efficiently renderable 4D representation mainly used for complex real-world objects. They used vector quantization (VQ) and Lempel-Ziv entropy encoding for the compression. Wong et al. [27] addressed relighting of light fields encoding the light-dependence for every fixed view with spherical harmonics. Miller et al. [18] 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 [28] and by PCA applied to subsets (around a vertex of the parameterized mesh) of the resampled data [2]. The results vary in approximation quality and rendering speed.

While light fields allow renderings from novel viewpoints, reflectance fields enable rendering under arbitrary lighting conditions. Debevec et al. [5] acquired and rendered surface reflectance fields from human faces. Interactive generation of novel viewpoints was made possible by exploiting a human skin reflectance model. Matusik et al. [14] compressed re-

flectance fields by block-wise PCA. The Polynomial Texture Map by Malzbender et al. [17] efficiently approximates and compresses nearly planar and diffuse surface reflectance fields by fitting a polynomial to each texel.

Furukawa et al. [6] extended surface light field rendering by (sparsely) sampling the whole angular space. To handle the increased amount of data, they used a generalization of matrix factorization 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.

2.2 Real-Time BTF-Rendering

BTF-rendering can be understood as rendering of spatially varying materials under varying light and view conditions. A first real-time algorithm approaching the task of BTF-rendering was published by Kautz and Seidel [8]. They introduced techniques for evaluating spatially varying BRDFs on graphics hardware by factoring the BRDFs into two-dimensional functions and storing the values in textures that are recombined for rendering. The technique was recently improved by Suykens et al. [26] who achieve good results for simple synthetic BTFs. McAllister et al. [15] published a different method exploiting graphics hardware that approximates the BTF by pixelwise Lafortune [10] models. They achieve good results for approximately flat materials with highly specular behavior. One year earlier already, Daubert et al. [4] published a similar approach in the context of rendering synthetic cloth BTFs. They additionally modulated the pixelwise Lafortune models with a view-dependent factor in order to cope with self-occlusion effects. Unfortunately, their representation is not as compact as the previously mentioned ones. Meseth et al. [16] proposed an improved material representation based on reflectance fields which increases the real-time rendering quality of materials, especially if they feature high depth variation. They as well suffer from insufficient data compression. A different approach was introduced by Sattler et al. [22], who utilize PCA to compute Eigen-Textures that are composed during runtime based on view- and light-dependent weights. While it is capable of producing photo-realistic renderings, it reduces the huge amounts of data contained in a BTF just by a small factor. The method was improved in terms of memory requirements and approximation quality by an approach of Müller et al. [19]. They perform clustering of the spatially varying BRDFs by employing LPCA in order to compute Eigen-BRDFs.

2.3 Real-Time Environmental Lighting

Real-time rendering of geometric models with complex surface materials in environments with arbitrary lighting situations has been an area of active research for some time already. The approaches of Heidrich and Seidel [7], Kautz et al. [9] and McAllister et al. [15] assume fixed BRDFs (e.g. Phong or Lafortune) and tackle environmental lighting by prefiltering the environment with appropriate kernels. Unfortunately their methods achieve real-time frame rates for fixed lighting settings only and are restricted to rather simple material representations. A more general technique which handles arbitrary isotropic BRDFs was introduced by Kautz and McCool [8]. Although it can handle BRDFs of arbitrary complexity in principle, complex BRDFs require extensive run-time computations in order to achieve high quality results. In addition, prefiltering times are quite high. A more efficient approach was presented by Latta et al. [11] which allows arbitrary isotropic BRDFs as well. It reduces the evaluation time at the cost of extensive precomputations which practically prohibits dynamic changes of the lighting situation. These overheads can efficiently be reduced by projecting the environmental light into a Spherical Harmonics basis, which has been done by several recent publications ([20][21][23]). Although real-time rendering can be achieved for low-frequency environments only, the results look very realistic. Unfortunately, these approaches cannot simply be applied to BTFs. Sloan et al. [25] presented such a method but it is limited to BTFs of rather small spatial extent. They also included Precomputed Radiance Transfer as introduced in [23] in order to support large-scale shadows and inter-reflections.

3. BTF Representation

As depicted in figure 3, a sampled BTF can either be interpreted as a collection of discrete textures:

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

where Δ denotes the set of discrete measured view- and light-directions, or as a set of tabulated BRDFs:

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

Please note, that these BRDFs do not fulfill physically demanded properties like reciprocity. In special they already contain a factor $(\mathbf{n} \cdot \mathbf{l})$ between incident direction and surface normal. This is also nicely illustrated in figure 3.

The BTF-data employed in this work is a high-quality RGB-sampling with $|\Delta|=81 \times 81$ and

$|I|=256 \times 256$ leading to more than 1.2GB of data (samples available at <http://btf.cs.uni-bonn.de>). In order to reduce the memory requirements we apply the LPCA algorithm to BTF_{Brdf} as in [19]. The resulting compressed representation has the form

$$\text{BTF}_{\text{Brdf}} \approx \left\{ \tilde{B}_{\mathbf{x}}^c \right\}_{\mathbf{x} \in I \subset \mathbb{N}^2}$$

where

$$\tilde{B}_{\mathbf{x}}^c = \bar{B}_{k_B(\mathbf{x})} + \sum_{i=1}^{c_B} \left\langle B_{\mathbf{x}} - \bar{B}_{k_B(\mathbf{x})}, E_{i,k_B(\mathbf{x})} \right\rangle \cdot E_{i,k_B(\mathbf{x})}$$

denotes the reconstruction of the BRDF at texel \mathbf{x} from c_B components of the PCA applied to the BRDFs in cluster $k_B(\mathbf{x})$ looked up from texel \mathbf{x} . The $E_{i,k}$ are the so called Eigen-BRDFs (i.e. the PCA-components in cluster k) and the mean is denoted by $\bar{B}_{k_B(\mathbf{x})}$. Adjusting the number of components c_B and the number of clusters $|k_B|$ allows for finding a flexible trade-off between memory-requirements, approximation quality and rendering speed. For real-time rendering c_B should be chosen as small as possible ($c_B < 8$), since rendering time depends on the number of components that have to be summed up. Depending on the material complexity compression ratios between 1:50 and 1:250 introducing a relative error of only about 2% can be achieved.

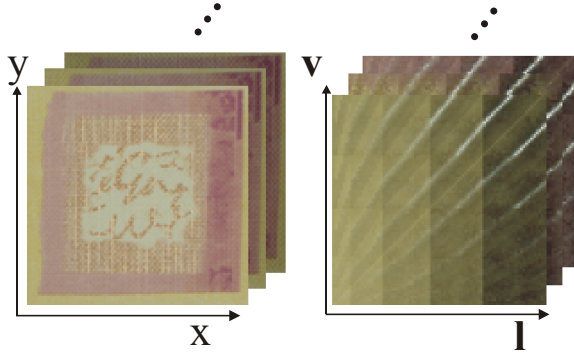


Figure 3. Two arrangements of the BTF-data: As set of images (left) and as set of BRDFs (right).

4. BTF Relighting

Relighting BTF-covered geometry with an arbitrary and distant illumination environment L_i requires computing the lighting integral

$$L_r(\mathbf{p}, \mathbf{v}) = \int_{\Omega_i} \text{BTF}(T(\mathbf{p}), \mathbf{v}, \mathbf{l}) \cdot L_i(\mathbf{R}_{\mathbf{p}}(\mathbf{l})) d\mathbf{l} \quad (1)$$

over the incident hemisphere Ω_i at surface point \mathbf{p} . Here T denotes a texture coordinate mapping from the surface into the BTF and $\mathbf{R}_{\mathbf{p}}$ represents a rotation matrix from the local coordinate frame at \mathbf{p} into global coordinates.

4.1 Directional Light Basis

Since the BTF-dataset represents the material's response to a set Λ of directional light sources \mathbf{l}_j for some different views, this integral always reduces to a sum via projecting the illumination into this finite light basis and summing up the corresponding measurements with the resulting weights:

$$L_r(\mathbf{p}, \mathbf{v}) = \sum_{j=1}^{|\Lambda|} \text{BTF}(T(\mathbf{p}), \mathbf{v}, \mathbf{R}_{\mathbf{p}}^{-1}(\mathbf{l}_j)) \cdot \langle L_i, \mathbf{l}_j \rangle$$

For environments containing large area light sources the evaluation of this sum is obviously rather inefficient since many weights will be non-zero. So this approach is only suitable for rendering of scenes containing only a few directional or point light sources.

4.2. Spherical Harmonics Light Basis

Large and slowly varying light sources are much better represented in a low-order Spherical Harmonics (SH) basis expansion. Therefore we represent the incident lighting L_i and the BTF for each texel \mathbf{x} and view direction \mathbf{v} in an n -th order (typically $n=5$) SH-basis expansion:

$$L_i(\mathbf{l}) \approx \sum_j^{n^2} a_j y_j(\mathbf{l})$$

$$\text{BTF}(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx \sum_j^{n^2} b_j^{\mathbf{x}, \mathbf{v}} y_j(\mathbf{l})$$

The projection coefficients a_j and $b_j^{\mathbf{x}, \mathbf{v}}$ can be obtained via numerical integration. Due to the orthogonality of the SH basis the lighting-integral now reduces to a simple dot product:

$$L_r(\mathbf{p}, \mathbf{v}) = \langle \mathbf{b}^{\mathbf{v}, T(\mathbf{p})}, \mathbf{M}_{\mathbf{p}}(\mathbf{a}_i) \rangle \quad (2)$$

By \mathbf{a}_i and $\mathbf{b}^{\mathbf{v}, T(\mathbf{p})}$ we denote the vectors of stacked SH coefficients and $\mathbf{M}_{\mathbf{p}}$ denotes the high-dimensional SH-rotation matrix that rotates the lighting environment into the local coordinate frame. As in [23] this matrix could also represent a transfer matrix encoding for example large scale shadows and inter-reflections.

Sloan et al. [25] published a BTF rendering algorithm based on this formulation.

4.3 Combining Clustered Transfer and LPCA encoded BTFs

Unfortunately the approach in [25] cannot easily be applied to BTFs that require high spatial resolution like our measured ones.

The first problem is that we need to store about 130 million SH-coefficients per channel for a fifth order SH-expansion of our measured BTFs resulting in almost 400MB storage using only one byte per coefficient. The second problem of the method is the bottleneck formed by transferring incident lighting into the local coordinate frame, which has to be done on the CPU for every mesh vertex.

We propose LPCA encoding for the BTF coefficients to tackle the first problem. Instead of storing SH-coefficients for every texel only the Eigen-BRDFs have to be represented in the SH-basis.

The second problem can be solved by applying LPCA also to the transfer matrices as done in [24]. Then transferring incident lighting can be handed over from CPU to GPU decoupling rendering speed from the size of the mesh.

A straight-forward implementation would now simply combine the algorithms from [24] [25] and [19]. Unfortunately the costs still remain relatively high: For every pixel and every frame – even for fixed lighting – (c_B+1) dot products between n^2 -component vectors (view-dependent BTF components and lighting) have to be computed and summed up with the corresponding weights. Additionally for every vertex the transferred lighting vector has to be reconstructed and interpolated over the triangle. To avoid this large overhead we propose an alternative method that allows for faster rendering with fixed lighting and decouples the computation of the dot products from screen resolution. It is based on the LPCA expansion of the transfer matrices and the BTF.

Expressed in its LPCA basis the transfer matrix \mathbf{M}_p is given by

$$\mathbf{M}_p \approx \sum_{j=0}^{c_M} \alpha_{p,j} \mathbf{m}_{j,k_M(p)}$$

The view-dependent SH-coefficients for every BTF-texel can be reconstructed from the mean and Eigen-BRDF SH-coefficients \mathbf{e}_{j,k_B}^v :

$$\mathbf{b}^{v,x} \approx \sum_{j=0}^{c_B} \beta_{x,j} \cdot \mathbf{e}_{j,k_B}^v(x)$$

The cluster index look-up is denoted with $k_M()$ and $k_B()$ respectively. Now equation (2) can be expressed in terms of the LPCA components:

$$L_r(\mathbf{p}, \mathbf{v}) = \langle \mathbf{b}^{v,T(p)}, \mathbf{M}_p(\mathbf{a}_i) \rangle \approx \sum_{j=0}^{c_M} \sum_{m=0}^{c_B} \alpha_{p,j} \beta_{T(p),m} \langle \mathbf{e}_{m,k_B(T(p))}^v, \mathbf{m}_{j,k_M(p)}(\mathbf{a}_i) \rangle \quad (3)$$

Equation 3 reduces the computation of the lighting integral for a fixed lighting environment to a simple weighted sum of dot products that can be precomputed independently from screen resolution and mesh-complexity. The dot products

$$\lambda_{m,k_B,j,k_M}^v(\mathbf{a}_i) = \langle \mathbf{e}_{m,k_B}^v, \mathbf{m}_{j,k_M}(\mathbf{a}_i) \rangle$$

must be recomputed only if the lighting vector \mathbf{a}_i changes. Of course the cost for this computation increases with the number of clusters and components since $|k_M|*(c_M+1)$ matrix-vector multiplications and $|k_B|*|k_M|*(c_B+1)*(c_M+1)*|v|$ dot products have to be computed for each color channel. But there is always the possibility to trade quality for speed simply by using fewer components. Thus we implemented a simple adaptive rendering scheme that is described in detail in the following section.

5. Real-Time Rendering

Rendering meshes covered with LPCA encoded BTFs in environments with arbitrary lighting (represented as environment maps) consists of two steps: precomputing the parameters for the current lighting situation on the CPU and rendering the mesh using these parameters.

5.1 Precomputation of PRT Values

The precomputation step has to compute the abovementioned dot-products and store them in textures in a format suitable for graphics hardware. We found parabolic hemispherical maps [7] to be a good choice since they provide an approximately equal sampling of the hemisphere and since values stored in this format can easily be interpolated by graphics hardware. That way, interpolation between sampled view-directions can be achieved. In all our tests a resolution of 16×16 texels led to satisfying results.

Since current graphics hardware prohibits linear interpolation of floating-point valued textures, we have to map the high-dynamic range PRT values λ_{m,k_B,j,k_M}^v to 8 bit integers. Since the values differ significantly but can roughly be categorized into four

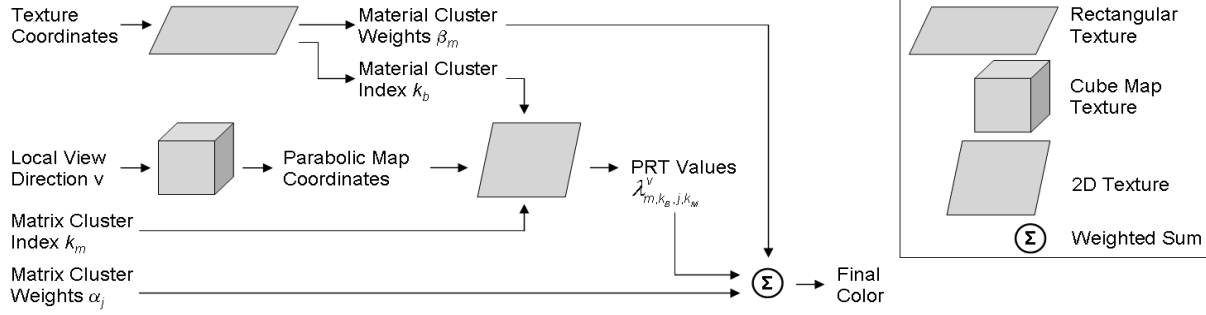


Figure 4. Rendering setup of the fragment shaders

categories (one category for $m=j=0$, a second for $m=0$ and $j \neq 0$, a third for $m \neq 0$ and $j=0$, and a fourth for $m \neq 0$ and $j \neq 0$), we compute four different scaling values, apply them to the PRT values before storage in a texture and send the inverse factors to the graphics board.

5.2 Rendering the Mesh

Rendering the mesh given the precomputed parameters for the current lighting situation is done using a setup similar to Sloan et al. [24]. The triangles of the mesh are sorted into bins according to the matrix clusters of the vertices of the triangles (i.e. whenever at least one vertex of the triangle belongs to the matrix cluster assigned to the bin, the triangle is inserted into the bin). For every triangle in every bin we specify three Boolean parameters indicating whether the vertices belong to the bin's cluster. At run-time the triangles in the bins are rendered independently. Taking into account the Boolean parameters, only vertices belonging to the bin's rotation cluster contribute to the partial image produced by the bin. The partial results are accumulated by setting the blending mode to add.

Other than in Sloan et al. [24] the vertex shader simply computes the local view direction and sets the matrix cluster weights to zero if the Boolean parameter of the vertex is set to false. The majority of computation is performed in the fragment shaders (see figure 4 for a schematic description). The inputs to the shader are the texture coordinates of the current fragment, the local view direction (which is interpolated from the local view directions at the vertices), the matrix cluster index and the interpolated matrix cluster weights. Based on the texture coordinates we lookup the material cluster index and the material cluster weights. The local view direction serves as index into a cube map that returns corresponding Parabolic Map coordinates. Based on these coordinates, the matrix and material cluster indices, we lookup PRT values from a 2D texture – one for each material and matrix cluster. The final color of the fragment is computed according to equation 3.

Please note that figure 4 omits the necessary scaling of the 8 bit PRT values.

5.3 Results

We implemented the rendering algorithm on an Intel Pentium IV 2.6 GHz processor with a GeForce FX 5900 graphics board using OpenGL.

Precomputing the PRT values whenever the light situation changes takes about 0.8 seconds for 32 matrix clusters, 8 matrix components, 16 BTF clusters and 4 BTF components – a high-quality setting that suffices for the tested models and materials. Since this number is much too high for interactive light changes, we choose a smaller numbers of components whenever the model is rotated in the environment (i.e. only 4 matrix and 1 BTF components). Since the preprocessing time decreases approximately linearly with the number of clusters and components, the precomputation time is reduced so much that even real-time navigation becomes possible. Although not implemented in our framework, this technique naturally applies to varying lighting environments (e.g. due to video textures) as well.

Rendering times for the meshes mainly depend on the screen projection size of the model and on the number of components (both matrix and BTF) that we employ. As you can see in the accompanying video, the frame rates for the high-quality setting are close to real-time (about 12 fps) even at high resolutions. These frame-rates can as well be achieved for models covered with multiple BTFs.

Figure 5 shows two models covered with LPCA encoded BTFs which are lit by image-based lighting. Please note the different colors on the Isis statue resulting from the colored lighting environment (i.e. the red-dish-brown light reflected from the table below the statue which lights the bottom right part of the statue's dress and the rather white, dominating light from the window which lights the remainder of the statue). Figure 6 compares the appearance of the Max Planck bust



Figure 5. On the left a shirt covered with corduroy is shown in the UBO environment. On the right an Isis statue covered with plasterstone is lit by the kitchen environment.

lit by several different lighting environments and demonstrates the effect of large-scale shadowing.

The memory requirements for the PRT values λ sum up to about 16.9 MB if 8 bit integer values are used for each RGB channel (which we found to yield pleasing results in our tests). This potentially enables use of multiple BTF materials in scenes lit by environmental lighting. Unfortunately, current driver versions still support automatic filtering of textures with width and size both being a power of two only although the `ARB_texture_non_power_of_two` [1] extension was specified already. We therefore have to use textures of size 4096×4096 that consume approximately 50 MB of texture memory. With upcoming driver versions this necessity will be removed.

6. Conclusions

In this paper we presented a method for efficient re-lighting of LPCA encoded BTFs. We exploited the SH-basis for fast evaluation of the lighting integral and proposed a rendering algorithm that computes substantial parts of a reformulated lighting integral independent from screen resolution and geometry size. The results of these computations can be reused to speed up rendering for fixed lighting. Fast lighting change is

supported by decreasing the quality of the reconstruction temporarily.

As future work, we will try to better and more flexibly balance precomputation and rendering time.

7. Acknowledgments

This work was partially funded by the European Union under project RealReflect (IST-2001-34744). Special thanks belong to Ralf Sarlette who measured the BTFs and the UBO HDR environment. We thank Paul Debevec for providing the other HDR environments.

8. References

- [1] `ARB_texture_non_power_of_two`. http://oss.sgi.com/projects/ogl-sample/registry/ARB/texture_non_power_of_two.txt
- [2] Chen, W., Bouguet, J., Chu, M., and Grzeszczuk, R., “Light field mapping: Efficient Representation and Hardware Rendering of Surface Light Fields”, In *SIGGRAPH 2002*, pp. 447-456, 2002
- [3] Dana, K., van Ginneken, B., Nayra, S., and Koenderink, J., “Reflectance and Texture of Real World Surfaces”, In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 151-157, 1997



Figure 6. From left to right: Plasterstone covered Max Planck bust lit by Galileo, UBO and the RNL environment. The rightmost image is without large-scale shadows.

- [4] Daubert, K., Lensch, H., Heidrich, W., and Seidel, H.-P., "Efficient Cloth Modeling and Rendering", In *12th Eurographics Workshop on Rendering*, pp. 63-70, 2001
- [5] Debevec, P., Hawkins, T., Tchou, C., Duiker, H., Sarokin, W., and Sagar, M., "Acquiring the reflectance field of a human face", In *SIGGRAPH 2000*, pp. 145-156, 2000
- [6] Furukawa, R., Kawasaki, H., Ikeuchi, K., and Sakauchi, M., "Appearance based object modeling using texture database: Acquisition, compression and rendering", In *13th Eurographics Workshop on Rendering*, 2002
- [7] Heidrich, W., and Seidel, H.-P., "Realistic, Hardware-Accelerated Shading and Lighting", In *SIGGRAPH 1999*, pp. 171-178, 1999
- [8] Kautz, J., and Seidel, H.-P., "Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs", In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 51-58, 2000
- [9] Kautz, J., Vázquez, P.-P., Heidrich, W., and Seidel, H.-P., "A Unified Approach to Prefiltered Environment Maps", In *11th Eurographics Workshop on Rendering*, pp. 185-196, 2000
- [10] Lafortune, E., Foo, S.-C., Torrance, K., and Greenberg, P., "Non-linear approximation of reflectance functions", In *SIGGRAPH 1997*, pp. 117-126, 1997
- [11] Latta, L., and Kolb, A., "Homomorphic Factorization of BRDF-based Lighting Computation", In *SIGGRAPH 2002*, pp. 509-516, 2002
- [12] Lensch, H., Goesele, M., Kautz, J., Heidrich, W., and Seidel, H.-P., "Image-Based Reconstruction of Spatially Varying Materials", In *12th Eurographics Workshop on Rendering*, pp. 103-114, 2001
- [13] Levoy, M., and Hanrahan, P., "Light Field Rendering", In *SIGGRAPH 1996*, pp. 31-42, 1996
- [14] Matusik, W., Pfister, H., Ngan, A., Ziegler, R., and McMillan, L., "Acquisition and Rendering of Transparent and Refractive Objects", In *13th Eurographics Workshop on Rendering*, pp. 267-278, 2002
- [15] McAllister, D., Lastra, A., and Heidrich, W., "Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions", In *Graphics Hardware 2002*, pp. 78-88, 2002
- [16] Meseth, J., Müller, G., and Klein, R., "Reflectance Field Based Real-Time, High-Quality Rendering of Bidirectional Texture Functions", In *Computers & Graphics*, Volume 28, pp. 105-112, February 2004
- [17] Malzbender, T., Gelb, D., and Wolters, H., "Polynomial texture maps" In *SIGGRAPH 2001*, pp. 519-528, 2001
- [18] Miller, G., Rubin, S., and Ponceleon, D., "Lazy Decompression of Surface Light Fields for Precomputed Global Illumination", In *9th EGWR*, pp. 281-292, 1998
- [19] Müller, G., Meseth, J., and Klein, R., "Compression and Real-Time Rendering of Measured BTfFs using Local PCA", In *Vision, Modeling and Visualization*, pp. 271-280, 2003
- [20] Ramamoorthi, R., and Hanrahan, P., "An Efficient Representation for Irradiance Environment Maps", In *SIGGRAPH 2001*, pp. 497-500, 2001
- [21] Ramamoorthi, R., and Hanrahan, P., "Frequency Space Environment Map Rendering", In *SIGGRAPH 2002*, pp. 517-526, 2001
- [22] Sattler, M., Sarlette, R., and Klein, R., "Efficient and Realistic Visualization of Cloth", In *Eurographics Symposium on Rendering*, 2003
- [23] Sloan, P.-P., Kautz, J., Snyder, J., "Precomputed Radiance Transfer for real-time rendering in dynamic, low-frequency lighting environments", In *SIGGRAPH 2002*, pp. 527-536, 2002
- [24] Sloan, P.-P., Hall, J., Hart, J., and Snyder, J., "Clustered Principal Components for Precomputed Radiance Transfer", In *SIGGRAPH 2003*, pp. 382-391, 2003
- [25] Sloan, P.-P., Liu, X., Shum, H.-Y., and Snyder, J., "Bi-Scale Radiance Transfer", In *SIGGRAPH 2003*, pp. 370-375, 2003
- [26] Suykens, F., vom Berge, K., Lagae, A. and Dutré, P., "Interactive rendering of bidirectional texture functions", In *Eurographics 2003*, pp. 463-472, 2003
- [27] Wong, T.-T., Heng, P.-A., Or, S.-H. and Ng, W.-Y., "Image-based Rendering with controllable illumination", In *EGWR 1997*, pp. 13-22, 1997
- [28] Wood, D., Azuma, D., Aldinger, K., Curless, B., Duchamp, T., Salesin, D., and Stuetzle, W., "Surface Light Fields for 3D Photography", In *SIGGRAPH 2000*, pp. 287-296, 2000