Reflectance Field based real-time, high quality Rendering of Bidirectional Texture Functions

Jan Meseth, Gero Müller, Reinhard Klein

Department of Computer Science II, Bonn University, Römerstraße 164, 53117 Bonn, Germany

Abstract

The Bidirectional Texture Function (BTF) is a suitable representation for the appearance of highly detailed surface structures under varying illumination and viewing conditions. Since real-time rendering of the full BTF data is currently not feasible, approximations of the six-dimensional BTF are used such that the amount of data is reduced and current graphics hardware can be exploited. While existing methods work well for materials with low depth variation, realism is lost if the depth variation grows. In this article we analyze this problem and devise a new real-time rendering paradigm based on linear interpolation of reflection fields, which provides significant improvements with respect to realism for such highly structured materials without sacrificing the general applicability and speed of previous algorithms. We propose real-time rendering algorithms for this new method supporting either point light sources or image-based lighting and demonstrate the capabilities of our new approach with several examples.

Key words: Color, shading, shadowing, texture

1. Introduction

Realistic rendering of real-world objects incorporates complex geometric models and sophisticated modelling of the object's surface reflectance behavior. In the field of real-time rendering, for many years the latter task has been covered by the well-known Phong-model[27] because of its simplicity and computational efficiency. The (lambertian) diffuse term of the model was allowed to vary spatially via texture mapping. Due to the dramatic development of rendering hardware in the last few years, it became possible to render surfaces using more enhanced and physically plausible approximations like the Lafortune[14] or the Ashikhmin[1] model and even arbitrary bi-directional reflectance distribution functions (BRDFs)[10] in real-time. Since the BRDF captures only the physical reflectance behavior at a microscopic scale, Dana et al.[4] introduced the bidirectional texture function (BTF) which can roughly be interpreted as a tabulated BRDF-per-texel representation but which additionally includes local effects like self-shadowing, self-occlusion and subsurface scattering. Due to the sheer size of a BTF (hundreds of megabytes), real-time rendering of the full data is currently not feasible. Hence approximations of the 6D-BTF are used.

Existing approaches to real-time BTF rendering assume common BRDF models on a per-texel basis. Implementing low-parameter but expressive models, this results in an extraordinary data compression as recently shown by McAllister et al.[21]. Although this kind of approach seems to work well for materials with low-depth variation, it leads to unsatisfying results if the depth variation grows. In fact, one experiences a significant loss of 3D-structure of the surface and therefore a loss of realism in the visualization of highly structured surfaces (compare figure 5).

As main contributions of this paper, we first provide an *in-depth analysis* of fitting existing reflectance functions

Email address: meseth@cs.uni-bonn.de (Jan Meseth).

to BTF datasets. Based on the results and further observations, we devise implementing *BTF rendering as linear interpolation of reflectance fields*. We test our approach for two different reflectance field approximation functions and show that this approach results in drastically reduced fitting errors and thus significantly improves the visual quality of rendered images, at the expense of higher texture memory requirements. In addition, we propose *real-time rendering algorithms* that support either point- and directional light sources or image-based lighting, and describe the *integration of BTF rendering into OpenSG*.

The rest of the paper is organized as follows: After reviewing related work in section 2 we analyze the BTFs of highly depth-varying surfaces in greater detail in section 3. In section 4 we present our new BTF approximation. In section 5 our hardware-accelerated rendering algorithms and their integration into OpenSG are discussed and some results are presented. Finally, we conclude and describe directions for future research in section 6.

2. Related Work

Truly realistic renderings of real world materials have to simulate the physics of light and reflection for every surface point. Such an approach is infeasible given today's computing power and will likely remain in the near future.

Although simple texture and bump map representations lead to impressive results for very simple materials, more complex models are required to simulate the real appearance of natural materials. Early results approximated a single BRDF by a Ward[32] or Lafortune[14] model. Kautz and McCool[10] approximate the four-dimensional BRDF by a product of two two-dimensional functions which are stored as textures and combined during the rendering step. McCool et al.[22] improved the above method by employing homomorphic factorization, leading to approximations with user-controllable quality features. The above approaches were further improved by [28,30,16,17,31] which all enable the BRDF to be lit by image-based lighting while relying on different approximation functions.

In the context of rendering spatially varying materials, Debevec et al.[6] presented a method for reflectance field rendering (fixed view, varying light). They acquired and relighted human faces exploiting a human skin reflectance model. Malzbender et al.[20] compressed the reflectance fields of each texel of a measured material by fitting polynomials of low degree. Ashikhmin and Shirley[2] used basis textures lit by a steerable light basis for relighting. Levoy and Hanrahan[19] and Gortler et al.[8] simultaneously introduced light field rendering (fixed light, varying view). Miller et al.[24] parameterized light fields over surfaces introducing surface light fields and employed JPEG-like compression for the images. Following publications[34,3] concentrated on the application of different data compression schemes.

BTF-rendering can be understood as rendering of spatially varying materials under varying light and view conditions. Due to the enormous amount of data in a BTF, only few real-time rendering algorithms have been published so far. A foundation was set by Kautz and Seidel[12] since they introduced techniques for evaluating spatially varying BRDFs on graphics hardware. They factor the BRDFs - given as factors of simple reflectance models - into two-dimensional functions and store the values in textures that are evaluated with hardware supported operations and dependent texture lookups. McAllister et al.[21] published a method based on these techniques that approximates the BTF by pixelwise Lafortune models, which can efficiently be evaluated in current graphics hardware. One year earlier already, Daubert et al.[5] 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. In an approach similar to [30] Kautz et al.[13] rendered spatially varying BRDFs by simply employing higher-dimensional look-up tables.

3. BTFs of Highly Depth-Varying Surfaces

The BTF can be defined as RGB-texture that varies with light and view direction. In this work we employ a high-quality sampling of this function consisting of 256x256 texels in size and 81x81 poses for light and viewing direction leading to more than 1.2GB of data (consider [29] for details on the measurement procedure). Even with today's most powerful graphics hardware real-time BTF rendering via linear interpolation of this data is rather intractable. Thus, some kind of lossy compression has to be used.

If the per-texel data is assumed to exhibit a BRDF-like behavior, one can simply apply every known BRDF model/approximation (e.g. spherical harmonics[33], spherical wavelets[15] or analytical models like generalized cosine lobes[14,1] to name a few) to each texel independently. The demand of real-time rendering rules out linear-basis decompositions, since they tend to use too many coefficients in modelling specularities. Unfortunately, analytical models are not always suitable either, especially in the case of rough and highly depth varying surfaces where the per-texel data is strongly influenced by the surface points' neighborhood. For such surfaces the data measured at a single texel exhibits many asymmetric effects which reciprocal BRDF-models are not able to reconstruct. Figure 1 illustrates some of these effects. Note the roughness of the data and the several dark stripes for fixed view direction leading to significant asymmetry.

As also shown in figure 1 for the Lafortune-model such effects can not be reconstructed by reciprocal BRDF-models. The omission of such effects during rendering leads to significant loss of depth impression as illustrated in figure 5.

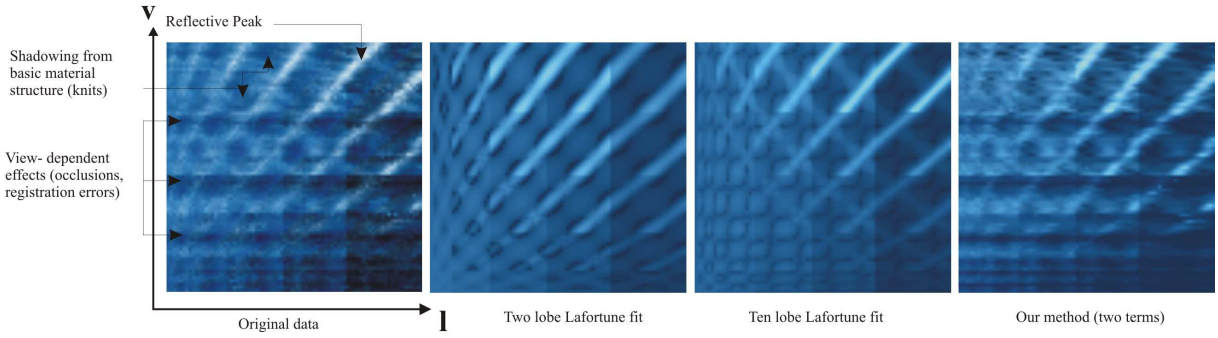

Fig. 1. On the left the measured data for one texel of the blue knitted wool dataset is shown ($\mathbf{l}$ denotes light and $\mathbf{v}$ view direction). Note, that the foreshortening term $\mathbf{n} \cdot \mathbf{l}$ is included. In the middle two corresponding Lafortune fits with two and ten lobes are depicted. These fits reconstruct only basic features of the material and can not capture the asymmetric parts of the data. A fit with our method is drawn on the right. More details and asymmetric features are preserved.

Even if we increase the complexity of the model (by adding more lobes in this case) the result is only slightly improved.

A practical problem in using analytical BRDF models like the Lafortune or Ward model arises from the fact, that the fit to the measured data requires costly non-linear optimization by employing for example a Levenberg-Marquardt algorithm. The results of such an optimization heavily depend on a good initialization and the running times increase super-linear in the number of parameters. For example the fits depicted in figure 1 took 8 seconds for the two-lobe and 143 seconds for the ten-lobe fit on a 3GHz Pentium 4 employing the LINPACK-implementation of the Levenberg-Marquardt algorithm. Hence fitting a ten lobe Lafortune model to every texel of a 256x256-sized BTF on a single PC would take about 100 days. The result in figure 1 generated by our method took only 2 seconds computation time. Besides leading to impractical fitting times increasing the number of components also hampers real-time rendering, since all these components have to be combined per pixel during rendering.

Since neither linear interpolation of measured values nor fitting of simple BRDF-style models can achieve both high quality and real-time rendering, we propose in the following a combination of the memory inefficient but high quality linear interpolation strategy for rendering the full BTF data and the efficient yet low-quality fitting strategy.

3.1. *4D BTF-Slices*

Fixing the incident direction $\mathbf{l}$ of a BTF, we arrive at a 4D function called the *surface light field*:

$$LF_{\mathbf{l}}(\mathbf{x}, \mathbf{v}) := BTF(\mathbf{x}, \mathbf{l}, \mathbf{v})$$

Otherwise, fixing exitant direction $\mathbf{v}$, the resulting function is the patch's *surface reflectance field*:

$$RF_{\mathbf{v}}(\mathbf{x}, \mathbf{l}) := BTF(\mathbf{x}, \mathbf{l}, \mathbf{v})$$

Now we propose to implement BTF rendering as mapping and rendering a discrete set of discrete light fields (LFs) or reflectance fields (RFs) of the measured surface onto arbitrary geometry. Employing either LFs or SFs, the color of a BTF-textured surface element with texture coordinate $\mathbf{x}$ given local light and view direction $(\mathbf{l}, \mathbf{v})$ can be computed as follows:

- Approximate the BTF by a set of independently fit LFs/RFs.
- Compute the color of $\mathbf{x}$ according to every fitted LF/RF and interpolate the final color from the partial results.

4. Reflectance Field BTF Approximation

During our research, we tested both RF and LF approximation and found that the surface reflectance field leads to better approximations in our approach since the RF turned out to be generally smoother than the LF for the samples we tested. Refer to figure 1 where the view-dependent asymmetry leading to discontinuities for fixed light is nicely illustrated.

Therefore the RF is better suited for fitting by compact functional representations (e.g. polynomials) than the LF. The discontinuous view-dependence will be preserved by our approach, since the piecewise linear function as induced by the linear interpolation captures this high-frequency content of the data.

4.1. *A Non-Linear RF Approximation*

An implementation of our approach can now be obtained by applying view interpolation and a suitable RF approximation which should be efficiently renderable on

today's consumer graphics hardware and minimize the approximation error. Possible candidates that we tested are biquadratic polynomials as in [20] and the following non-linear function:

$$RF_{\mathbf{v}}(\mathbf{x},\mathbf{l}) \approx \rho_d(\mathbf{x}) + \rho_{s,\mathbf{v}}(\mathbf{x}) \sum_{j=1}^{k} s_{\mathbf{v},j}(\mathbf{x},\mathbf{l})$$
$$= \rho_d(\mathbf{x}) + \rho_{s,\mathbf{v}}(\mathbf{x}) \sum_{j=1}^{k} (t_{\mathbf{v},j}(\mathbf{x}) \cdot \mathbf{l})^{n_{\mathbf{v},j}(\mathbf{x})} \quad (1)$$

with $t_{\mathbf{v},j}(\mathbf{x})$ being a three dimensional vector and $s_{\mathbf{v},j}(\mathbf{x},\mathbf{l})$ similar to a Lafortune lobe discarding the exitant direction, ρ_d and ρ_s denoting diffuse and specular albedo. This model is well suited for fitting specularities and directional diffuse lobes. The parameter k controls the number of lobes. Since we apply the function for luminance data only, the final color is computed as in [20]. A Levenberg-Marquardt algorithm is used for the fitting[14,21,18]. Convergence is improved via detecting principal directions with a high-pass filter and using the recovered directions as an initialization for the optimization.

4.2. *Results*

In order to measure the approximation quality of our model quantitatively we compared the average reconstruction error per texel $\mathbf{x}$ given as follows:

$$\epsilon^{M}_{\mathrm{BTF}}(\mathbf{x}) = \sum_{(\mathbf{v},\mathbf{l})\in\Delta} \frac{|\mathrm{BTF}(\mathbf{x},\mathbf{v},\mathbf{l}) - M(\mathbf{x},\mathbf{v},\mathbf{l})|}{|\Delta|} \quad (2)$$

Here, Δ denotes the set of discrete measured view und light directions. M denotes the corresponding BTF-approximation. We compared our model with the Lafortune model for four lobes (higher numbers did not lead to significantly better results but increased the fitting times drastically as mentioned above), and the more sophisticated, asymmetric model of Daubert et al.[5].

Note, that the polynomials work well for mainly diffuse materials like *proposte* or *corduroy*. For materials with strong specularities like *stone* the non-linear modeling is favorable since the polynomials tend to blur specularities. As expected, our method is especially suited for depth varying materials like *corduroy* while flat materials like *aluminium* are also well approximated by simpler models.

5. Real-Time Rendering

The task of the real-time rendering algorithm is to evaluate, for each surface point $\mathbf{x}$, the following formula:

		LAF	SLAF	RFP	RFNL
Proposte	avg	0.0978	0.0794	0.0689	0.0736
	min	0.0645	0.0537	0.0514	0.0490
	max	0.1149	0.1003	0.0829	0.0924
Knitted Wool	avg	0.0788	0.0693	0.0719	0.0582
	min	0.0570	0.0527	0.0592	0.0459
	max	0.1056	0.0875	0.0857	0.0729
Stone	avg	0.0904	0.0816	0.0797	0.0640
	min	0.0368	0.0345	0.0523	0.0284
	max	0.1847	0.1767	0.1335	0.1321
Corduroy	avg	0.1114	0.0859	0.0513	0.0537
	min	0.1003	0.0761	0.0416	0.0437
	max	0.1223	0.0948	0.0604	0.0639
Aluminium	avg	0.0572	0.0558	0.0846	0.0479
	min	0.0391	0.0389	0.0773	0.0324
	max	0.0935	0.0889	0.0985	0.0782

Table 1
ϵ for some materials and the Lafortune model with four lobes (LAF) and additional look-up table (SLAF), the RF approximation using biquadratic polynomials (RFP) and using two non-linear terms (RFNL).

$$L_r(\mathbf{x},\mathbf{v}) = \int_{\Omega_i} f_{r,x}(\tilde{\mathbf{v}},\tilde{\mathbf{l}}) L_i(\tilde{\mathbf{l}})(\mathbf{n}\cdot\tilde{\mathbf{l}})\, d\tilde{\mathbf{l}} \quad (3)$$

where $f_{r,x}$ is the BRDF in point $\mathbf{x}$, L_i is the incoming radiance, Ω_i is the incident hemisphere over the surface point $\mathbf{x}$, $\mathbf{n}$ is the surface normal, $\tilde{\mathbf{l}}$ and $\tilde{\mathbf{v}}$ represent local light and view direction, and L_r is the exitant radiance.

5.1. *Scenes containing Simple Light Sources*

Many typical applications using computer graphics use a finite number of point light sources only since these are supported by the common graphics APIs. For these cases, the integral in the above rendering equation reduces to a sum. Substituting $f_{r,x}$ by a suitable RF approximation and interpolating the view direction, we obtain the equation:

$$L_r(\mathbf{x},\mathbf{v}) = \sum_{\tilde{\mathbf{l}}\in L} \left(\sum_{v\in N(\tilde{\mathbf{v}})} w_v(\tilde{\mathbf{v}}) RF_v(\mathbf{x},\tilde{\mathbf{l}}) \right) L_i(\tilde{\mathbf{l}})(\mathbf{n}\cdot\tilde{\mathbf{l}})$$
$$= \sum_{v\in N(\tilde{\mathbf{v}})} w_v(\tilde{\mathbf{v}}) \sum_{\tilde{\mathbf{l}}\in L} \left(RF_v(\mathbf{x},\tilde{\mathbf{l}}) L_i(\tilde{\mathbf{l}})(\mathbf{n}\cdot\tilde{\mathbf{l}}) \right) \quad (4)$$

where L represents the set of light sources, $N(\tilde{\mathbf{v}})$ denotes the index set of view directions from the measured BTF data that are neighboring $\tilde{\mathbf{v}}$, and w_v denotes the weight for the reflectance field RF_v.

The rendering process that evaluates the rendering equation is depicted in figure 2. 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.

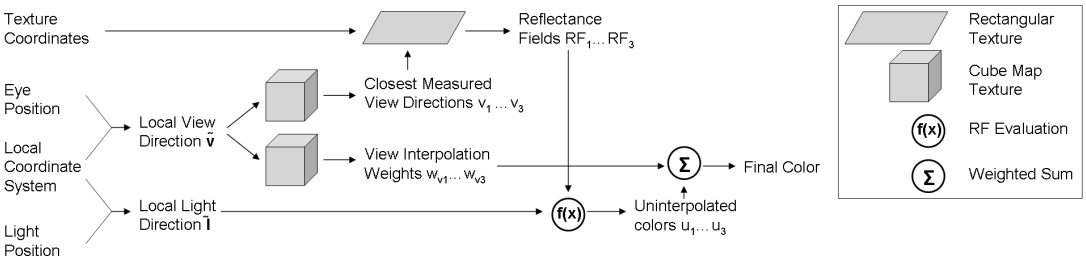

Fig. 2. Data flow of the rendering algorithm for scenes containing simple light sources only.

We first compute view and light directions and transform them into the pixel's coordinate system. Using cube maps, we lookup the indices to the three closest view-directions from the measurement process, together with their interpolation weights. Combining the current texture coordinates and the index to the closest view direction, we lookup the parameters for the respective RF which are stored in a rectangular texture. We evaluate the pixel's color according to the chosen RF approximation. If interpolation is used, this evaluation is repeated for the two other closest view directions. The unweighted colors u_i are multiplied with the respective interpolation weights w_i and summed to form the final color of the pixel.

5.2. *Scenes lit by Image-Based Lighting*

While simple light sources yield sufficient results in many cases, they cannot represent natural lighting as e.g. in outdoor scenes. This is far better accomplished by Image-Based Lighting techniques[7]. Previous methods combining reflection models and image-based lighting (e.g. [9,11,21]) are commonly based on two simplifying assumptions. First, the lighting environment is assumed to be at an infinite distance from the lit object. Second, for non-diffuse reflections the Lambertian cosine term accounting for the area foreshortening is approximated[11]. With these assumptions, analogously to McAllister et al.[21], the rendering equation for our non-linear RF approximation changes to:

$$L_r(\mathbf{x},\mathbf{v}) \approx \rho_d(\mathbf{x}) D(\mathbf{n}) + \sum_{v \in N(\tilde{\mathbf{v}})} w_v(\tilde{\mathbf{v}}) \rho_{s,v}(\mathbf{x}) \cdot \sum_{j=1}^{k} S\left(\frac{t_{v,j}(\mathbf{x})}{\|t_{v,j}(\mathbf{x})\|}, n_{v,j}(\mathbf{x})\right) \|t_{v,j}(\mathbf{x})\|^{n_{v,j}(\mathbf{x})} \left(\mathbf{n} \cdot t_{v,j}(\mathbf{x})\right)$$

where $D(\mathbf{n})$ and $S(\mathbf{t}, n)$ denote diffuse and specular prefiltered environment maps. Please note that in this case the global rather than the local light direction is used. Please note as well that in this case $t_{v,j}$ (from equation 1) denotes a vector concerning the global coordinate system. Other than in [21] our rendering method requires view interpolation of the specular contribution.

In addition to the data used for scenes with simple light sources only, we employ a cube map that stores for every texel the coordinates for a parabolic map[9]. A rectangular texture stores the prefiltered lighting environments for various sampled exponents n (here: $n = 2^{\frac{i}{2}}$ for $i = 0 \ldots 10$).

The rendering process is depicted in figure 3. The inputs are standard texture coordinates, the eye's position, a per-pixel coordinate system and the inverse transformation matrix – both interpolated from the matrices at the vertices which are again specified with the geometry.

Computing the local view direction, determining closest measured view directions and interpolation weights, and fetching the RF parameters is analogous to the rendering algorithm for simple light sources. For the evaluation of the rendering algorithm, we first transform each lobe's direction $t_{\mathbf{v},j}(\mathbf{x})$ into the global coordinate system (in which the lighting environment is defined as well). The resulting directions $\mathbf{t}$ are used to first determine parabolic map texture coordinates $\mathbf{p}$ which - combined with the rounded lobe exponent - serve as indices to lookup the incoming radiance L from the prefiltered environment maps.

Finally, the incoming radiances for the various lobes are scaled according to the exponentiated lengths of the respective lobes, weighted by the approximated foreshortening term and summed. View interpolation is again the same as for the case of simple light sources.

5.3. *Results*

Figure 4 shows a comparison between RF based BTF rendering and simple bump-mapped texturing. The BTF textured version captures the look-and-feel of the corduroy material far better. The seat model renders at about 21 fps on an NVidia Geforce FX 5800 graphics board using our OpenGL implementation. Figure 6 shows a shirt model lit by natural, image-based lighting. The rendering speed is somewhat slower (about 14 fps). Please note that the frame rates are largely independent of the complexity of the rendered model since our algorithm is mainly fill-rate-limited.

During rendering of the non-linear RF approximation, we sometimes experienced disturbing white pixels, which result from the fitting procedure: the computed lobes may

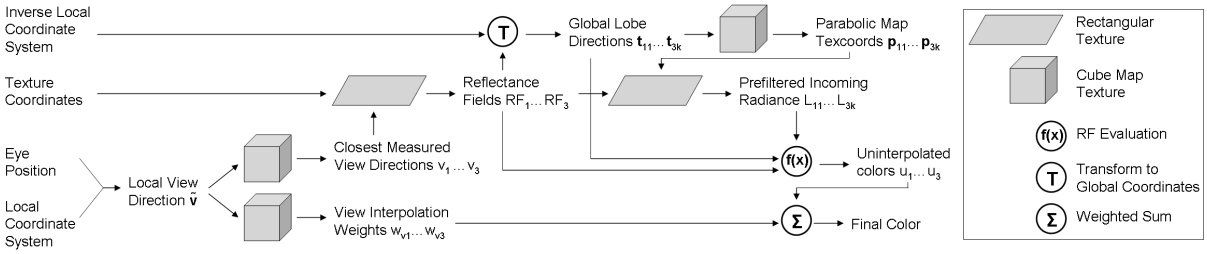

Fig. 3. Data flow of the rendering algorithm for scenes lit by image-based lighting.

lead to floating point overflows in the graphics hardware during rendering, which is partially due to the specific graphics hardware and partially to rather high exponents. Using 16 bit instead of 32 bit floating point values, these problems occurred more frequently.

Another problem we needed to solve is memory consumption. Due to the large number of parameters per material texel in our model and their representation as floating-point numbers, our rendering method requires about 365 MB for a 256×256 BTF with 81 reflectance fields if 32 bit floating-point values are used. This large amount of memory can either be reduced using texture synthesis methods as done by Meseth et al.[23] (which additionally solves the problem of texturing large model regions but remains limited to rather small base textures) or using clustering approaches like Müller et al.[25].

5.4. *Rendering within OpenSG*

In order to make BTF rendering available for virtual reality applications, we chose to integrate it into the open source scene graph system OpenSG[26].

The basic control element for rendering in OpenSG is the Chunk, which is used to control the OpenGL state. Especially useful for our implementation are the subclasses TextureChunk, CubeTextureChunk, VertexProgramChunk and FragmentProgramChunk. The first two ones are used to encapsulate the various texture formats offered by OpenGL (2D, 3D, cube) whereas the second two encapsulate vertex- and fragment-programs.

Since our BTF rendering algorithms require per pixel parameters to be stored as floating-point values in order to achieve sufficient quality, a few new classes need to be implemented that are currently not supported by OpenSG.

OpenGL offers high precision by providing 16 bit and 32 bit floating point values to be stored in textures. In OpenSG, textures are rather tightly bound to the Image class, which unfortunately does not support floating point formats explicitely so far. Therefore, either the Image class needs to be extended or a new class FloatImage needs to be implemented. Additionally, loaders have to implemented for the floating point file formats in which the BTF data is stored. Finally, a new texture class has to be implemented that encapsulates rectangular textures. These textures allow more efficient storage by allowing arbitrary heights and widths and - for NVidia based graphics boards - are the only way to access floating point valued textures.

The higher level primitive which is used to finally encapsulate the BTF rendering is the Material. An implementation should use an instance of the ChunkMaterial class that gets handed the correct textures, fragment- and vertex-programs as Chunks at initialization time.

6. Conclusions

In this work, we presented an in-depth analysis of the real-time rendering problem for highly depth varying BTF materials. We demonstrated why existing algorithms have problems with such materials and proposed a new method that achieves high-quality results at the expense of consuming more texture memory than existing real-time methods.

The rather poor frame rates that we experienced are apparently largely due to the OpenGL driver since it appears not to be optimized to handle large amounts of texture memory (especially if floating-point valued textures are employed). We expect these problems to be resolved by future driver versions. In addition, newer generation graphics boards already feature higher transfer rates and will likely be equipped with bigger and more efficient texture caches in the future which would both speed up our rendering algorithms.

For future work, we will simultaneously try to further reduce the amount of memory required by our method and increase its approximation quality, implement mip-mapping techniques and fully integrate our method into OpenSG.

Acknowledgements

This work was partially funded by the European Union under the project RealReflect (IST-2001-34744). We want to thank André Nicoll for helping with the implementation, Mirko Sattler for fruitful discussions and Paul Debevec for the HDR environments. Special thanks belong to Ralf Sarlette who provided the BTF measurements.

References

[1] Ashikhmin M, Shirley P. An Anisotropic Phong BRDF Model. Journal of Graphics Tools 2000;**5**(2): 25–32.

[2] Ashikhmin M, Shirley P. Steerable Illumination Textures. ACM Transactions on Graphics 2002;**21**(1): 1–19.

[3] Chen W-C, Bouguet J-Y, Chu MH, Grzeszczuk R. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. Proceedings of SIGGRAPH 2002, 2002. p. 447–56.

[4] Dana KJ, van Ginneken B, Nayra SK, Koenderink JJ. Reflectance and Texture of Real World Surfaces. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997. p. 151–57.

[5] Daubert K, Lensch H, Heidrich W, Seidel H-P. Efficient Cloth Modeling and Rendering. Proceedings of 12th Eurographics Workshop on Rendering, 2001. p. 63–70.

[6] Debevec P, Hawkins T, Tchou C, Duiker H-P, Sarokin W, Sagar M. Acquiring the Reflectance Field of a Human Face. Proceedings of SIGGRAPH 2000, 2000. p. 145–56.

[7] Debevec P, Lemmon D. Image-Based Lighting. SIGGRAPH 2001 Course notes, 2001.

[8] Gortler S, Grzeszczuk R, Szeliski R, Cohen M. The Lumigraph. Proceedings of SIGGRAPH 1996, 1996. p. 43–54.

[9] Heidrich W, Seidel H-P. Realistic, Hardware-Accelerated Shading and Lighting. Proceedings of SIGGRAPH 1999, 1999. p. 171–78.

[10] Kautz J, McCool MD. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. Proceedings of Tenth Eurographics Workshop on Rendering, 1999. p. 281–92.

[11] Kautz J, McCool MD. Approximation of Glossy Reflection with Prefiltered Environment Maps. Proceedings of Graphics Interface 2000, 2000. p. 119–26.

[12] Kautz J, Seidel H-P. Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. Proceedings of Graphics Hardware 2000, 2000. p. 51–58.

[13] Kautz J, Sloan P-P, Snyder J. Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonic. Proceedings of 13th Eurographics Workshop on Rendering, 2002. p. 301–08.

[14] Lafortune EPF, Foo S-C, Torrance KE, Greenberg DP. Non-linear Approximation of Reflectance Functions. Proceedings of SIGGRAPH 1997, 1997. p. 117–26.

[15] Lalonde P, Fournier A. A Wavelet Representation of Reflectance Functions. IEEE Transactions on Visualization and Computer Graphics 1997;**3**(4):329–36.

[16] Latta L, Kolb A. Homomorphic Factorization of BRDF-based Lighting Computation. Proceedings of SIGGRAPH 2002, 2002. p. 509–16.

[17] Lehtinen J, Kautz J. Matrix Radiance Transfer. Proceedings of Symposium on Interactive 3D Graphics, 2003. p. 56–64.

[18] Lensch H, Goesele M, Kautz J, Heidrich W, Seidel H-P. Image-Based Reconstruction of Spatially Varying Materials. Proceedings of 12th Eurographics Workshop on Rendering, 2001. p. 103–14.

[19] Levoy M, Hanrahan P. Light Field Rendering. Proceedings of SIGGRAPH 1996, 1996. p. 31–42.

[20] Malzbender T, Gelb D, Wolters H. Polynomial Texture Maps. Proceedings of SIGGRAPH 2001, 2001. p. 519–28.

[21] McAllister DK, Lastra A, Heidrich W. Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. Proceedings of Graphics Hardware 2002, 2002. p. 78–88.

[22] McCool
MD, Ang J, Ahmad A. Homomorphic Factorization of BRDFs for High-Performance Rendering. Proceedings of SIGGRAPH 2001, 2001. p. 171–78.

[23] Meseth J, Müller G, Klein R. Preserving Realism in Real-Time Rendering of Bidirectional Texture Functions. Proceedings of OpenSG Symposium 2003, 2003. p. 89–96.

[24] Miller G, Rubin S, Ponceleon D. Lazy Decompression of Surface Light Fields for Precomputed Global Illumination. Proceedings of 9th Eurographics Workshop on Rendering, 1998. p. 281–92.

[25] Müller G, Meseth J, Klein R. Compression and real-time Rendering of measured BTFs using local PCA. To appear in Vision, Modeling, and Visualization 2003

[26] www.opensg.org

[27] Phong BT. Illumination for Computer Generated Pictures. Communications of the ACM 1975;**18**(6): 311–17.

[28] Ramamoorthi R, Hanrahan P. Frequency Space Environment Map Rendering. Proceedings of SIGGRAPH 2002, 2002. p. 517–26.

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

[30] Sloan P-P, Kautz J, Snyder J. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. Proceedings of SIGGRAPH 2002, 2002. pp. 527–36.

[31] Sloan P-P, Hall J, Hart J, Snyder J. Clustered Principal Components for Precomputed Radiance Transfer. Proceedings of SIGGRAPH 2003, 2003. p. 382–91.

Fig. 4. Comparison of our rendering technique (left) with approximated bump-mapping (right). The same light configurations were used in both pictures. Using our technique, the 3D structure of the corduroy material on the car seat appears realistic, while bump-mapping clearly misses the highlights for grazing light angles.

Fig. 5. Comparison of BTF rendering approaches. While the 2-lobe Lafortune model (left) fails to generate depth impression, the asynchronous model of Daubert (middle) achieves more realistic results. With out method (right), the depth impression is maximized.

[32] Ward GJ. Measuring and Modeling Anisotropic Reflection. Proceedings of SIGGRAPH 1992, 1992. p. 265–72.

[33] Westin SH, Arvo JR, Torrance KE. Predicting Reflectance Functions from Complex Surfaces. Proceedings of SIGGRAPH 1992, 1992. p. 255–64.

[34] Wood DN, Azuma DI, Aldinger K, Curless B, Duchamp T, Salesin DH, Stuetzle W. Surface Light Fields for 3D Photography. Proceedings of SIGGRAPH 2000, 2000. p. 287–96.

Fig. 6. Example of Image-Based Lighting using our non-linear RF approximation. Top: A shirt covered with Aluminium in the Galileo environment. Bottom left and right: diffuse and specular part of appearance.