COMPUTER GRAPHICS forum

DOI: 10.1111/j.1467-8659.2009.01585.x

COMPUTER GRAPHICS forum
Volume 29 (2010), number 1 pp. 175–190

Bidirectional Texture Function Compression Based on Multi-Level Vector Quantization

V. Havran[1], J. Filip[2] and K. Myszkowski[3]

[1]Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
[2]Institute of Information Theory and Automation of the AS CR, Czech Republic
[3]MPI Informatik, Saarbrücken, Germany

Abstract

The Bidirectional Texture Function (BTF) is becoming widely used for accurate representation of real-world material appearance. In this paper a novel BTF compression model is proposed. The model resamples input BTF data into a parametrization, allowing decomposition of individual view and illumination dependent texels into a set of multi-dimensional conditional probability density functions. These functions are compressed in turn using a novel multi-level vector quantization algorithm. The result of this algorithm is a set of index and scale code-books for individual dimensions. BTF reconstruction from the model is then based on fast chained indexing into the nested stored code-books. In the proposed model, luminance and chromaticity are treated separately to achieve further compression. The proposed model achieves low distortion and compression ratios 1:233–1:2040, depending on BTF sample variability. These results compare well with several other BTF compression methods with predefined compression ratios, usually smaller than 1:200. We carried out a psychophysical experiment comparing our method with LPCA method. BTF synthesis from the model was implemented on a standard GPU, yielded interactive framerates. The proposed method allows the fast importance sampling required by eye-path tracing algorithms in image synthesis.

Keywords: bidirectional texture function, BRDF, compression, SSIM

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading, texture I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Quantization, Reflectance

1. Introduction

Realistic visualization of surface appearance has been one of the main challenges in computer graphics since the early eighties. Nowadays, with constantly increasing capabilities of graphics hardware, increasing attention is paid in almost all industrial sectors to such applications as computer visual safety simulations and computer aided material design. All these applications require realistic reproduction of material behaviour under complex illumination and viewing conditions.

One method to capture real material appearance is based on the measurement of reflectance with respect to varying light and viewing directions. This so called 'Bidirectional Reflectance Distribution Function' (BRDF) was first described in [NJH*77]. BRDF has been compressed and approximated by a variety of empirical and analytical models in the past [LGC*05]. The BRDF itself does not preserve texture information, so it is suitable only for homogeneous materials. However, a large number of real, rough surfaces have a complicated spatial structure that causes effects such as shadowing, masking, inter-reflection, and subsurface scattering, all of which vary with illumination and viewing directions.

To preserve at least some of these effects, a new representation of real-world materials, the 'Bidirectional Texture Function' (BTF), was presented in [DvGNK99]. A monospectral BTF is a six-dimensional (6D) function which, unlike BRDF, accounts for the dependence of viewing and illumination

Figure 1: *Example images rendered by the proposed BTF compression methods for illumination by point light and by different environment maps.*

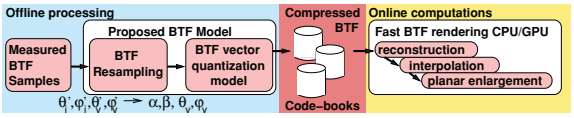


Figure 2: *BTF data processing pipeline.*

measurements on planar material position. An appropriately measured BTF contains information about material properties as anisotropy, masking, or self-shadowing. Examples of rendered images using BTF are depicted in Figure 1.

In contrast to BRDFs, BTF requires a very large amount of data storage. BTFs take up to several giga-bytes per sample in raw format. The storage space requirements of raw BTF data prevents their direct use for fast rendering in modern graphics hardware. Hence a BTF data compression that produces a compact representation is necessary. Such a method should provide:

- reasonably high compression ratios,
- fast random access data synthesis (convenient for GPU implementation and rendering algorithms),
- fidelity comparable with existing BTF compression algorithms.

In addition, the method should allow fast importance sampling for high-quality rendering applications using path-tracing algorithms, and spatial enlargement of measured BTF samples. The processing/workflow pipeline for BTF data is shown in Figure 2.

Contribution of the paper. In this paper, we present a novel BTF compression technique based on efficient multi-level vector quantization, allowing fast importance sampling for a given viewing direction as well as efficient multi-sample compression into a single shared database. This can be used efficiently in rendering algorithms such as path tracing. To our knowledge there is no other BTF compression method that has these features.

The rest of the paper is organized as follows: The following section describes the basic terminology used in the paper. Section 3 outlines prior work in the fields of BTF compression and importance sampling. Section 4 explains individual parts of the proposed model. Section 4.1 proposes a novel BTF data parametrization and interpolation. In Section 4.2, a vector quantization algorithm of interpolated data is explained, and this is followed by a description of a novel multi-level vector quantization method introduced in Section 4.3. Section 4.4 discusses a similarity measure applied throughout the model for BTF and BRDF data. Section 4.5 describes the use of scalar quantization to achieve further compression. The GPU implementation is briefly described in Section 4.6. Properties of the model and its application to fast importance sampling are discussed in Section 5. The results of our method are described in Section 6. A comparison of the method with other existing methods is shown in Section 7. Section 8 concludes the paper.

2. Basic Terminology and Notation

In this section, we describe basic terminology and notation used in the paper. The incoming light direction is denoted by $\omega_i = [\theta_i, \varphi_i]$ and viewing direction by $\omega_v = [\theta_v, \varphi_v]$. BRDF

is a four-dimensional (4D) function $BRDF(\omega_i, \omega_v)$ and has two main important properties [DF97]. The first one is the *Helmholtz reciprocity rule* stating that if the illumination and viewing direction are reversed, the value of the BRDF should not change. The second property is the *energy conservation law*, which states that the ratio of total outgoing radiance from the material and incoming total radiance from the light sources must be less than or equal to one for all possible illumination directions.

Monospectral BTF is a 6D function, $BTF(\mathbf{x}, \omega_i, \omega_v)$, which unlike BRDF, accounts for the dependence of viewing and illumination measurements on planar material position $\mathbf{x} = [x, y]$. BTF can be decomposed into a set of illumination and viewing direction dependent texels specifying pixel-wise BRDFs. We will describe such a texel as an *apparent BRDF* and denote it as $F_{\mathbf{x}}(\omega_i, \omega_v)$ in this paper. Contrary to BRDF, due to masking, shadowing, etc. effects the apparent BRDF does not fulfill the Helmholtz reciprocity rule [MMS*04], i.e. the role of illumination and viewing direction cannot be interchanged without any effect on a reflectance value. In general, the energy balance is also not preserved. This happens due to such effects as occlusion and masking or subsurface scattering inside a rough material structure.

3. Previous Work

In this section, we review relevant BTF compression methods and importance sampling algorithms for reflectance data.

3.1. BTF compression methods

Since the main purpose of this paper is to introduce a novel BTF compression technique we discuss here the principles and basic properties of methods for BTF compression. These methods can be roughly divided into the three following groups.

The first group is based on linear basis decomposition. This approach was presented by Koudelka *et al.* in [KMBK03]. Individual BTF images are ordered into the columns of a matrix. The corresponding symmetric matrix is created and subsequently decomposed using SVD. The compression method of [VT04] decomposes BTF space, ordered into tensor, by means of multi-modal SVD. Even though both methods enable realistic BTF rendering, they are not suitable for fast BTF applications since they require linear combinations of a large number of components. In [SSK03] the principal components for BTF images with the same view position are computed separately. [MMK03] exploited vector quantization of BTF data-space while each resulting cluster was represented by a local PCA model. This method was also applied for compression of psychophysically reduced BTF data in [FCGH08]. Another BTF vector quantization approach based on azimuthal rotation of resampled data $F_{\mathbf{x}}$ was mentioned in [KM06]. In [LM01] introduced a BTF recognition method that captured surface appearance under different illumination and viewing conditions by using three-dimensional (3D) textons constructed by means of K-means clustering of responses to selective linear filters applied at individual planar positions in BTF. This idea was exploited by [TZL*02] for BTF compression and fast rendering. The same authors [LZT*04] extended the method with a scheme for reduction of response vectors based on SVD. [MCT*05] presented an approach for BTF LOD rendering based on a Laplacian pyramid of resampled BTF data compressed by PCA. Recently, [RK09] applied K-SVD algorithm to decompose a massive BTF data tensor into a small dictionary and two sparse tensors.

The next group of compression methods represents BTF by means of analytical reflectance models. The pioneering work was done by [MLH02], who represented the reflectance of each pixel in BTF using the Lafortune reflectance lobes [LFTG97] parametrized by both view and illumination direction. A similar method using an additional look-up table to scale reflectance lobes and handle shadowing and masking was published in [DLHS01]. The spatial inconsistency of individual pixels in BTF for different view directions led to separate modeling of BTF images corresponding to only one view direction. [MGW01] represented each pixel in BTF by means of per-pixel polynomials. [MMK03] fit BTF by several pixel-wise Lafortune lobes for fixed viewing direction. The lobes are used only for fitting luminance values, which are used to modulate an albedo-map of individual colour channels. In [FH05] only one lobe is used per colour channel. The obtained results are then corrected by means of polynomials representing histogram matching functions between original and restored images. A similar method proposed in [ND06] uses a combination of histogram matching and steerable pyramids for sparsely sampled BTF compression. The BTF compression technique proposed in [MCC*04] models average BTF reflectance by the Phong model. Differences between the model and the original BTF are stored in residual BTF textures that are approximated by appearance perturbation parameters. BTF volumetric compression by a stack of semi-transparent layers through the surface texture was presented in [MK06].

The final group of compression methods achieved even better compression ratios and were based on probabilistic BTF modeling [HF07]. All of these models approximate a regular rough structure dependent on illumination position by means of a combination of a displacement filter and Markov random field-based texture synthesis of BTF subset images. Although these methods allow synthesis of arbitrary resolution BTFs, and reach impressive compression ratios, they sometimes compromise the visual quality of highly non-Lambertian materials. These methods also do not allow fast data synthesis for random access of individual apparent BRDFs. A recent comparison of some of the methods reviewed here is shown in a BTF survey paper [FH09].

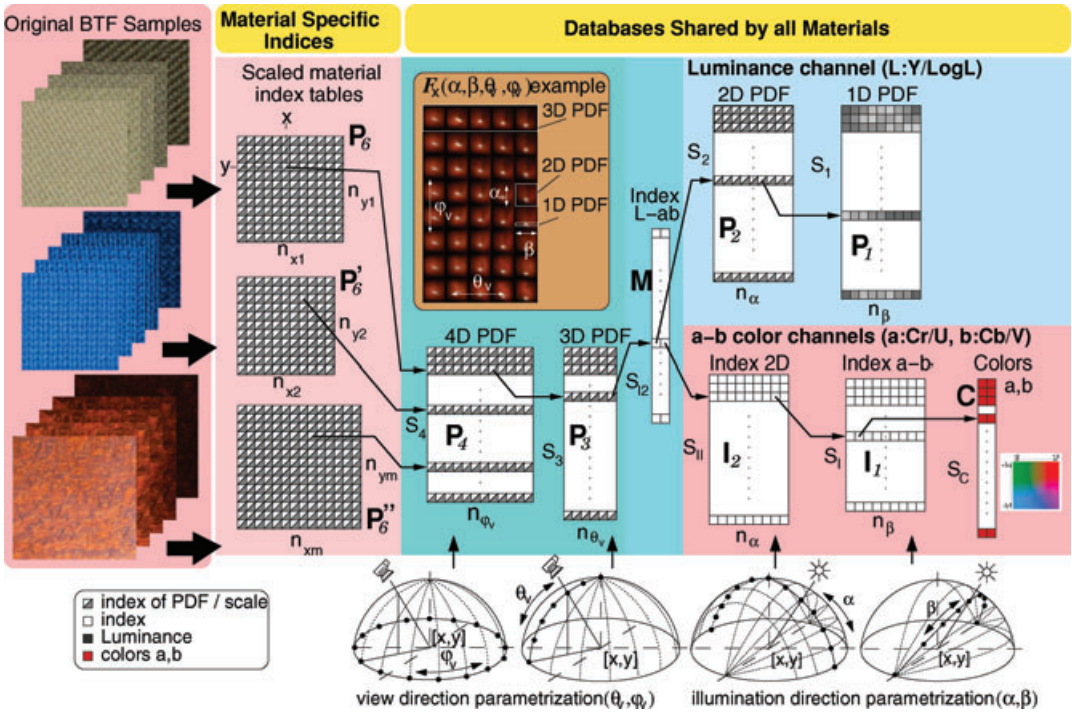


Figure 3: *The proposed BTF model scheme illustrating dependencies of individual VQ code-books.*

Although all the above mentioned methods allow very fast BTF rendering, they mostly achieve relatively moderate compression ratios, less than 1:200. Only some of them allow fast GPU implementation, only two ([LM01, KM06]) use a variant of vector quantization, and only a few of them allow importance sampling of BTF without reconstruction. Our method is designed to provide all the features required for CPU-based and GPU-based rendering algorithms.

3.2. Importance sampling

For random walk algorithms [DBB03] such as path tracing and bidirectional path tracing an efficient importance sampling algorithm according to material reflectance is necessary to reduce the variance of integral estimates.

Below we review the most relevant work on importance sampling for BRDF data. Lawrence *et al.* [LRR04] described the method of fast importance sampling of BRDF based on non-negative matrix factorization. The terms of this factorization are then used for the computation of cumulative distribution functions (CDF). This method provides much better results for the same number of samples than sampling based on CDF computation using analytical BRDF models, and a more compact representation than sampling based on tabulating the full BRDF. Another technique for reducing the size of tabulated CDF, by one to three orders of magnitude, based on a curve approximation algorithm is presented in [LRR05]. Another method based on non-linear PCA is presented in the thesis of Matusik [Mat03, p.107].

To the best of our knowledge, no other paper achieves fast importance sampling of highly compressed BTF data. Obviously, the importance sampling of BTF can be implemented for any BTF compression method by the inverse transform method [Fis96, DBB03] reconstructing apparent BRDF and computing CDF. We tested such an approach for [SSK03]; it is about 300 times slower than the direct support of BTF importance sampling in the proposed compression model.

4. A Novel BTF Model

The scheme of the proposed BTF model is shown in Figure 3. The compression scheme is based on subsequent decomposition of 4D, 3D, two-dimensional (2D) and one-dimensional (1D) slices of BTF data. These slices are obtained by resampling original BTF data to a novel parametrization of illumination and viewing directions. The model's compression is achieved by vector quantization of slices to individual dimensions to obtain a set of code-books. These code-books work as nested look-up tables of indices and scales of the individual slices while only the code-books at the lowest level contain the resampled original BTF data.

4.1. Model parametrization

The key motivation of the model was to propose a light direction parametrization over a hemisphere that enables not only efficient data compression but also fast rendering and importance sampling. In addition, as the proposed model decomposes the function to parts in the individual dimensions separately, we want to align the data at these individual dimensions.

We considered several different parametrizations proposed for BRDFs, e.g. half-angle parametrization by Rusinkiewicz [Rus98], by Stark *et al.* [SAS05] and by Edwards *et al.* [EBJ*06]. However, we decided to abandon them for three reasons. First, the published parametrizations do not preserve monotonicity between the generated direction and the bivariate uniform variable in the input domain. That is, when we generate a similar pair of random numbers we want to get a similar generated direction for all random pairs. This is discussed in more detail in Section 5.2. Although, parametrization proposed in [HDS03] preserves the monotonicity it would be complicated to use it the proposed multi-level quantization scheme. Second, many BTF samples have distinct properties from BRDF samples. We found experimentally by visualization that since BTF also captures the geometry of the surface, the alignment of the data features is not the same as for BRDF data measured on a smooth surface. Typically, there can be several specular highlights that are not aligned with the direction of an ideal reflected ray. Therefore the conditions are not satisfied under which other parametrizations such as halfway vector disk parametrization [EBJ*06, SAS05, Rus98] were proposed. The third reason is that we want to compress not only one but many apparent BRDFs. We want to align their perceptually similar features as we expect similarity among apparent BRDFs across a BTF sample. Hence the design of the parametrization proposed here specifically for BTF data compression is tightly coupled with a multi-level vector quantization method described in Section 4.3.

The proposed parametrization defines BTF slices that can be represented as conditional probability density functions (PDF). These PDFs are treated as input data into the vector quantization scheme proposed in the Section 4.2.

The proposed $[\alpha, \beta]$ parametrization is based on an 'onion slices' concept of a hemisphere of illumination directions, as illustrated in Figure 4. The hemisphere is divided into a set of meridian slices running between points **A** and **B** lying at its bottom part. Each slice is parametrized by angle $\beta \in \langle -\pi/2, \pi/2 \rangle$ with a zero value at the upper pole **E** of the hemisphere. A uniform placement of individual 1D slices over a hemisphere is controlled by angle $\alpha \in \langle -\pi/2, \pi/2 \rangle$ with zero value at the upper pole as well. A mapping $\mathcal{M}$ between standard hemispherical $[\theta, \varphi]$ parametrization and the proposed $[\alpha, \beta]$ parametrization can be stated as follows:

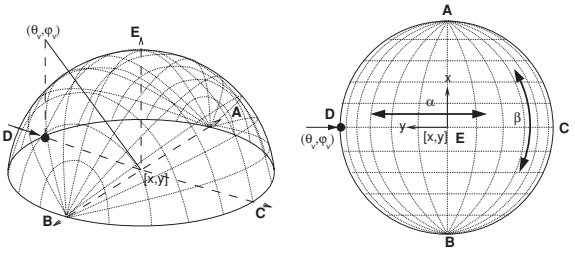


Figure 4: *Illumination direction parametrization over a hemisphere.*

$$\begin{aligned} &\mathcal{M}(\theta, \varphi) && \rightarrow \{\alpha, \beta\} \\ &\theta \in \langle 0, \pi/2 \rangle && \alpha \in \langle -\pi/2, \pi/2 \rangle \\ &\varphi \in \langle 0, 2\pi \rangle && \beta \in \langle -\pi/2, \pi/2 \rangle. \end{aligned} \tag{1}$$

A corresponding unit 3D directional vector can be specified by means of the $[\theta, \varphi]$ and $[\alpha, \beta]$ parametrization, respectively as

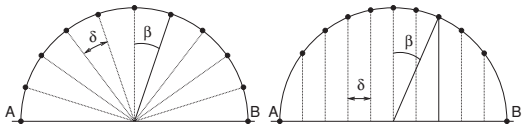
$$\begin{aligned} [x, y, z] &= [\cos\varphi \cdot \sin\theta,\ \sin\varphi \cdot \sin\theta,\ \cos\theta] \\ [x, y, z] &= [\sin\beta,\ \sin\alpha \cdot \cos\beta,\ \cos\alpha \cdot \cos\beta]. \end{aligned} \tag{2}$$

While the illumination direction ω_i is specified by $[\alpha, \beta]$, the viewing direction ω_v is given by standard $[\theta, \varphi]$ parametrization only resampled to regular sampling steps of angles θ and φ. On one hand this resampling causes dense sample distribution near the pole of the hemisphere but on the other hand it allows direct factorization of samples along angles θ_v and φ_v. Such a resampling consequently allows better compression of underlying data samples.

Now we describe how $F_{\mathbf{x}}$ is resampled from original spherical parametrization $\theta_i', \varphi_i', \theta_v', \varphi_v'$

$$\begin{aligned} &\{\alpha, \beta, \theta_v, \varphi_v\} \leftarrow \mathcal{M}\left(\theta_i', \varphi_i', \theta_v', \varphi_v'\right) \\ &\theta_v = \theta_v' \qquad \beta = \arcsin\left(\sin\theta_i' \cdot \cos\left(\varphi_i' - \varphi_v'\right)\right) \\ &\varphi_v = \varphi_v' \qquad \alpha = \arccos\left(\frac{\cos\theta_i'}{\cos\beta}\right). \end{aligned} \tag{3}$$

Note that the hemisphere is oriented in such a way that an outline between points **A** and **B** is always perpendicular to the azimuth of viewing direction φ_v. Such an arrangement guarantees that the ideal mirrored reflection is embedded in the plane given by points **D**, **C** and **E** (i.e. when $\beta = 0$). This means that the highest probability of a steep change in reflectance is in the middle of the slice, so this part should be sampled more densely than its tails. In addition, we need to achieve equitable distribution of samples on the hemisphere in $[\alpha, \beta]$ parametrization. Due to this reason the sample distribution along β slice is not chosen uniformly according to β angle shift (Figure 5(a)), but uniformly in $\cos\beta$ so that the projection of the samples is equidistant as shown in Figure 5(b). The poles **A** and **B** are accounted only once for



(a) uniform in β (b) uniform in cos(β)

Figure 5: *Sample points distribution along the 1D slice.*

2D PDF. These arrangements enable more uniform sampling of the illumination hemisphere and avoid dense sampling near the points **A** and **B** (see Figure 4).

In this paper, BTF data from [SSK03] were used. These data provide uniform distribution of measurement points at 81 illumination and 81 viewing angles over the hemisphere. Such a distribution is obtained by using variable quantization of azimuth angle φ for individual elevation angles θ. These BTF data were resampled into proposed illumination $[\alpha, \beta]$ and viewing $[\theta_v, \varphi_v]$ direction parametrization by means of a two-step interpolation scheme based on radial basis functions [CBC*01]. In the first step, the data for all illumination directions ω_i and the fixed viewing direction $\omega'_v = [\theta'_v, \varphi'_v]$ are interpolated, and these interpolated values for all combinations of α and β angles are then interpolated into a new $\omega_v = [\theta_v, \varphi_v]$ viewing direction discretization. The resulting data (see example in orange part of Figure 3) are then used as direct input into the proposed multi-level vector quantization based BTF model.

4.2. Vector quantization

The proposed BTF compression model is based on the principle of lossy block coding often referred to as vector quantization (VQ) [GG92]. VQ is based on an assumption that a set of data vectors can be represented by its representative subset – the code-book. This subset is obtained by representing similar vectors m by a suitable code-vector $\hat{m}$ according to a predefined maximal allowed distance. The similarity between them is defined by a distance measure $d(m, \hat{m}) > 0$.

Let us mention the important theorem for lossy compression methods related to our work [GG92, p.313]: *When the code-book is set optimally then no other coding system exists that can do better than VQ.* So a careful design of the code-vectors is the main issue. Even if we cannot claim the selection of thresholds to be optimal in the proposed algorithm, this theoretical result both motivates and justifies the use of vector quantization in lossy compression schemes, including our proposed BTF data compression scheme.

In this paper, a vector code-book is based on selective elimination of input data-vectors until a final set of input data-vectors remains as the code-book, a procedure also known as pruning [TG74]. This idea of code-vectors generation can be explained as:

1. Put the first input data-vector V_1 in the empty code-book
2. With each new input data-vector V_x, find the nearest code-vector V_{NN} in the code-book
3. *If* the minimum found distance between the vector V_x and V_{NN} is not within some threshold ε, add the input data-vector V_x to the code-book and return its index. *Continue* to 2. *Else* return index of the nearest code-vector V_{NN}. *Continue* to 2.

The selection of a distance measure appropriate for input data and the setting of the corresponding threshold ε have a crucial influence on the performance of the vector quantizer.

4.3. Multi-level vector quantization

Now we can connect all the building blocks described above and explain our BTF compression model. As input data a $F_{\mathbf{x}}$ is converted from original θ'_i, φ'_i and θ'_v, φ'_v data parametrization into a novel parametrization $[\alpha, \beta]$ and $[\theta_v, \varphi_v]$ as described in Section 4.1. An example of $F_{\mathbf{x}}$ for lacquered wood material is depicted in the orange part of Figure 3.

The general scheme of the proposed BTF model is shown in Figure 3. The resulting 4D function $F_{\mathbf{x}}(\alpha, \beta, \theta_v, \varphi_v)$ is decomposed along a viewing azimuth angle φ_v into a set of 3D functions. Similarly, each 3D function is decomposed along a viewing elevation angle θ_v into a set of 2D functions. Each 2D function describes the behaviour of material reflectance along all slices in $[\alpha, \beta]$ parametrization, where data of a single slice can be considered as a 1D function. To enable perceptually correct matching of individual data patterns and sharing of some common material features, the input BTF data were converted from standard RGB space into more perceptually uniform colour space. YCrCb colour space was used for LDR BTF samples and LogLUV [Lar98] for HDR BTF samples.

The advantage of both colour models is mutual independence of luminance and colour channels that can be treated and compressed separately. In the rest of the paper, regardless of the colour-space that is used the luminance channel is denoted by L and chromaticity channels by a and b.

The original 1D, 2D, 3D and 4D luminance functions are normalized to obtain corresponding conditional probability functions (PDF), which are used as training vectors for our VQ scheme. The proposed BTF model is based on BTF data decomposition into several code-books of indices and scale coefficients, while only the code-books on the lowest level contain the resampled original BTF data as 1D vectors.

The vector quantization of luminance BTF data is carried out separately for individual dimensions as shown in Figure 3. As a result of 1D PDFs quantization we obtain code-book $\mathbf{P}_1$(size $S_1 \times n_\beta$) of normalized 1D data slices along illumination angle β. This code-book is indexed by

the $\mathbf{P}_2$(size $S_2 \times n_\alpha$) code-book of 2D PDFs representing luminance values where each item contains indices and scales c_{P_2} of individual 1D slices along illumination angle α in $\mathbf{P}_1$. Items in $\mathbf{P}_2$ are indexed from the auxiliary code-book $\mathbf{M}$(size $S_{I2} \times 2$). $\mathbf{M}$ in fact only merges indices pointing into luminance and colour code-books ($\mathbf{P}_2$ and $\mathbf{I}_2$) and is indexed from the code-book of 3D PDFs $\mathbf{P}_3$(size $S_3 \times n_{\theta_v}$), where for each item indices and scales c_{P_3} corresponding to viewing angle θ_v are stored. The $F_\mathbf{x}$ encoding is finished by the last shared code-book $\mathbf{P}_4$(size $S_4 \times n_{\varphi_v}$), which provides items corresponding to viewing angle φ_v with indices and scales c_{P_4} to $\mathbf{P}_3$.

Chromaticity channels a and b (Cr/Cb or U/V) of BTF data are quantized in a slightly different way. The $\mathbf{C}$(size $S_C \times 2$) code-book stores basic a and b colour values. Possible colour variations along 1D slices are described in items of the $\mathbf{I}_1$(size $S_I \times n_\beta$) code-book and the corresponding colour can be looked-up by indexing into $\mathbf{C}$. Colour variations for all illumination directions are obtained by means of items of the $\mathbf{I}_2$(size $S_{I1} \times n_\alpha$) code-book. Each such item of length n_α determines which colour variations from $\mathbf{I}_1$ are used for individual positions of angle α.

The code-books $\mathbf{P}_2$ and $\mathbf{I}_2$ are stored separately to allow the use of different colour variations for the same luminance distribution over a hemisphere of different illumination directions. This arrangement also makes it possible to save considerably fewer $\mathbf{P}_2$ slices when, e.g., BTFs of similar material structure but different colour are encoded. The luminance and colour information is merged by means of the auxiliary code-book $\mathbf{M}$ indexed from $\mathbf{P}_3$. $\mathbf{M}$ contains only index to $\mathbf{P}_2$ and index to $\mathbf{I}_2$. The remaining $\mathbf{P}_4$ and $\mathbf{P}_3$ code-books have the same function as in the luminance channel.

During BTF compression, individual $F_\mathbf{x}$ are compared with reconstructed $\hat{F}_\mathbf{x}$ in $\mathbf{P}_4$ by means of nested indexing through all code-books. If a similar code-vector is not found, $F_\mathbf{x}$ is decomposed into a set of less dimensional slices and the same process continues on all levels of the model either until the similar slice is found, or the $\mathbf{P_1}$ or $\mathbf{C}$ code-books are reached. Then the new unique data are inserted into the code-book $\mathbf{P_1}$ in the form of a luminance vector of length n_β along a slice parametrized by angle β or a chromaticity in the code-book $\mathbf{C}$. The insertion to $\mathbf{P_1}$ and $\mathbf{C}$ corresponds to standard vector quantization. During insertion the data-vector is compared so that the luminance is normalized in both the inserted data-vector and the data-vector in the code-book. When the match is found, this then provides a corresponding scale for upper-level code-book.

All the code-books described so far enable efficient coding of colour $F_\mathbf{x}$ and can be shared by more BTF samples (i.e., materials). However, individual apparent BRDFs $F_\mathbf{x}$ do not provide any information about sample structure, so for coding of an entire BTF a material-specific planar index is needed. Such an index is obtained by VQ of individual $F_\mathbf{x}$ and stored in a form of $\mathbf{P}_6(n_{x_m} \times n_{y_m})$ code-book where $n_{x_m} \times n_{y_m}$ is

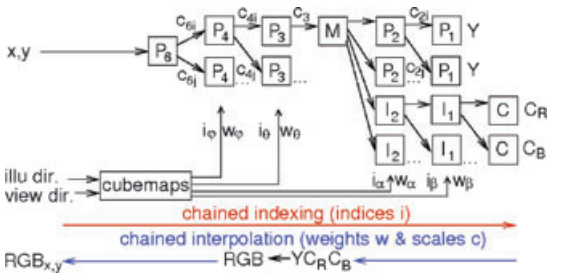


Figure 6: *Pixel reconstruction in a fragment shader.*

the spatial resolution of the mth BTF sample. $\mathbf{P}_6$ contains an index to $\mathbf{P}_4$ together with its scale value c_{P_6}.

The scale values are used for scaling of the stored PDFs to obtain correct reconstruction of 4D PDF function, i.e. $F_\mathbf{x}$, in the form of a compound function as follows

$$\begin{aligned} c_{\text{scale}} &= c_{P_6} \cdot c_{P_4} \cdot c_{P_3} \cdot c_{P_2} \\ k &= \mathbf{P}_3(\mathbf{P}_4(\mathbf{P}_6(x, y), \varphi_v), \theta_v) \\ F_{\mathbf{x}L} &= c_{\text{scale}} \cdot \mathbf{P}_1(\mathbf{P}_2(\mathbf{M}(k, 1), \alpha), \beta) \\ F_{\mathbf{x}\{a,b\}} &= \mathbf{C}(\mathbf{I}_1(\mathbf{I}_2(\mathbf{M}(k, 2), \alpha), \beta), \{1, 2\}) \\ F_{\mathbf{x}\{L,a,b\}} &\rightarrow F_{\mathbf{x}\{R,G,B\}}. \end{aligned} \tag{4}$$

A scheme of pixel value reconstruction and interpolation is shown in Figure 6.

4.4. Similarity measure

For VQ in the proposed BTF compression model we need a similarity measure between the input data-vector and the stored code-vector; this is of crucial importance for the compression algorithm. The data-vector corresponds to either a 1D, 2D, 3D or 4D slice of $F_\mathbf{x}$ of BTF at a given planar position. As the proposed BTF compression model can use any similarity measure, we studied and tested several possibilities. The first group of measures comes from comparing probability density functions (PDF), as BTF data decomposed to $F_\mathbf{x}$ correspond to the PDF. We have been experimenting with a number of similarity measures (i.e. distance functions), including f-divergences [RFS03] (e.g. Hellinger distance [Hel09] and total variation) and information based distances (mutual information and entropy). The second group of measures includes traditional distances for comparing functions. It is for example Euclidean distance corresponding to MSE, which is related to the power of a function when viewed as a signal. As a third group of measures we tested similarity measures developed in the perception for visual image quality assessment. Below we describe our final choices for this paper, but the selection of the optimal similarity measure in BTF compression remains an open problem.

4.4.1. *BRDF data compression*

As BRDF data lacks the spatial neighborhood information, we decided to use the mean square error (MSE) as a distance function between the original and the compressed data. The computation of MSE, which corresponds to computing Euclidean distance, has one big advantage. We can specify for each code-book $\mathbf{P}_1, \mathbf{P}_2, \mathbf{M}, \mathbf{P}_3$ and $\mathbf{P}_4$ the maximum MSE that is acceptable for compression. This allows us to effectively control the maximum MSE achieved for each BRDF sample. While the maximum MSE for $\mathbf{P}_4$ is user specified, the MSE thresholds for other code-books are smaller by multiplicative constants such as 0.4 and squares of multiplicative constants among code-vectors. This is possible thanks to the function decomposition scheme described in Section 4.3.

The MSE for $\mathbf{P}_4$, $\mathbf{P}_3$ and $\mathbf{M}$ is computed directly in sRGB colour space according to the definition of MSE, but the MSE for $\mathbf{P}_2$ and for $\mathbf{P}_1$ is computed for luminance only. The thresholds for code-books $\mathbf{I}_1, \mathbf{I}_2, \mathbf{C}$ of colour components of YCrCb/LogLUV space are set to small constants; $\mathbf{I}_2$ threshold $= 0.5$, $\mathbf{I}_1$ threshold $= 0.2$, $\mathbf{C}$ threshold $= 0.1$.

4.4.2. *BTF data compression*

After experiments with several similarity measures we finally decided to analyze BTF samples using a structural similarity index measure (SSIM) [WBSS04], which compares in power to other visual assessment methods such as a visible difference predictor [Dal93]. Another advantage of SSIM over other standard image quality measures as MSE, PSNR, etc. is that SSIM also takes into account both the surroundings of the compared pixels and local visual masking effects. SSIM measures the local structure similarity in their local neighborhood of an $R \times R$ window of pixels in an image (usually 11×11, [WBSS04]). The basic idea of SSIM is to separate the task of similarity measurement into comparisons of luminance, contrast, and structure. These independent components are then combined into one similarity function

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}, \quad (5)$$

whose formulation should be qualitatively consistent with human perception of difference. Where $\mu_x, \mu_y, \sigma_x, \sigma_y$ and σ_{xy} are mean values, standard deviations, and mutual variance of values in the local neighbourhood of compared images X and Y. C_1, C_2 are specific non-zero constants. The valid range of SSIM for a single pixel is $[-1, 1]$, with higher values indicating higher similarity. When the local neighbourhood is evaluated for each pixel we obtain the SSIM difference of two images (or two sets of images) as a mean value of SSIM values across all pixels. This mean value is understood as MSSIM in the rest of the paper.

The MSSIM is computed for $\mathbf{P}_2$ and for $\mathbf{P}_1$ over luminance only, for $\mathbf{P}_4, \mathbf{P}_3$ and $\mathbf{M}$ in YCrCb/LogLUV colour space, for $\mathbf{I}_1$ and $\mathbf{I}_2$ and $\mathbf{C}$ in two chromaticity channels of YCrCb/LogLUV colour space. We tested the method of weighting MSSIM values from three channels using the following weights $\{\text{Y,LogL}\} = 0.8$ and $\{\text{Cr,U,Cb,V}\} = 0.1$ as proposed in [WLB04]. However, as the weighting method appeared not to be discriminative enough for chromaticity of colour images, we propose a different method. For example for $\mathbf{P}_4$ we compute MSSIM for all three channels (e.g Y, Cr and Cb) for all combinations of viewing and illumination direction for the selected discretization over the hemisphere, which yields $3 \times \mathbf{n}_{\varphi_\mathbf{v}} \times \mathbf{n}_{\vartheta_\mathbf{v}} \times \mathbf{n}_\alpha \times \mathbf{n}_\beta$ values. As a similarity measure, we then compute the 98th-percentile from MSSIM values for all three channels. The 50th-percentile is the median and the 100th-percentile is the maximum error for a set of values, which corresponds to the worst similarity. In our approach we allow only 2% of outliers.

The proposed approach is computationally efficient, as it allows us to prune the vector comparisons during the search as soon as we achieve the percentile value already found as the current best in the code-book found. The Nth-percentile of MSSIM values is consistently computed over all code-books either from luminance ($\mathbf{P_2}, \mathbf{P_1}$), two chromaticity channels ($\mathbf{C}, \mathbf{I_2}, \mathbf{I_1}$), or all three channels ($\mathbf{P_4}, \mathbf{P_3}, \mathbf{M}$).

The decomposition of PDF to levels during insertion of new code-vectors is natural for the percentile method. For example, when a code-vector V_x for $\mathbf{P_4}$ is constructed from $\mathbf{P_3}$ vectors using the proposed percentile method given a threshold ε_{P3}, it is assured that the constructed code-vector V_x has a similarity measure smaller than ε_{P3}. Therefore, there is no need for a multiplicative factor for thresholds, as is the case for the MSE method described in the previous section.

4.5. Scalar quantization and compact indices for code-books

Scalar quantization. During compression, we store the indices and scale values in code-books simply by 32-bits for an integer index and for floating point in 32-bits in IEEE-754 format. However, the scale values are limited to a small range of values in the majority of code-books. Therefore, after the BTF sample is compressed, we apply a simple scalar quantization [GG92] for floating point values. First, we compute minimum and maximum values stored in each code-book separately. For simplicity and ease of decompression we use scalar quantization to 8 bits for LDR BTF samples for all levels, as the original data also have only 8 bits precision. For HDR BTF samples it is necessary to increase the precision for $\mathbf{P}_2$ to 16 bits. The maximum relative error of a value due to the scalar quantization is far below 1% in all cases, typically the relative error yields values in range from 10^{-4} to 10^{-3}.

Compact indices. Similarly, the size of code-books is reduced. Therefore the index in a code-book $\mathbf{P}_i$ pointing to another code-book $\mathbf{P}_{i-1}$ of size $\mathbf{S}_{i-1}$ can be represented only by $N = \lceil \log_2(S_{i-1}) \rceil$ bits.

Table 1: *BTF decompression speed in frames per second on CPU and two GPUs for point light and three objects. The texture resolution is* 1024×1024*, window size is* 800×600 *pixels.*

3D model	CPU Intel C2D 2 GHz	ATI Mobility Radeon™ ×1600	NVIDIA GeForce 8800 GT
Bunny (2k faces)	2.0	18	92
Sphere (7k f.)	1.2	10	170
Dragon (115k f.)	1.1	8	10

The final representation of BTF data comprising scalar quantized data and compact indices is saved to a file. Both scalar quantization and shorter indices substantially improve compression and the impact is shown in the results section, as documented in Table 2. We verified experimentally that the used scalar quantization that we used does not reduce the visual quality for either the LDR or the HDR BTF data.

4.6. CPU/GPU implementation

The BTF reconstruction is similar on CPU as well as on GPU. The GPU implementation works on both GL shading language and CUDA platforms. As GPU implementation is more specific we discuss it more in detail. The individual code-books were stored in rows forming stripes of widths $n_\alpha \ldots n_{\theta_v}$ in RGBA channels of four 16-bit integer textures (indices of $\mathbf{P}_6$, $\mathbf{P}_4$, $\mathbf{P}_3$, $\mathbf{P}_2$, $\mathbf{M}$, $\mathbf{I}_2$, $\mathbf{I}_1$) and two 16-bit floating point textures (scales of $\mathbf{P}_6$, $\mathbf{P}_4$, $\mathbf{P}_3$, $\mathbf{P}_2$ and data of $\mathbf{P}_1$ and $\mathbf{C}$). Texture resolution depends on sizes of individual code-books. The textures of 2048×2048 pixels were sufficient for all tested samples. We use two indices for texture indexing – x specifies which vertical stripe to use and y specifies the row in the stripe. The reconstruction of each pixel was implemented by chained indexing and interpolation of values from code-books and YCrCb/LogLUV to RGB conversion in a fragment program as shown in Figure 6. To avoid visible seams on the rendered objects, the correct value at each database level for given illumination/view angle $(\alpha, \beta, \theta_v, \varphi_v)$ was computed as linear interpolation between the two closest values m, n corresponding to discretization of individual angles (Figure 6). Due to this, the reconstruction of a single pixel for arbitrary given illumination/view directions requires 47 reads of integer texture ($1 \times \mathbf{P}_6$, $2 \times \mathbf{P}_4$, $4 \times \mathbf{P}_3$, $8 \times \mathbf{M}$, $8 \times \mathbf{P}_2$, $8 \times \mathbf{I}_2$, $16 \times \mathbf{I}_1$) and 63 reads of float texture ($1 \times \mathbf{P}_6$, $2 \times \mathbf{P}_4$, $4 \times \mathbf{P}_3$, $8 \times \mathbf{P}_2$, $32 \times \mathbf{C}$, $16 \times \mathbf{P}_1$). No further interpolation of illumination/view directions is required. The performance of our CPU and GPU implementation is shown in Table 1.

5. Discussion

This section discusses features of the proposed model and its application for importance sampling of BTF data.

5.1. Vector quantization scheme

There are several advantages of the proposed model. The reconstruction of BTF values is computed by fast chained indexing in individual code-books. The individual code-books of luminance and colour slices and their indices can be shared by an arbitrary number of BTF samples and can therefore enable even higher compression. Theoretically, the more BTF materials are compressed, the higher the compression ratio that could be achieved. Data compression is carried out on all levels of the proposed model and can be effectively controlled by dedicated thresholds. We set the thresholds in such a way that the low-level code-books contain most of the code-vectors. Note that the higher the compression ratio, the shorter is the compression time, since individual code-books have fewer items to be evaluated. In contrast methods based on PCA [MMK03, VT04] or spherical harmonics [WL03], which have predefined compression ratios, our approach can adapt to variance in input data-vectors and can keep the relative error constant (given by the predefined thresholds ε of the required MSE or SSIM error). In addition, whenever a new BTF sample arrives it can be easily processed by our model, using some scaled variant of already stored code-vectors if possible, and adding some of its own typical luminance and/or colour characteristics. This is much more difficult, or even not feasible, with the other methods mentioned above. Furthermore, the compressed BTF can be spatially enlarged using any pixel or patch-based texture synthesis algorithm applied to the $\mathbf{P}_6$ code-book.

5.1.1. *Generation of optimized code-books*

The order of processing apparent BTFs across a BTF sample has a large impact on the final results. To guarantee that code-books describe a perceptual variety of pixels across a whole BTF sample, and to ensure a sufficient compression ratio, the individual code-books are generated in a three-step progressive sampling algorithm. First, a small set (e.g. 1%) of apparent BTFs $F_{\mathbf{x}}$ across the whole BTF sample is randomly progressively sampled from BTF data with a predefined threshold and is used in the VQ scheme. The samples are taken from a Halton pattern. Second, the threshold is increased (e.g. by 2.5 times) and the same process is repeated for a larger set of $F_{\mathbf{x}}$ (e.g. 4%), again for the whole BTF sample. In the third step we do not modify the code-books and compress the rest of the pixels (e.g. 95%) in any order.

Let us describe the motivation for the necessity of the three-step sampling progressive algorithm. The sampling strategy in the first step samples the material and creates the representative code-books to capture the visual appearance of the whole BTF sample with sufficient quality. In the second step we add to the code-books the remaining relevant visual features that have been undersampled in the first step. We process only a small portion of the whole BTF material in the first two steps, and we fix the code-books for the third step. This way we guarantee that the compression ratio will

remain sufficiently high, irrespective of the thresholds that are selected. The selection of the thresholds then influences the visual quality that is achieved. This appears to be similar for setting the same thresholds and using different BTF samples.

The number of generated items in individual data sets can become so high that finding the closest code-vector can be very slow. For BRDF data comparison we take advantage of the fact that we are using MSE, which is a true metric, and so we address this problem by implementing a dynamic version of the LAESA method [MOV94] (see book [Sam06] for other nearest neighbor search algorithms in high dimensional spaces). For SSIM and 98th-percentile the efficient search pruning is implemented as described in Section 4.4.

5.1.2. *Thresholds setting*

A common problem of all VQ algorithms is finding the optimal quantization thresholds that provide either a required compression ratio or satisfy a defined quality measure. In our case, such a measure is the computational model of perceived difference between rendered images using the VQ BTF compression scheme and images rendered using the original BTF data. When MSE is used as similarity measure, the maximum MSE difference allowed or required for each generation of a code-book is specified by a user.

When SSIM is a similarity measure the situation is also simple, because the measure directly estimates the perceptual difference between the original and the modeled data. The big advantage is that no material-specific setting is required and we can again set required generic SSIM thresholds for individual code-books. To summarize, the setting of thresholds effectively controls the trade-off between the compression ratio of the proposed VQ compression scheme and the visual fidelity of the resulting rendered images. There is an obvious limitation of our current approach – the SSIM is only a mathematical model of visual fidelity given two images. So the visual fidelity achieved is limited by accuracy of SSIM. The ranking of visual fidelity can be only approximate.

5.1.3. *Mipmapping*

Since rendering at different scales is important for direct visualization of BTF data on the visible object, it is necessary to address an anti-aliasing. In our model, mipmapping [Wil83] can be used in the same way as for ordinary textures. The reflectance data are averaged, and the data are compressed from fine scale to coarse scale, each level separately. This requires extension of the spatial index $\mathbf{P}_6$. Obviously, the compression ratio is decreased up to one third as for standard texture mipmapping. The speed of decompression is also decreased as more data need to be accessed.

5.2. Importance sampling

Importance sampling is not supported by current BTF models, but our algorithm design allows it efficiently. It is implemented via a standard inverse transform method for discrete PDFs [Fis96] directly from the $\mathbf{P_2}$ code-book, without the necessity to compute the 2D PDF, as in for several other BTF compression methods. The proposed parametrization over 2D slices guarantees that for strictly positive values $F_{\mathbf{x}}$, and given the viewing direction ω_v and a couple of random numbers $\xi_{1,2} \in [0-1]^2$, we can generate the illumination direction ω_i. The implementation computes cumulative distribution functions (CDF) along 1D slices in $\mathbf{P}_1$ for ξ_1, and similarly CDF for across particular 2D slices in $\mathbf{P}_2$ for ξ_2. The probability density values have to be properly interpolated because of the discretization in θ_v and φ_v.

In contrast to [Mat03, LRR04, EBJ*06] our hemispherical parametrization of apparent BRDF allows us to preserve monotonicity between the generated direction and the bivariate uniform variable in the input domain, and avoids discontinuities at the same time (only for the hemisphere). If for random numbers ξ_1, ξ_2 the function generating direction is $\omega_i = DirIS(\xi_1, \xi_2)$, then it holds $\lim_{\delta_1 \to 0, \delta_2 \to 0} |DirIS(\xi_1 + \delta_1, \xi_2 + \delta_2) - DirIS(\xi_1, \xi_2)| = 0$ for $\forall \xi_1, \xi_2 \in [0, 1]^2$. Informally, for a small change of input random variables we get also a small change of the generated direction as a result of importance sampling. This is important for adaptive importance sampling schemes, in particular for those based on quasi-Monte Carlo numbers, as the importance sampling algorithm does not increase the intrinsic dimensionality of the problem solved. Therefore the variance of the mean estimate is not increased by the importance sampling algorithm.

The sampling from the proposed parametrization allows fast sampling for a given viewing direction. This is the most frequent importance sampling required in path-tracing and photon mapping, when tracing the rays from the camera towards the scene. For the other case of importance sampling, given the illumination direction, we have two options. First, we can use a standard inverse transform method for the values of 2D PDF reconstructed over the hemisphere in the parametrization as $[\alpha, \beta]$. Because importance sampling given illumination direction is much less frequently used in rendering algorithms (for example, only for shooting photons in photon mapping), the proposed data organization is more efficient for most rendering applications. Second, to achieve the fast importance sampling for both cases it is possible to compress the BTF data twice – first for the fixed viewing direction and second for fixed illumination direction. This approach decreases the compression ratio by half.

In addition, the proposed parametrization allows fast computation of albedo [NJH*77] for a viewing direction using

$$a(\mathbf{x}, \omega_v) = \int_{\Omega} F_{\mathbf{x}}(\omega_i, \omega_v) \cos\theta_i \, \mathrm{d}\omega_i \qquad (6)$$

Table 2: *Comparison of our method with three other methods in terms of compression ratio and $MSSIM_W$ [WBSS04] values in YCrCb space for all tested materials. The range of MSSIM is $\langle 0.0, 1.0 \rangle$, where value 1.0 corresponds to equal images. $C.R.^1$ is the compression ratio for representing code-book indices by 32-bits and floating point values by 32 bits. $C.R.^2$ is the compression ratio for representing indices by minimum numbers of bits and floating point values by 32 bits. $C.R.^3$ is the compression ratio for representing indices by 32 bits and floating point values by 8 bits except $\mathbf{P}_2$ where 16 bits are used for HDR samples and 8 bits for LDR samples. $C.R.^4$ and $size^4$ is the compression ratio and the compressed size of BTF sample for representing indices by minimum numbers of bits and floating point values by 8 or 16 bits in the same way as for $C.R.^3$. Compression ratio $C.R.^5$ uses the same representation as $C.R^4$, but several BTF samples are compressed together for sharing luminance characteristics. The combined compression yields improvement in compression ratio by 14% for 13 LDR materials and 41% for 4 HDR materials. The visual quality is not changed for different representations of data in the proposed algorithm.*

No	BTF sample	Our C.R.[1]	Our C.R.[2]	Our C.R.[3]	**Our C.R.[4]**	Our C.R.[5]	Our size[4] (Bytes)	Time (hours)	Our SSIM	PCA SSIM	LM SSIM	LPCA[‡] SSIM
1	alu*	1:253	1:399	1:453	**1:1002**	–	80 460	0.39	0.850	0.929	0.882	0.941
2	corduroy	1:128	1:246	1:173	**1:418**	1:484	3 084 692	19.2	0.748	0.921	0.898	0.916
3	fabric1	1:117	1:181	1:188	**1:362**	1:419	3 558 788	29.3	0.737	0.915	0.889	0.915
4	fabric2	1:217	1:342	1:353	**1:710**	1:822	1 815 996	18.1	0.779	0.994	0.987	0.993
5	foil1	1:574	1:918	1:950	**1:2040**	1:2364	632 352	19.5	0.859	0.924	0.893	0.923
6	foil2	1:334	1:527	1:553	**1:1138**	1:1319	1 133 896	17.2	0.814	0.995	0.990	0.994
7	impalla	1:162	1:249	1:269	**1:522**	1:604	2 466 808	21.8	0.730	0.971	0.959	0.970
8	leather	1:366	1:581	1:601	**1:1244**	1:1441	1 036 352	17.2	0.802	0.994	0.992	0.994
9	proposte	1:236	1:418	1:349	**1:806**	1:934	1 599 412	18.0	0.710	0.990	0.979	0.988
10	pulli	1:87	1:131	1:141	**1:264**	1:306	4 873 008	27.1	0.699	0.964	0.950	0.955
11	wallpaper	1:222	1:369	1:340	**1:728**	1:843	1 771 420	28.8	0.776	0.955	0.940	0.963
12	wood1	1:101	1:247	1:118	**1:352**	1:408	3 664 060	22.3	0.866	0.827	0.789	0.811
13	wood2	1:75	1:200	1:87	**1:278**	1:322	4 625 772	17.2	0.886	0.954	0.892	0.957
14	wool	1:77	1:153	1:95	**1:233**	1:270	5 514 260	50.2	0.684	0.969	0.953	0.964
15	ceiling◇	1:235	1:451	1:345	**1:780**	1:1102	2 653 188	20.1	0.711	–	–	–
16	floortile◇	1:136	1:217	1:197	**1:360**	1:509	5 383 352	28.7	0.772	–	–	–
17	pinktile◇	1:711	1:1028	1:1258	**1:2267**	1:3205	853 496	15.6	0.961	–	–	–
18	walkway◇	1:102	1:149	1:147	**1:257**	1:363	7 514 860	37.4	0.884	–	–	–
		1:230	1:378	1:368	**1:764**	1:924	Average C.R.		–	1:14	1:16	1:275

*Sample size 64 × 64 only, ◇ HDR sample.
‡Computed 128 × 128 pixels only due to extreme computational demands.

where Ω comprises hemisphere (aligned with surface normal) of applicable illumination directions.

6. Results

For our experiments we have used BTF data from the University of Bonn [SSK03]. Individual BTF measurements in low dynamic range (LDR) and high dynamic range (HDR) have spatial resolution 256×256 and angular resolution $|\omega_i| \times |\omega_v| = 81 \times 81$. Single BTF material in RGB for LDR data (8 bits per colour channel) takes up to 1.2 GBytes. HDR data are considered to have resolution 12 bits per colour channel (1.8 GBytes per material).

During BTF rendering for arbitrary viewing and illumination directions a linear interpolation between sampled $F_{\mathbf{x}}$ points is carried out in each code-book separately. All results presented in the paper were computed for discretization $n_\alpha = 11$, $n_\beta = 11$, $n_{\theta_v} = 7$ and $n_{\varphi_v} = 16$. We used this discretization to capture original BTF measurements accurately enough ($81 \times 81 = 6561 < 13552 = 11 \times 11 \times 7 \times 16$). We report here the results without the mipmapping described in Section 5.1.3

The model synthesis is fast as it implements conditional look-ups into stored code-books with additional interpolation for arbitrary ω_i, ω_v and conversion from YCrCb to RGB colour-space (4). We use linear interpolation between two closest values in all code-books. Implemented on a CPU, our model yields 310 000–1 360 000 BTF evaluations per second, depending on the coherence of queries. According to our comparison, it is about 1.5 times faster than the standard single-lobe Lafortune model [LFTG97] computed separately in all RGB channels and for individual ω_v [FH05]. All the tests in this section were performed on a single core of PC with the processor 2.83 GHz, Intel(R) Xeon(R) CPU, 6 MBytes L2 cache and 16 GBytes RAM DDR2 400 MHz.

The performance of the proposed model GPU implementation was tested on two graphics cards and their results for various 3D objects are shown in Table 1, side-by-side with reference speed of our CPU (single core) implementation. The implementations performs BTF decoding for a single point light. We can obtain interactive frame-rates even for complex 3D objects. Results suggest that performance on GPU is dependent on surface curvature complexity as well as on GPU texture caching algorithms.

Table 3: *The maximum sizes of code-books and achieved compression ratios for individual code-books for the proposed method. C.R. represents overall sample's compression ratio. The discretization used:* $n_\alpha = 11,\ n_\beta = 11,\ n_{\theta_v} = 7$, *and* $n_{\varphi_v} = 16$.

BTF sample	Size	C.R.	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{C}$	$\mathbf{I}_1$	$\mathbf{I}_2$	$\mathbf{M}$	$\mathbf{P}_3$	$\mathbf{P}_4$
# Uncompressed code-vectors			5.0×10^6	4.5×10^5	5.5×10^7	5.0×10^6	4.5×10^5	4.5×10^5	6.5×10^4	4.1×10^3
alu	64^2	**1:1002**	$1{:}3.7 \times 10^3$	$1{:}2.7 \times 10^3$	$1{:}2.8 \times 10^7$	$1{:}1.0 \times 10^6$	$1{:}1.1 \times 10^5$	$1{:}2.6 \times 10^2$	$1{:}1.2 \times 10^2$	$1{:}1.2 \times 10^2$
# Uncompressed code-vectors			8.1×10^7	7.3×10^6	8.8×10^8	8.0×10^7	7.3×10^6	7.3×10^6	1.0×10^6	6.6×10^4
corduroy	256^2	**1:418**	$1{:}7.1 \times 10^3$	$1{:}1.7 \times 10^2$	$1{:}3.1 \times 10^7$	$1{:}2.7 \times 10^3$	$1{:}2.0 \times 10^2$	$1{:}1.5 \times 10^2$	$1{:}8.3 \times 10^2$	$1{:}7.7 \times 10^1$
fabric1	256^2	**1:362**	$1{:}4.8 \times 10^3$	$1{:}1.0 \times 10^2$	$1{:}5.6 \times 10^7$	$1{:}9.1 \times 10^4$	$1{:}1.1 \times 10^4$	$1{:}1.0 \times 10^2$	$1{:}5.0 \times 10^1$	$1{:}5.0 \times 10^1$
fabric2	256^2	**1:710**	$1{:}7.7 \times 10^3$	$1{:}2.0 \times 10^2$	$1{:}1.0 \times 10^8$	$1{:}7.7 \times 10^4$	$1{:}1.2 \times 10^4$	$1{:}2.0 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
foil1	256^2	**1:2040**	$1{:}2.3 \times 10^4$	$1{:}9.1 \times 10^2$	$1{:}1.3 \times 10^8$	$1{:}7.7 \times 10^5$	$1{:}2.2 \times 10^5$	$1{:}9.1 \times 10^2$	$1{:}1.2 \times 10^2$	$1{:}8.3 \times 10^1$
foil2	256^2	**1:1138**	$1{:}1.1 \times 10^4$	$1{:}3.8 \times 10^2$	$1{:}9.1 \times 10^7$	$1{:}1.8 \times 10^5$	$1{:}4.0 \times 10^4$	$1{:}3.8 \times 10^2$	$1{:}9.1 \times 10^1$	$1{:}8.3 \times 10^1$
impalla	256^2	**1:522**	$1{:}3.4 \times 10^3$	$1{:}1.7 \times 10^2$	$1{:}6.2 \times 10^7$	$1{:}3.6 \times 10^4$	$1{:}1.5 \times 10^3$	$1{:}1.6 \times 10^2$	$1{:}6.2 \times 10^1$	$1{:}8.3 \times 10^1$
leather	256^2	**1:1244**	$1{:}1.3 \times 10^4$	$1{:}4.5 \times 10^2$	$1{:}1.3 \times 10^8$	$1{:}1.4 \times 10^5$	$1{:}2.4 \times 10^4$	$1{:}4.3 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
proposte	256^2	**1:806**	$1{:}1.4 \times 10^4$	$1{:}2.8 \times 10^2$	$1{:}8.3 \times 10^7$	$1{:}1.2 \times 10^4$	$1{:}7.1 \times 10^2$	$1{:}2.4 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
pulli	256^2	**1:264**	$1{:}2.4 \times 10^3$	$1{:}7.1 \times 10^1$	$1{:}1.0 \times 10^8$	$1{:}2.6 \times 10^4$	$1{:}1.5 \times 10^3$	$1{:}7.1 \times 10^1$	$1{:}5.0 \times 10^1$	$1{:}5.0 \times 10^1$
wallpaper	256^2	**1:728**	$1{:}9.1 \times 10^3$	$1{:}2.3 \times 10^2$	$1{:}4.8 \times 10^7$	$1{:}1.9 \times 10^4$	$1{:}1.0 \times 10^3$	$1{:}2.1 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
wood1	256^2	**1:352**	$1{:}1.3 \times 10^4$	$1{:}2.1 \times 10^2$	$1{:}1.6 \times 10^7$	$1{:}7.7 \times 10^2$	$1{:}1.6 \times 10^2$	$1{:}1.5 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
wood2	256^2	**1:278**	$1{:}5.3 \times 10^3$	$1{:}2.0 \times 10^2$	$1{:}1.5 \times 10^7$	$1{:}4.5 \times 10^2$	$1{:}1.4 \times 10^2$	$1{:}1.4 \times 10^2$	$1{:}8.3 \times 10^1$	$1{:}8.3 \times 10^1$
wool	256^2	**1:233**	$1{:}7.7 \times 10^3$	$1{:}1.1 \times 10^2$	$1{:}4.0 \times 10^7$	$1{:}1.3 \times 10^3$	$1{:}8.3 \times 10^1$	$1{:}7.7 \times 10^1$	$1{:}5.0 \times 10^1$	$1{:}5.0 \times 10^1$
ceiling$^\diamond$	256^2	**1:780**	$1{:}4.5 \times 10^3$	$1{:}2.3 \times 10^2$	$1{:}2.7 \times 10^7$	$1{:}3.4 \times 10^3$	$1{:}4.3 \times 10^2$	$1{:}2.2 \times 10^2$	$1{:}9.1 \times 10^1$	$1{:}6.2 \times 10^1$
floortile$^\diamond$	256^2	**1:360**	$1{:}2.0 \times 10^3$	$1{:}9.1 \times 10^1$	$1{:}2.1 \times 10^7$	$1{:}3.8 \times 10^3$	$1{:}4.5 \times 10^2$	$1{:}9.1 \times 10^1$	$1{:}5.0 \times 10^1$	$1{:}5.0 \times 10^1$
pinktile$^\diamond$	256^2	**1:2267**	$1{:}5.3 \times 10^3$	$1{:}9.1 \times 10^2$	$1{:}5.6 \times 10^7$	$1{:}1.4 \times 10^5$	$1{:}2.4 \times 10^4$	$1{:}8.3 \times 10^2$	$1{:}1.7 \times 10^2$	$1{:}9.1 \times 10^1$
walkway$^\diamond$	256^2	**1:257**	$1{:}1.0 \times 10^3$	$1{:}6.7 \times 10^1$	$1{:}2.3 \times 10^7$	$1{:}3.0 \times 10^3$	$1{:}4.2 \times 10^2$	$1{:}6.7 \times 10^1$	$1{:}5.0 \times 10^1$	$1{:}5.0 \times 10^1$

$^\diamond$ HDR sample.

Compression ratios achieved for individual BTF samples with corresponding compression times are shown in columns 2–6 of Table 2 and for individual code-books in Table 3. From the results we can conclude, that lower compression ratios correspond to textile materials having higher structural variability and complex occlusion/translucency effects, such as corduroy, impalla, proposte and pulli.

The average compression time of a BTF sample (size 256^2) using unoptimized implementation of the proposed VQ algorithm on a single CPU core, was about 23.4 hours, including BTF data resampling to the proposed parametrization. The numbers of data-vectors in individual code-books depend on the variability of BTF material, as shown in Table 3. This shows how the proposed compression model adapts to various characteristics of different BTF samples. When we compress 13 BTF LDR samples (except alu) to a shared representation, the compression ratio is increased further by a factor from 15%. When compressing 4 HDR samples to a shared representation, the compression ratio is increased by 40% (these figures are not reported in Table 3).

Images rendered using our BTF model for point light and environment lighting (Grace Cathedral, St. Peter's Basilica courtesy of Paul Debevec (http://www.debevec.org), and grassplain) are depicted in Figure 1.

The proposed BTF compression method can also be used for BRDF when the BRDF samples are understood as apparent BRDFs. We compressed 100 isotropic BRDF measured samples (courtesy of Wojciech Matusik and MERL BRDF database [Mat03]) with an original data size of each sample of $90 \times 90 \times 180 \times 3$ numbers (= 16.69 MBytes of data) for various discretizations. For example, for the discretization $n_\alpha = 91,\ n_\beta = 91,\ n_{\theta_v} = 45$ and $n_{\varphi_v} = 1$ we compressed 100 BRDF samples with a negligible average MSE error to 41 MBytes (compression ratio C.R. $\approx$ 1:42). For another setting and $n_\alpha = 45,\ n_\beta = 45,\ n_{\theta_v} = 35$ with visually indistinguishable error we can compress these BRDF samples to 11.1 MBytes (C.R. $\approx$ 1:150). The shared data in the code-books are luminance characteristics in the code-books $\mathbf{P}_1$ and $\mathbf{P}_2$.

We have measured the speed of the importance sampling algorithm using the proposed model for a single process. Given a viewing direction and the pair of random numbers we computed illumination direction at rates of 450 000–1 600 000 samples per second, depending on the coherence of queries in the spatial domain. To test the functionality of the importance sampling algorithm we attached our BTF compression framework to a CPU-based rendering system that implements ray tracing and path tracing. Examples of rendered images are shown in Figure 7. Further, in Figure 8(a) a 2D PDF slice of original data of a particular $F_{\mathbf{x}}$ is depicted for fixed ω_v viewing direction (yellow line on the left). A visualization of the compressed data and importance sampling algorithm is shown in Figure 8(b).

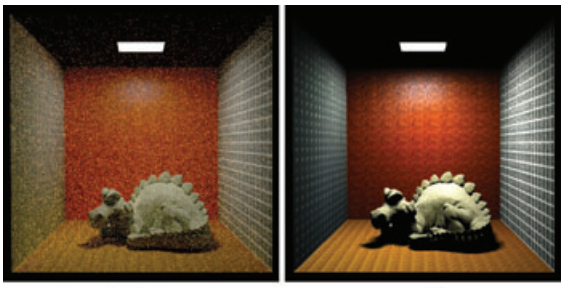


Figure 7: *Example images from software-based renderer for Phlegmatic dragon in Cornell Box: (a) path tracing with BTF model for five BTF materials (except ceiling), (b) ray casting with BTF model for five BTF materials. For path tracing 200 paths per pixel were computed, for both algorithms five shadow rays were cast towards the area light source at each bounce of eye paths.*

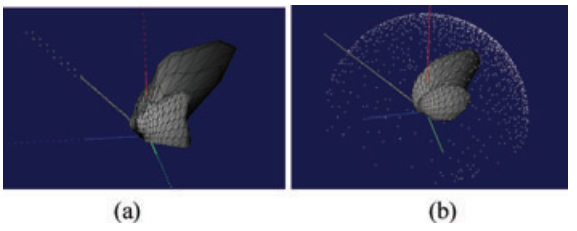


Figure 8: *Example a 2D PDF stored in* $\mathbf{P}_2$ *for the anisotropic apparent BRDF of wood. (a) original data (b) compressed data with visualization of importance sampling. Yellow line represents* ω_v*, red line normal, green line tangent and blue-line bi-tangent.*

7. Comparison with Other Methods

We have compared the proposed method in terms of data compression with three different BTF compression techniques described in Section 3: PCA of each view [SSK03] using five components/view direction (PCA for C.R. 1:14); Lafortune reflectance model [FH05] (LM with C.R. 1:16); and PCA representing BTF clusters [MMK03] using seven clusters, five components/cluster (LPCA).

While LM and PCA do not reach the compression ratio of our method, the LPCA has been proven to be efficient compression method for BTF data. Figure 9 shows compression of the most challenging BTF samples (corduroy, proposte, pulli) by means of the LPCA (left) and the proposed technique (right), compared with image rendered from original uncompressed data (middle). In average the proposed method provided subjectively comparable overall visual quality across all tested samples, however, in average provides compression ratio more than twice higher than the LPCA method (settings

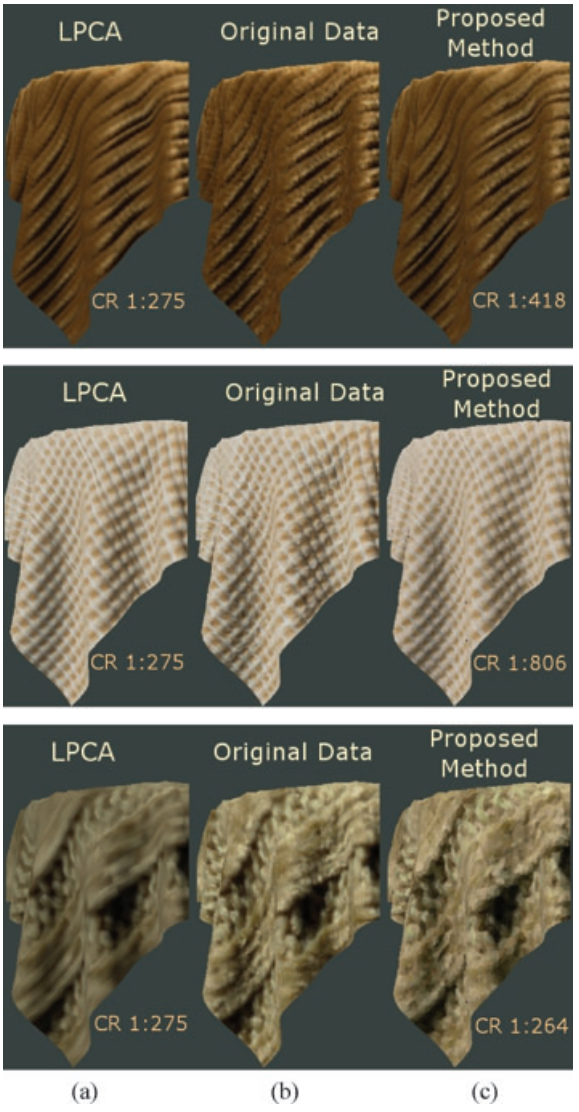


Figure 9: *Example comparison of the methods for 3 BTF samples: (top row) corduroy, (middle row) proposte, and (bottom row) pulli. (a) LPCA based compression (b) reference uncompressed data (c) the proposed method.*

seven clusters with five components per cluster – C.R. 1:275, our method on average C.R. 1:764). Let us remind that in our compression method we have used the SSIM index, which is focused on preserving overall texture structure, but cannot be calibrated precisely in terms of visibility thresholds as perceived by the human observer. Even less reliable results can be obtained using MSE and CIE LAB metrics [WB02].

To objectively compare visual fidelity of these two methods we performed a simple psychological experiment with 19 participants. The subjects with normal or corrected vision of average age 27 years were shown 14 animated sequences

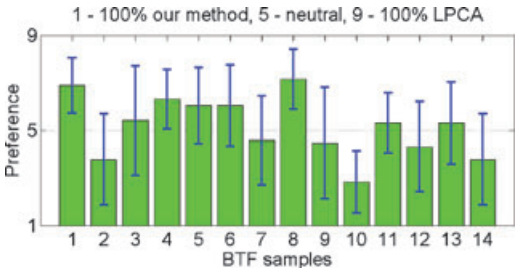


Figure 10: *Evaluation of the psychological experiment for 19 participants and 14 LDR BTF samples. The error bars represent twice the standard error across subjects.*

of the rotating tablecloth objects with mapped BTF as shown in Figure 9, i.e. the video rendered from original data always in the middle and from the compressed data by the proposed method and the LPCA, side-by-side in random order. The video for each BTF sample has been shown for 25 seconds, the whole test took between 7 and 9 minutes for each subject. The subjects' task was to evaluate which of the method provides more realistic visual experience given the reference data in the middle. For each person 14 LDR BTF samples were shown, which gave 266 individual answers.

As can be seen in Figure 10 summarizing our perceptual experiment the LPCA works better for materials with relatively small spatial appearance variations across images (please refer to a list of materials in Table 2). This is typical for such materials as alu, fabric2, foil1, foil2 and leather. Our compression allows better adaptation to complex materials having large variety of non-typical features such as corduroy, impalla, proposte, pulli, wood1 and wool. This is to be expected because our approach assumes a similarity on the level of apparent BRDFs allowing the efficient representation of irregularities thanks to the multi-level decomposition of data, while in LPCA the features are easier to represent by limited set of basis functions. The very small p-value ($p = 2.2 \times 10^{-16}$) of ANOVA test indicates that differences between samples' means are highly significant. The mean evaluation is 5.15, where 5 means undecided and 6 very low preference of the LPCA. This means that the proposed method is comparable with the tested methods in terms of visual performance. However, it has much lower memory requirements during sample analysis, it compresses each sample according to its variability, it allows more materials to be compressed efficiently into one data set, and it performs fast importance sampling.

We can also compare the proposed method based on multi-level vector quantization with standard (i.e. one-level) vector quantization, with the results are shown in the last column of Table 3. The standard vector quantization for the same parametrization reaches compression ratios up to 1:80, which is significantly lower than that achieved by the multi-level approach proposed by our technique.

8. Conclusion and Future Work

The main contribution of this paper is a novel BTF compression method based on vector quantization enabling high compression ratios between 1:233 and 1:2267 (on average 1:764) depending on material sample variability. This is further increased by 15–40% when several BTF materials are compressed to a common representation. The method can also be used for compression of multiple BRDF data. For compression of BTF samples we directly use the SSIM metric to control the estimated visual similarity between the original and the compressed data. For compression of BRDF data we can specify the maximum MSE of the compressed BRDF. The proposed algorithm can therefore efficiently control quality versus a compression ratio, and the quality metric can be changed in future to a more efficient one. In addition, the proposed method allows fast importance sampling of BTF/BRDF data, which is a desirable feature of high-quality rendering applications exploiting path tracing techniques. We verified this by implementation on a CPU.

We tested the functionality of the algorithm for 18 distinct BTF materials in HDR and LDR format, and have thoroughly compared the achieved results with results from three other recently published compression methods. High fidelity of the results was also verified against true measured data in a z-buffer based renderer for both point and environment lighting. In addition, we have implemented the BTF decoding algorithm on the standard GPU with framerates up to 170 FPS depending on the scene complexity.

The proposed BTF framework can be elaborated in several ways in future work. First, other similarity measures among apparent BRDFs that better exploit known perceptual properties of human vision can be researched. Second, when multi-spectral BTF measurements are available, we believe that our model can be simply extended by a more accurate colour models.

Acknowledgments

We would like to thank Patrick R. Green for proof-reading, Robert Herzog, Michal Haindl, Petr Somol, Jan Přikryl, Bedřich Beneš, Michael J. Chantler, and Igor Vajda for their comments and support. Further, we would like to thank BTF database Bonn for providing their data. This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program MSM 6840770014 and LC-06008 (Center for Computer Graphics), the Aktion Kontakt OE/CZ grant no. 2009/6, grant GAČR 102/08/0593, grant MŠMT 1M0572 (DAR), and EC Marie Curie ERG No 239294.

References

[CBC*01] Carr J., Beatson R., Cherrie J., Mitchell T., Fright W., McCallum B., Evans T.: Reconstruction and representation of 3D objects with radial basis functions.

In *ACM SIGGRAPH 2001* (New York, NY, USA, 2001), ACM Press, pp. 67–76.

[Dal93] DALY S.: The visible differences predictor: an algorithm for the assessment of image fidelity. In *Digital. Images and Human Vision*. A. B. Watson (Ed.), MIT Press, Cambridge, MA, 1993, pp. 179–206.

[DBB03] DUTRÉ P., BEKAERT P., BALA K.: *Advanced Global Illumination*. A. K. Peters (Ed.), Natick, Massachusetts, 2003.

[DF97] DEYOUNG J., FOURNIER A.: Properties of tabulated bidirectional reflectance distribution functions. In *Proceedings of the Graphics Interface 1997 Conference* (1997), pp. 47–55.

[DLHS01] DAUBERT K., LENSCH H. P. A., HEIDRICH W., SEIDEL H.-P.: Efficient cloth modeling and rendering. In *Proceedings of Rendering Techniques 2001* (June 2001), Springer Verlag, Berlin, pp. 63–70.

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

[EBJ*06] EDWARDS D., BOULOS S., JOHNSON J., SHIRLEY P., ASHIKHMIN M., STARK M., WYMAN C.: The halfway vector disk for BRDF modeling. *ACM Transactions on Graphics*, *25*, 1 (2006), 1–18.

[FCGH08] FILIP J., CHANTLER M., GREEN P., HAINDL M.: A psychophysically validated metric for bidirectional texture data reduction. *ACM Transactions on Graphics*, *27*, 5 (December 2008), 138.

[FH05] FILIP J., HAINDL M.: Efficient image based bidirectional texture function model. In *Texture 2005* (October 2005), Heriot-Watt University, pp. 7–12.

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

[Fis96] FISHMAN G. S.: *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, New York, NY, 1996.

[GG92] GERSHO A., GRAY R. M.: *Vector Quantization and Signal Compression. Communications and Information Theory*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[HDS03] HAVRAN V., DMITRIEV K., SEIDEL H.-P.: Goniometric diagram mapping for hemisphere. In *Short Presentations (Eurographics 2003)*, (2003), pp. 293–300.

[Hel09] HELLINGER E.: Neue begründung der theorie quadratischen formen von unendlichen vielen veränderlichen. *Journal für die reine und angewandte Mathematik*, *136* (1909), 210–271.

[HF07] HAINDL M., FILIP J.: Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 10 (October 2007), 1859–1865.

[KM06] KAWAI N., MATSUFUJI K.: Azimuth-rotated vector quantization for BTF compression. In *GRAPHITE 2006, ACM SIGGRAPH Research posters* (2006).

[KMBK03] KOUDELKA M., MAGDA S., BELHUMEUR P., KRIEGMAN D.: Acquisition, compression, and synthesis of bidirectional texture functions. In *Texture 2003* (October 2003), pp. 47–52.

[Lar98] LARSON G. W.: LogLuv encoding for full-gamut, high-dynamic range images. *Journal of Graphics Tools: JGT*, *3*, 1 (1998), 15–31.

[LFTG97] LAFORTUNE E. P., FOO S. C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Computer Graphics 31, Annual Conference Series* (1997), pp. 117–126.

[LGC*05] LENSCH H., GÖSELE M., CHUANG Y.-Y., HAWKINS T., MARSCHNER S., MATUSIK W., MÜLLER G.: Realistic materials in computer graphics. In *SIGGRAPH 2005 Tutorials* (2005).

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

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics*, *23*, 3 (2004), 496–505.

[LRR05] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Adaptive numerical cumulative distribution functions for efficient importance sampling. In *Proceedings of Eurographics Symposium on Rendering 2005* (June 2005), pp. 11–20.

[LZT*04] LIU X., ZHANG J., TONG X., GUO B., SHUM H.-Y.: Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, *10*, 3 (June 2004), 278–289.

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

[MCC*04] MA W.-C., CHAO S.-H., CHEN B.-Y., CHANG C.-F., OUHYOUNG M., NISHITA T.: An efficient representation of complex materials for real-time rendering. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, November 2004), ACM, pp. 150–153.

[MCT*05] MA W.-C., CHAO S.-H., TSENG Y.-T., CHUANG Y.-Y., CHANG C.-F., CHEN B.-Y., OUHYOUNG M.: Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics* (New York, NY, USA, April 2005), ACM, pp. 187–194.

[MGW01] MALZBENDER T., GELB D., WOLTERS H.: Polynomial texture maps. In *ACM SIGGRAPH 2001* (New York, NY, USA, 2001), ACM Press, pp. 519–528.

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

[MLH02] MCALLISTER D. K., LASTRA A., HEIDRICH W.: Efficient rendering of spatial bi-directional reflectance distribution functions. In *Graphics Hardware* (2002), pp. 77–88.

[MMK03a] MESETH J., MÜLLER G., KLEIN R.: Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium 2003* (April 2003), Eurographics Association, Switzerland, pp. 89–96.

[MMK03b] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (November 2003), pp. 271–280.

[MMS*04] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis and rendering of bidirectional texture functions. In *Proceedings of Eurographics 2004 State of the Art Reports* (September 2004).

[MOV94] MICÓ L., ONCINA J., VIDAL E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, *15*, 1 (1994), 9–17.

[ND06] NGAN A., DURAND F.: Statistical acquisition of texture appearance. In *Eurographics Symposium on Rendering 2006* (June 2006), pp. 31–40.

[NJH*77] NICODEMUS F. J. C. R., HSIA J., GINSBURG I., LIMPERIS T.: Geometrical considerations and nomenclature for reflectance. In *NBS Monograph 160, National Bureau of Standards* (October 1977), pp. 1–52.

[RFS03] RIGAU J., FEIXAS M., SBERT M.: Refinement criteria based on F-divergences. In *Proceedings of Eurographics Symposium on Rendering 2003* (June 2003), pp. 260–269.

[RK09] RUITERS R., KLEIN R.: BTF compression via sparse tensor decomposition. *Computer Graphics Forum*, *28*, 4 (2009), 1181–1188.

[Rus98] RUSINKIEWICZ S.: A new change of variables for efficient BRDF representation. In *Proceedings of Eurographics Workshop on Rendering* (1998), pp. 11–22.

[Sam06] SAMET H.: *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, San Francisco, CA, 2006.

[SAS05] STARK M. M., ARVO J., SMITS B.: Barycentric parameterizations for isotropic BRDFs. *IEEE Transactions on Visualization and Computer Graphics*, *11*, 2 (2005), 126–138.

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Eurographics Symposium on Rendering 2003* (2003), pp. 167–177.

[TG74] TOU J., GONZALEZ R.: *Pattern Recognition Principles*. Addison-Wesley, Reading, MA, 1974.

[TZL*02] TONG X., ZHANG J., LIU L., WANG X. GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. In *ACM SIGGRAPH 2002* (New York, NY, USA, 2002), ACM Press, pp. 665–672.

[VT04] VASILESCU M., TERZOPOULOS D.: TensorTextures: Multilinear image-based rendering. *ACM SIGGRAPH 2004 23*, 3 (New York, NY, USA, August 2004), ACM Press, 336–342.

[WB02] WANG Z., BOVIK A. C.: A universal image quality index. *IEEE Signal Processing Letters*, *9*, 3 (March 2002), 81–84.

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*, 4 (2004), 600–612.

[Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH Computer Graphics*, *17*, 3 (1983), 1–11.

[WL03] WONG T., LEUNG C.: Compression of illumination-adjustable images. *IEEE Transactions on Circuits and Systems for Video Technology*, *13*, 11 (November 2003), 1107–1118.

[WLB04] WANG Z., LU L. G., BOVIK A. C.: Video quality assessment based on structural distortion measurement. *Journal of Signal Processing: Image Communications*, *19*, 2 (February 2004), 121–132.