EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

A Sparse Parametric Mixture Model for BTF Compression, Editing and Rendering

Hongzhi Wu Julie Dorsey Holly Rushmeier

Computer Graphics Group, Yale University

Abstract

Bidirectional texture functions (BTFs) represent the appearance of complex materials. Three major shortcomings with BTFs are the bulky storage, the difficulty in editing and the lack of efficient rendering methods. To reduce storage, many compression techniques have been applied to BTFs, but the results are difficult to edit. To facilitate editing, analytical models have been fit, but at the cost of accuracy of representation for many materials. It becomes even more challenging if efficient rendering is also needed. We introduce a high-quality general representation that is, at once, compact, easily editable, and can be efficiently rendered. The representation is computed by adopting the stagewise Lasso algorithm to search for a sparse set of analytical functions, whose weighted sum approximates the input appearance data. We achieve compression rates comparable to a state-of-the-art BTF compression method. We also demonstrate results in BTF editing and rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

The bidirectional texture function (BTF) was introduced by Dana et al. [DvGNK99] to capture the appearance of complex materials. It is essentially a 6D function parameterized by position, lighting direction as well as view direction. BTFs can represent a wide range of materials, from simple plastic to carpeting, which has complex non-height-field meso-structure and spatially varying optical properties. Various effects like inter-reflection, self-shadowing and masking are incorporated in the representation, which makes rendering using a BTF extremely realistic. However, in practice there are three major shortcomings with BTFs: huge storage requirements due to their high dimensional nature, the lack of intuitive editing methods, and the challenge to efficiently render them. Previous work has made significant progress in removing one or the other limitation, but not all.

In comparison, many BRDFs can be well approximated using analytical models [DRS07] with compact storage. The parameters of these models usually have physical meanings (e.g. controlling the specularity), which allows intuitive direct manipulations. Moreover, many of these models offer analytical importance sampling for efficient rendering in a Monte-Carlo setting. Unfortunately, it has been known that analytical BRDF models cannot faithfully represent BTF data [MMS*05].

What is missing for efficiently representing BTFs, is a high-quality general parametric model, which is, *at once*, compact, easily editable and can be efficiently rendered. In this paper, we propose a Sparse Parametric Mixture Model (SPMM) for general BTFs, in an effort to complete this missing piece. Unlike previous work which fits just one analytical BRDF model, we use a sparse linear combination of *rotated* analytical BRDFs of different types to fit each BTF texel. The resultant approximation error is small and coherent, which can be further reduced with modest additional space. Since we are using analytical models, we can represent specular materials beyond the angular resolution of any data-driven representation. Furthermore, the sparseness property gives us a small set of analytical BRDFs, which are easily editable. We handle general BTFs, as our method does not assume height-field, meso-structure geometry as some previous work does (e.g. [MG09]). Please refer to Fig. 1 for an illustration of our pipeline.

The key contribution of our paper is a compact, editable,

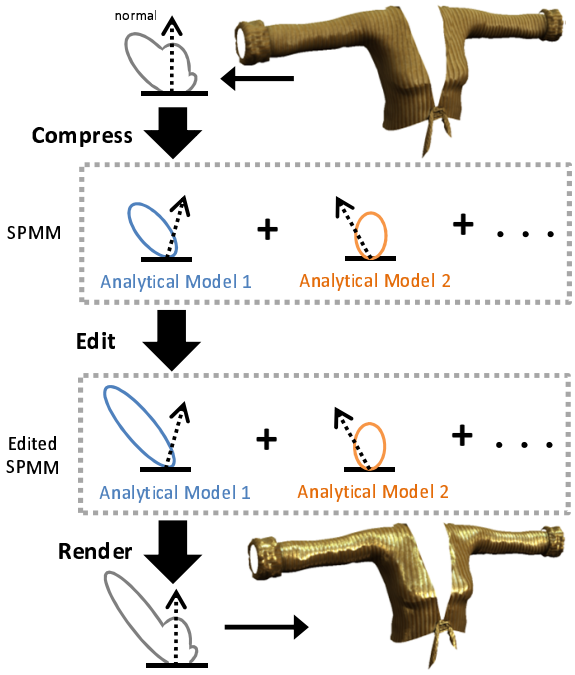


Figure 1: *A diagram of our pipeline. For each BTF texel, we first compress it into our SPMM representation, consisting of a sparse linear combination of rotated analytical BRDF models. Next, we can perform various editing operations by manipulating this representation. For example, we have narrowed the lobe of Analytical Model 1 in the above figure. The result is then rendered by summing the linear combination of all analytical models. In the above example, we have added a metallic look to the original material.*

yet efficiently renderable, high-quality parametric representation for general 6D appearance data, which can be used to represent BTFs analogous to the way analytical BRDF models are used to represent measured BRDF data. In addition, we present the first algorithm to our knowledge to fit multiple, rotated analytical BRDFs of different types, based on a recent advance in machine learning. This algorithm could also be used in other challenging settings to express a signal as a sparse linear combination of non-orthogonal parametric basis functions, where the basis function types, the function parameters, as well as the weights are all unknown.

2. Previous Work

The BTF has received a great deal of attention since it was first introduced by Dana et al. [DvGNK99]. A comprehensive survey of the acquisition, synthesis and rendering of BTFs was presented by Müller et al. [MMS*05], several methods for compressing BTFs are described. One class of methods is fitting analytical BRDF models such as Lafortune lobes. The results have meaningful parameters for editing, and they are suitable for fast rendering. Unfortunately, such methods are accurate representations only for materials with meso-scale geometry with very small variations in height. Alternative methods use numerical compression based on matrix factorization. While a high compression rate is achieved, the results cannot be readily edited.

A subsequent survey by Filip and Haindl [FH09] includes additional work on BTF: Ma et al. [MCT*05] developed a compact representation using PCA and parametric fitting of Phong models on BTF data after a Laplacian transformation. The result is suitable for level-of-detail (LOD) rendering, but not for editing. Filip et al. [FCGH08] increased the numerical compression efficiency by ignoring the perceptually unimportant features of BTF data. Guthe et al [GMSK09] proposed a perceptual metric, which can be applied to a variety of compression techniques. Both methods are based on numerical compression and thus do not produce directly editable results.

Recently, Ruiters and Klein [RK09] used L1-constrained fitting to obtain a sparse set of numerical basis functions for BTF compression. While they achieve a high compression rate, the numerical basis functions do not have physical meanings and cannot be directly edited. Havran et al. [HFM10] decomposed the BTF data into multi-dimensional conditional probability density functions, which are then encoded using vector quantization. Impressive levels of compression are achieved. While excellent for importance sampling in rendering, this representation again does not lend itself to appearance editing.

Kautz et al. [KBD07] performed various operations directly on the raw BTF data to achieve interesting effects. Since the large size of the raw BTF does not fit in memory, an out-of-core architecture is carefully-designed to efficiently manage I/O data flow. Xu et al. [XWT*09] presented an improved method of propagating edits through the images representing the raw BTF.

In work that is similar in spirit, Menzel and Guthe [MG09] represented a BTF as the combination of an estimated meso-structure and fitted Ashikhmin's distribution BRDFs [AP07]. Although this approach allows intuitive editing, the major limitations are the height field assumption and a distribution BRDF of a single normal at each texel, which cannot approximate general BTFs with high quality. In comparison, our method is derived mathematically to represent general BTFs, and we do not assume height-field meso-structure.

Lawrence et al. [LBAD*06] proposed the Inverse Shade Tree (IST), a data-driven representation that is both compact and editable, for spatially-varying BRDFs (SV-BRDFs). By comparison, our representation differs from theirs in three key respects. First, similar to [MG09], [LBAD*06] also assumes one local frame at each texel. This is very limiting in terms of expressive power, we instead allow a different local frame for each lobe in an SPMM. Since the BTF is more complex and general than the SV-BRDF, it is unknown whether IST will generate efficient representations

Symbol	Description
n	a normal
b_x	a BTF at the point x
s_x	an SPMM
ε_x	a residual function
ρ_j	a parametric basis function
α_j	the weight of ρ_j in an SPMM
$\bar{\alpha}_j$	the weights for RGB channels
$f_j(\kappa_j,\cdot)$	an analytical BRDF model, κ_j are the model parameters
β_j	the combination of f_j, κ_j and a local frame
y	a cosine-weighted BTF texel
η	a step size in stagewise Lasso
$\mathcal{D}$	a dictionary of basis functions
λ	a regularization parameter
μ	a residual function in stagewise Lasso
S / $\bar{S}$	specular / non-specular lobes
γ	the probability of selecting a lobe in importance sampling S

Table 1: *Summary of the notation used in this paper.*

for BTFs. Second, since the SPMM is based on analytical models, we can directly edit model parameters and perform importance sampling analytically, neither of which is available in IST. Third, in representing highly specular materials, the analytical-model-based SPMM could be much more compact than a data-driven method like [LBAD*06].

3. A Sparse Parametric Mixture Model

3.1. Definitions

We start our derivation from the rendering equation for a BTF at a surface point x [MMS*05]:

$$L_r(x,\omega_o) = \int_\Omega L_i(x,\omega_i) b_x(\omega_i,\omega_o)(n_x\cdot\omega_i)d\omega_i, \quad (1)$$

where L_r denotes the reflected radiance, L_i denotes the incoming radiance, ω_i is the lighting direction and ω_o is the view direction. b_x denotes the BTF, n_x is the surface normal, $(\cdot)$ is the cosine term, and Ω is the upper hemisphere over x.

Next we represent the cosine-weighted BTF term in Eq. (1) using our Sparse Parametric Mixture Model:

$$b_x(\omega_i,\omega_o)(n_x\cdot\omega_i) = \sum_{j=1}^{m} \alpha_j\rho_j(\omega_i,\omega_o) + \varepsilon_x(\omega_i,\omega_o). \quad (2)$$

The first term in the right-hand side of the above equation is a linear combination of parametric functions $\{\rho_j\}$ with corresponding weights $\{\alpha_j\}$. ε_x is a residual function. The idea for using SPMM is that by employing a sparse linear combination of carefully-chosen parametric functions, we can compactly represent the original BTF, provide meaningful editing operations and allow efficient rendering as well. Previous methods (e.g. [MG09]), which fit only one analytical BRDF model, can be viewed as a special case of our model, where $m = 1$.

In our work, each parametric function ρ_j is defined as a cosine-weighted rotated BRDF:

$$\rho_j(\omega_i,\omega_o) = f_j(\kappa_j, R(\omega_i), R(\omega_o))(n_j\cdot\omega_i). \quad (3)$$

Here, R is a rotation, which transforms a vector from the global coordinate system to a local frame, $f_j(\kappa_j,\cdot)$ is one analytical BRDF model, with κ_j as its model parameters. n_j is the normal of the local frame. In this paper, we employ seven classic analytical BRDF models - Lambertian, Oren-Nayar, Blinn-Phong, Ward, Cook-Torrance, Lafortune and Ashikhmin-Shirley (see [NDM05]) - as candidates for f_j. Please see Fig.2 for an illustration of an SPMM representation. We note that SPMM is a general concept and is not limited to the aforementioned models. It is possible to use other analytical BRDF models for f_j, or even a different formulation of ρ_j from Eq. (3) for different applications. To simplify the notation, for a particular ρ_j, we denote the combination of the model type of f_j, the corresponding model parameters κ_j and its local frame (normal only if the analytical model is isotropic), as β_j. Note that for a given β_j, ρ_j is uniquely determined.

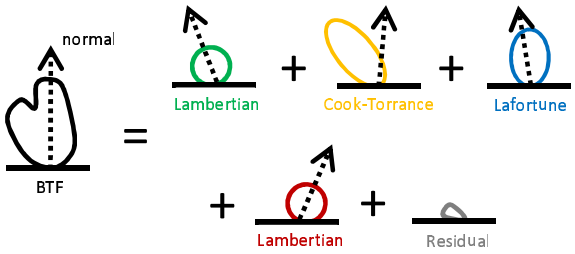


Figure 2: *A BTF slice represented using an SPMM. Two Lambertian, one Cook-Torrance and one Lafortune model are used in this example. Each analytical model has its own local frame.*

Although we derive our model mathematically, one may view SPMM as an approximate heterogeneous microfacet-based reflectance model. Each ρ_j describes a different apparent reflectance function of a microfacet oriented towards n_j. Effects like shadowing, masking and interreflection, which are not representable by the sparse linear combination of rotated BRDFs, are included in the residual ε_x. Note that in Eq. (2), our model approximates the cosine-weighted b_x, rather than b_x alone. This is because our representation implicitly has a normal distribution formed by various microfacets, instead of a single normal.

3.2. The Fitting Method

Challenges arise when we begin to design a fitting algorithm for an SPMM: the number of parametric functions, m, is unknown; the nonlinear parameters $\{\beta_j\}$ (including the normals $\{n_j\}$) are unknown; the corresponding weights $\{\alpha_j\}$

are also unknown. Existing algorithms for fitting analytical BRDF models (e.g. the Levenberg-Marquardt (LM) algorithm in [LFTG97]) cannot be used here for several reasons. First, the number of lobes is not known a priori, which is required as input to the LM algorithm. Second, it becomes more numerically unstable and more computationally expensive to fit for more than three lobes using the LM algorithm. But it is undesirable to limit the number of parametric functions in an SPMM. Third, in SPMM we are using BRDFs rotated according to each one's individual local frame, which has more degrees of freedom than classic BRDF fitting, where only one local frame is assumed. Fourth, there is no means to tradeoff between the sparsity and the fitting quality. Last but not least, previous algorithms fit for one type of analytical BRDF models only, while in SPMM, each ρ_j could be chosen from a set of different analytical models.

To tackle the above difficulties, we adopt the stagewise Lasso algorithm [ZY07] from the machine learning community. Stagewise Lasso computes an approximate optimal solution to the following sparsity-constrained fitting problem:

$$\underset{\alpha_i,\rho_i\in\mathcal{D}}{\operatorname{argmin}}||y-\sum_i \alpha_i\rho_i||^2+\lambda\sum_i|\alpha_i|. \tag{4}$$

Here, ρ_i is a basis function chosen from a predefined dictionary $\mathcal{D}$, and α_i is its weight. λ is a regularization parameter, which controls the balance between sparsity and the L2-norm fitting quality of the final results. Unlike some other related algorithms (e.g. Basis Pursuit [CDS98]), which only work for finite dictionaries, stagewise Lasso also works for infinite dictionaries. This is useful in our case, since the number of possible analytical reflectance functions with different model parameters is infinite.

We briefly describe our stagewise-Lasso-based fitting algorithm. For more details, please refer to the original paper [ZY07]. Assume that we are fitting an SPMM to a cosine-weighted BTF texel, denoted as y. Initially, a residual function μ is assigned as y. At each iteration, the algorithm finds a parametric function, which, after normalization, has the maximum dot product with the current μ. Then its corresponding weight α is increased by a small constant step size η. The weight is decreased if a backward step condition is satisfied. The residual function μ is updated at the end of each iteration. The algorithm is terminated if a computed regularization parameter λ is less than a user-specified value. Note that to find a parametric function ρ, which is best correlated with μ at each iteration, we employ IPOPT [NWW08], an interior-point based nonlinear numerical optimizer, to solve the following problem:

$$\underset{\rho,\beta}{\operatorname{argmax}}\frac{\langle\rho(\beta,\cdot),\mu\rangle}{||\rho(\beta,\cdot)||\,||\mu||}. \tag{5}$$

In practice, we test all analytical models and choose the one that yields the maximum dot product between normalized ρ and μ, along with the model parameters β. Physical constraints (e.g. n must reside in the upper hemisphere of y, α must be non-negative, etc.) are also taken into account in the entire process, so that we always end up with a feasible solution.

After we obtained the basis functions $\{\rho_j\}$ and their weights $\{\alpha_j\}$, we perform a hard-thresholding to eliminate the basis functions with very small weights, in order to avoid over-fitting. A typical threshold is 2η. Then, $\{\alpha_j\}$ are recomputed using the non-negative least square method [LH74], which exploits the remaining basis functions.

It is straightforward to extend our aforementioned algorithm to handle color information. For each $\{\rho_j\}$, we keep three weights $\bar{\alpha}_j=\{\alpha_j^r,\alpha_j^g,\alpha_j^b\}$ for three color channels. When searching for a best-correlated parametric function, we separately process each channel and record the one with the best correlation as the result. The rest of the algorithm remains unchanged.

4. BTF Compression

4.1. The Algorithm

The fitting method for one BTF texel is computationally expensive, since in each step we need to run the time-consuming nonlinear numerical optimization to find the best correlated parametric function. Directly applying the algorithm to each texel of a BTF would require a prohibitively long computation. To efficiently compress an entire BTF, we first exploit spatial coherence by doing a k-means clustering over all texels, similar to [MMK03]. Next, for each cluster, we sample a number of texels in a stratified fashion: we first generate sub-clusters using a further k-means clustering inside the current cluster; then we randomly sample a texel from each sub-cluster. After that, we run the fitting algorithm over these samples and take the union of all resultant basis functions as a dictionary for the entire cluster. Finally, a modified finite-dictionary version of stagewise Lasso is used to fit each texel in the cluster. This variant of the original fitting method is much faster, since it does not perform the expensive nonlinear optimization and only chooses basis functions from the dictionary we just computed.

The idea behind our compression algorithm is that after the initial k-means clustering, the texels in one cluster are close to each other. By doing a subsequent stratified sampling, the intra-cluster variation is also captured. The union of all basis functions computed from the stratified samples is a suitable dictionary for fitting all texels in the same cluster. We demonstrate our results in the next sub-section.

Once SPMM representations are fit for one cluster, we compute the average error function between the SPMM approximation and the original data for all texels in the current cluster, and use it as a common residual function ε [see Eq. (2)] to further improve fitting quality. The extra storage for

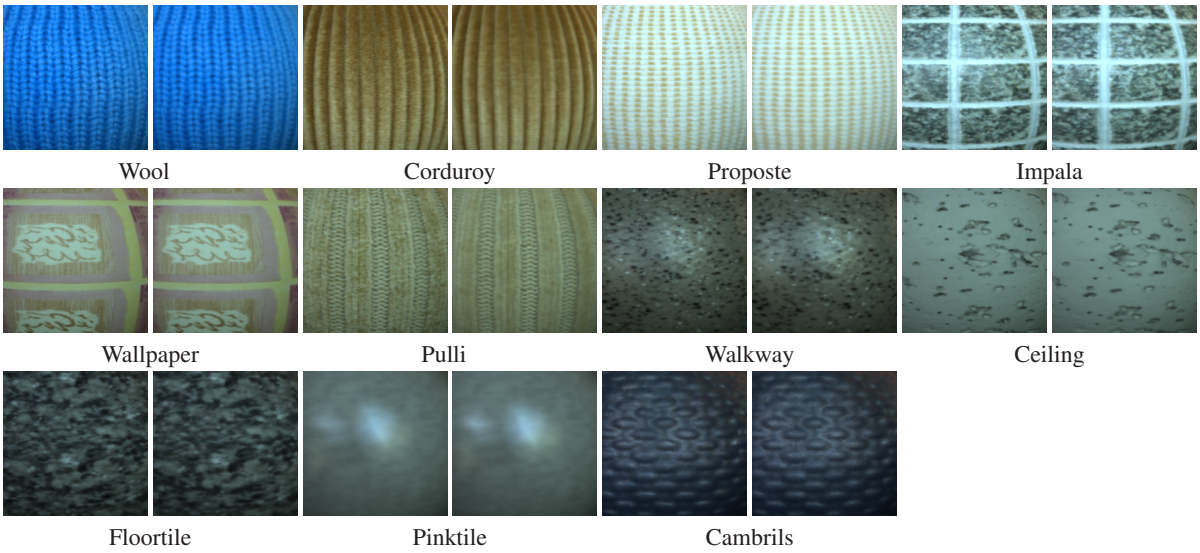


Figure 3: *Rendering comparisons. In each image pair for one material, the left image is the original BTF, and the right one is our representation.*

ε is modest as we only need to numerically store one additional function for a cluster, rather than for each texel. One approach would be to store a few additional basis error functions computed from PCA along with their coefficients for each texel. However, in experiments, we have found that this approach does not improve the quality of fitting too much, while the additional storage increases. So we simply use the single average error function for the whole cluster.

4.2. Results

We conducted experiments on a workstation with an Intel 2.4GHz quad-core processor and 4GB memory. All images are rendered using our own unoptimized Monte-Carlo ray-tracer, with 4 eye rays per pixel, based on Eqs. (1) and (2). All original BTF data (except for *cambrils*), which are from the University of Bonn [SSK03], have a spatial-angular resolution of $256^2 \times 81^2$, and three 8/12-bit color channels in RGB, which takes up 1.2GB/1.8GB. The material *cambrils*, with credit to PSA Peugeot Citroen, has a spatial-angular resolution of $72^2 \times 151^2$, and each of its samples takes up 16 bits per color channel. In all our experiments, we use $k = 32$ in the initial k-means clustering and 8 samples in stratified sampling for each cluster. We use 32-bit single-precision floats and 32-bit integers to store all output coefficients of our algorithm. No further quantization is performed. Similar to [HFM10], we measure the approximation quality of our representation using MSSIM [WBS*04] in YCrCb space for all materials. The range of MSSIM is [0.0, 1.0], where a value of 1.0 indicates that the images are identical. We show the rendered images using the original BTFs and our representations in Fig. 3. Please also see the accompany video for a comparison.

The precomputation time varies from 9∼21 hours, though this is a one-time process for a BTF. Our sparsity-constrained fitting algorithm picks up 4∼14 basis functions on average for each BTF texel. The fitting quality is further improved by adding one additional per-cluster residual function ε. The Peak Signal-to-Noise Ratio (PSNR) with and without using ε are shown in Tab. 2. For each BTF from the University of Bonn, the storage requirement of ε is $32 \times 81^2 \times 3 \times 4 = 2.40$MB (8.35MB for the material *cambrils*).

We compare our results with one state-of-the-art BTF compression method [HFM10] in Tab. 2. Our representation takes 6∼17MB for different materials, achieving a compression ratio of 1:71∼1:303. While the ratio is comparable to the results from [HFM10], we obtain a considerably higher approximation quality, as shown in Tab. 2. Furthermore, direct editing of BTF is not possible on their representation. Note that to make a fair comparison, we compare our compression rate with C.R.1 in [HFM10], where 32-bit integers and floats are also used. It would be an interesting future avenue to apply similar quantization techniques to further compress our representation and do a comparison in terms of compression rate and approximation quality. For more details about our experimental statistics, please refer to Tab. 2.

5. BTF Editing

In this section, we demonstrate how to achieve various editing effects, via three simple forms of manipulations to our representation. Please note that we focus on the less explored BTF angular domain editing. For spatial domain editing, existing techniques (e.g. [AP08]) can be used orthogonal to our method. We believe that one could achieve more editing ef-

BTF	Storage					Approximation Quality				Compression Time (hrs)
	Total basis #	Avg basis # / texel	Our size (MB)	Our C.R.	MLVQ C.R.1	PSNR (w/o ε)	PSNR (with ε)	Our MSSIM	MLVQ MSSIM	
Wool	3632	9	12.53	**1:98**	1:77	14.43db	20.27db	0.904	0.684	10.49
Corduroy	5115	14	17.22	1:71	1:128	11.56db	19.60db	0.920	0.748	14.75
Proposte	950	5	8.57	1:144	1:236	16.51db	23.67db	0.936	0.710	8.65
Impala	5397	8	11.76	1:105	1:162	17.82db	21.91db	0.934	0.730	14.19
Wallpaper	3353	6	9.22	1:133	1:222	17.66db	23.94db	0.941	0.776	12.42
Pulli	2057	5	8.88	**1:138**	1:87	13.62db	17.86db	0.883	0.699	10.59
Walkway$^{\diamond}$	3879	6	9.53	**1:194**	1:102	12.48db	19.78db	0.980	0.884	21.33
Ceiling$^{\diamond}$	2096	3	6.08	**1:303**	1:235	13.68db	22.42db	0.971	0.711	12.86
Floortile$^{\diamond}$	4352	4	7.43	**1:248**	1:136	15.95db	19.54db	0.927	0.772	15.49
Pinktile$^{\diamond}$	704	6	9.32	1:198	1:711	10.00db	32.42db	0.999	0.961	14.69
Cambrils*	2255	5	8.86	1:76	n/a	8.58db	13.16db	0.966	n/a	18.72

$^{\diamond}$: HDR sample, 12 bits/channel. * : HDR sample, 16 bits/channel. C.R. : Compression rate.

Table 2: *Various statistics from our experiments, with comparisons to Multi-Level Vector Quantization (MLVQ) [HFM10]. We achieve considerably higher approximation quality, while our compression rates are comparable to MLVQ.*

fects than the ones listed here, by creatively exploiting basic editing operations.

5.1. Adjusting the Weights $\{\bar{\alpha}_j\}$

One simple edit performed by adjusting $\{\bar{\alpha}_j\}$ is to change the overall intensity of the BTF: we just multiply all $\{\bar{\alpha}_j\}$ with the same constant c. However, to change the hue/saturation requires more effort, since these two quantities are not linear in RGB space, unlike the intensity. In other words, applying hue/saturation changes to $\bar{\alpha}_j$ of each lobe in an SPMM is not equivalent to applying the same change to the linear combination of these lobes, due to the nonlinearity. To address this issue, we compute one optimal linear approximation M, which is a 3×3 matrix, for each possible hue/saturation change e, based on the following equation:

$$\underset{M}{\operatorname{argmin}} \sum_{\forall c} ||e(c) - Mc||^2. \tag{6}$$

Then, we can multiply M with every $\bar{\alpha}_j$ to approximate the editing operation e:

$$\begin{aligned} e(\sum_{j=1}^{m} \bar{\alpha}_j \rho_j(\omega_i,\omega_o)) &\approx M \sum_{j=1}^{m} \bar{\alpha}_j \rho_j(\omega_i,\omega_o) \\ &= \sum_{j=1}^{m} M\bar{\alpha}_j\, \rho_j(\omega_i,\omega_o). \end{aligned} \tag{7}$$

In our experiments, we precompute M for $10,000$ different hue/saturation changes, using least squares. When computing each M according to Eq. (6), we sample $10,000$ unit vectors as c from the RGB color space. All precomputed M are only 352KB, and can be applied to any material instantly in interactive editing. We can see in Fig. 4 that the results using our linear approximation is close to directly applying the editing as a post-processing after evaluating the original SPMM.

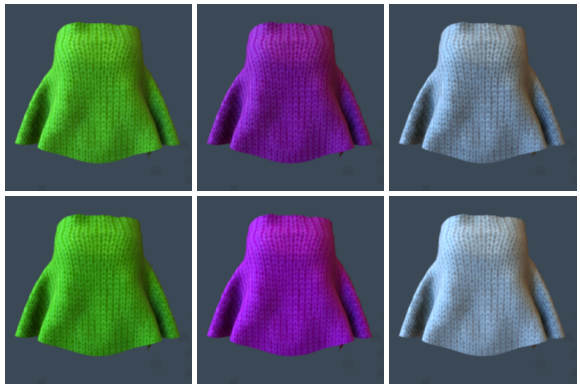


Figure 4: *Changing the hue/saturation of wool (original appearance shown in Fig. 3). Top row: our linear approximated hue/saturation editing. Bottom row: post-processing. From the left to the right column, shifting the hue by -110°, shifting the hue by 116° and applying desaturation.*

Another editing effect is to adjust specular/non-specular intensity. Thanks to our representation, we can discriminate specular/non-specular parametric BRDFs based on the following simple criterion: Lambertian and Oren-Nayar are classified as non-specular; all other models are classified based on the coefficients that control the specularity. For example, if the exponential parameter of a Blinn-Phong model is larger than an empirical threshold, the model is classified as specular; otherwise, we regard it as non-specular. Based on this criterion, we can edit the weights for all specular or non-specular $\{\rho_j\}$ separately to create various effects. In Fig. 5, we first increase the weights of all specular lobes to make the material look more specular. Then using Eq. (7), we multiply a precomputed M to all $\bar{\alpha}_j$ of specular lobes, in order to change the specular hue.

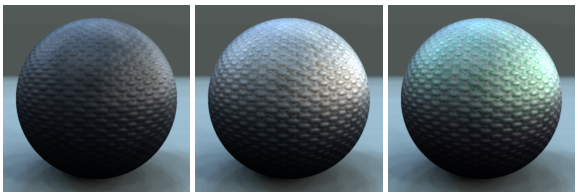
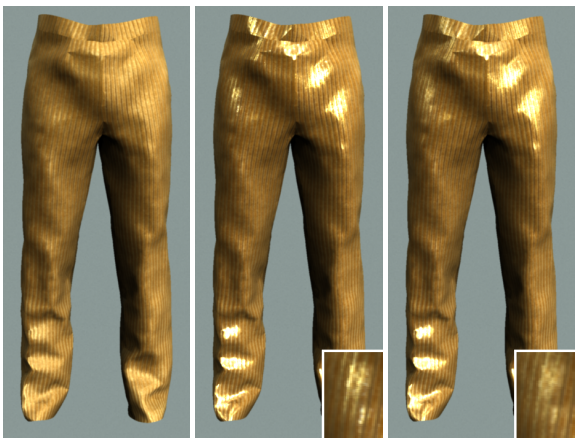


Figure 5: *Increasing specular intensity and changing specular hue. Left: original. Middle: increased specular intensity. Right: specular hue changed to green.*

5.2. Adjusting the Model Parameters $\{\beta_j\}$

We show how to create a more metallic look by adjusting $\{\beta_j\}$. Using the specular criterion described in previous subsection, we first pick up specular $\{\rho_j\}$ from our SPMM. Next, we adjust the coefficients for each specular ρ_j so that the specular lobe becomes narrower, yielding a much more metallic look to the material, as shown in Fig. 6(b). One additional benefit of using SPMM is that we are not limited to the angular resolution of the original BTF, in analogy to representing BRDF data using analytical models. This is particularly useful for faithfully representing highly specular materials. In Fig. 6(c), we can see that representing the edited material using the original BTF format results in more blurry highlights.



(a) Original (b) More metallic (c) Original format

Figure 6: *Narrowing specular lobes. (c) is representing (b) in the original BTF format. The highlight looks more blurry compared to (b).*

5.3. Adjusting the Normal Distribution

Recall from Sec. 3.1 that our representation implicitly forms a normal distribution. Actually, we can adjust this normal distribution to facilitate roughness editing. In Fig. 7, for each ρ_j, we change its normal by decreasing its elevation angle. Overall, the normal distribution is more spread out, which gives a rougher / more embossed look, as shown in the image on the right.

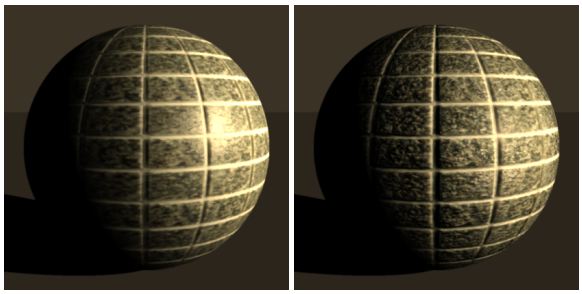


Figure 7: *Changing the normal distribution. Left: original. Right: increased roughness.*

5.4. User Interface

We have implemented an interactive material editing prototype system, as shown in Fig. 8, to facilitate editing with visual feedback. The SPMMs in editing are rendered on an object under user-specified lighting configuration and viewing angle. Spatial selection is visualized as a color mask over the original BTF slice, and angular editing operations are done by dragging individual sliders. To provide interactive visual feedback, we employ our multi-threaded raytracer to generate new images after an editing operation is applied. Various optimization techniques, such as caching the intersection test results for eye rays and shadow rays, are used to accelerate

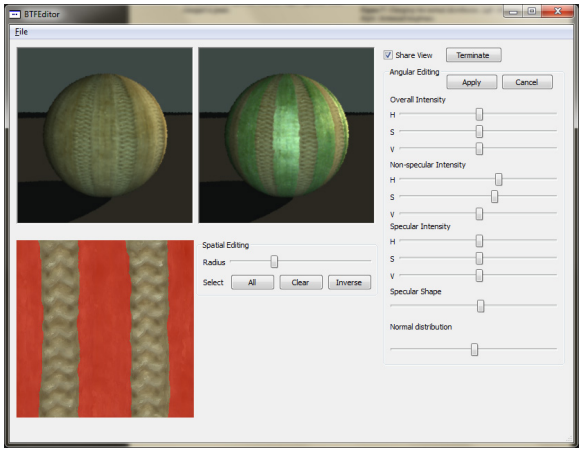
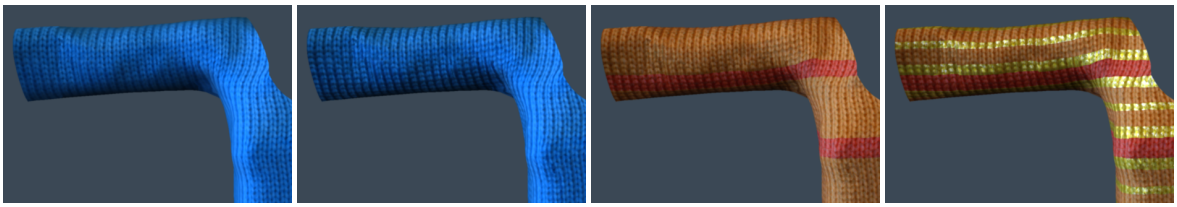


Figure 8: *The user interface of our interactive material editing system for SPMM. The top-left window shows the original appearance, and the edited look is displayed in the top-center window. The bottom-left window facilitates spatial selections. On the right side, various editing sliders are shown, including changing the hue, narrowing specular shapes, changing the normal distribution, etc.*



(a) Original (b) More embossed (c) Hue changes (d) Golden stripes

Figure 9: *A series of editing operations are applied to a wool blouse (a). First, we spread out the normal distribution of the SPMM to get a more embossed look (b). Then, red stripes (c) are added by shifting the hue. Finally, we add golden stripes (d) by both adjusting the color and narrowing the specular lobes.*

the rendering. On average, a rendering update of our system takes about half a second. In the future, it would be interesting to map our system onto GPUs for further performance improvement. Note that all editing examples shown in this paper are generated using this system. Please refer to Fig. 9 and our accompany video for an example of applying a series of editing operations.

Limitations Since our representation is mathematically derived, rather than strictly physically based, the results of editing operations might not be physically correct. This is similar to the editing on the raw BTF in [KBD07]. Nevertheless, we believe that in many applications of material editing, such as games and film production, the visual appearance is more important than physical accuracy to human observers. Our editing operations would be useful as long as the modified materials appear plausible.

6. BTF Rendering

In efficient Monte-Carlo rendering, we are interested in importance sampling of ω_i for a fixed ω_o. This is especially important for specular materials, where the reflectance changes quickly over the angular domain. So our idea here is to first divide all lobes in an SPMM into two groups, a specular group S and a non-specular one $\overline{S}$, using the empirical criterion described in Sec. 5.1, and then separately process them.

For the non-specular group $\overline{S}$, we have found in our experiments that it is sufficient to use one cosine lobe, whose local frame coincides with that of the BTF texel, as the probability distribution for importance sampling the sum of all lobes in $\overline{S}$, as they are slowly varying in the angular domain.

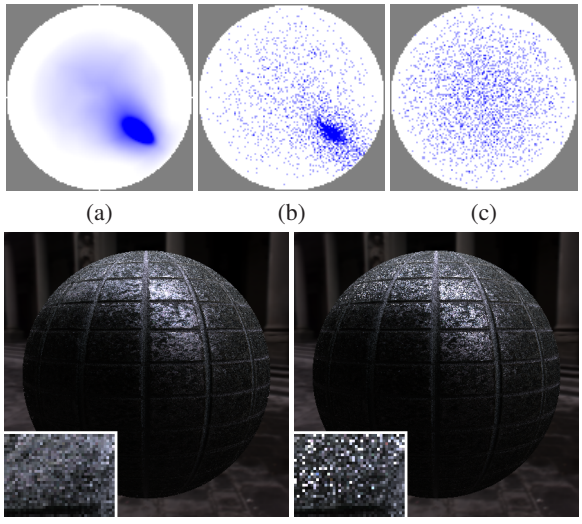
For the specular group S, recall that in Sec. 3.1, most of the BRDF models used in SPMM can be analytically importance sampled, except for Ward and Cook-Torrance. Therefore, if we fit the original BTF using only the models that can be analytically importance sampled, we can sample ω_i from the following mixture of analytical BRDFs:

$$\sum_{j\in S} \gamma_j f_j(\kappa_j, R(\omega_i), R(\omega_o)), \tag{8}$$

where γ_j is the precomputed ratio of the average intensity of lobe j over the average intensity of the above mixture,

$$\gamma_j = \frac{\iint \alpha_j f_j(\kappa_j, R(\omega_i), R(\omega_o)) d\omega_i d\omega_o}{\sum_{k\in S} \iint \alpha_k f_k(\kappa_k, R(\omega_i), R(\omega_o)) d\omega_i d\omega_o}. \tag{9}$$

To sample a direction from the mixture defined in Eq. (8), we first randomly choose a lobe j based on the probability distribution of $\{\gamma_j\}$. Then, we sample a ω_i using the analytical importance sampling method of f_j. Note that the probability of choosing ω_i is the weighted probability of sampling ω_i in each lobe in the mixture.



(a) (b) (c) (d) (e)

Figure 10: *SPMM-based importance sampling of an edited impala material. Images in the top row are generated by projecting the upper hemisphere of a local frame onto a planar circle. (a) BTF intensity distribution for a fixed ω_o. The more saturated blue, the higher the intensity. (b) Samples generated using our SPMM-based importance sampling. (c) Samples generated from a cosine lobe. In the bottom row are equal-time renderings using our method (d), and using cosine-lobe-based samples (e). The noise level in (d) is much lower than in (e).*

Finally, to importance sample the entire SPMM, we sim-

ply combine the sampling methods for S and $\overline{S}$, and use a weighted average as the combined probability distribution function. Since we are using a cosine lobe when sampling $\overline{S}$, it is guaranteed that every direction in the upper hemisphere has a nonzero probability to be sampled. Please refer to Fig. 10(a)-(c) for visualizations of different sampling methods. Our method generates samples whose density approximately matches the BTF intensity distribution. For the edited *impala* in Fig. 10, the additional storage for $\{\gamma_j\}$ and the weights to combine samples from S and $\overline{S}$ is only 0.73MB. Fig. 10(d) and (e) show a comparison of the noise level between the rendering results using our method and the conventional cosine-weighted method.

7. Conclusion and Future Work

We have proposed a compact, easily editable and efficiently renderable high-quality representation for general BTFs. We have also presented a stagewise-Lasso-based fitting algorithm to compute our representation from raw BTF data. To our knowledge, this is the first algorithm for fitting multiple, rotated analytical BRDFs of different types. We have shown the applications of our representation in BTF compression, editing and rendering.

In future work, we would like to implement SPMM on GPUs in order to achieve real-time rendering. We would also like to experiment with other analytical BRDF models, or even a different formulation of ρ_j, to further improve fitting quality or introduce more editing options.

Acknowledgements We would like to thank Huan Wang for inspiring discussions on Lasso algorithms, the University of Bonn and PSA Peugeot Citroen for BTF databases, and Soloumon Boulos and Jan Kautz for 3D models.

References

[AP07] ASHIKHMIN M., PREMOZE S.: Distribution-based BRDFs. *Unpublished Technical Report, University of Utah* (2007). 2

[AP08] AN X., PELLACINI F.: AppProp: all-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (2008), 1–9. 5

[CDS98] CHEN S. S., DONOHO D. L., SAUNDERS M. A.: Atomic decomposition by basis pursuit. *SIAM J. on Scientific Computing 20* (1998), 33–61. 4

[DRS07] DORSEY J., RUSHMEIER H., SILLION F.: *Digital Modeling of Material Appearance*. Morgan Kaufmann Publishers Inc., 2007. 1

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (1999), 1–34. 1, 2

[FCGH08] FILIP J., CHANTLER M. J., GREEN P. R., HAINDL M.: A psychophysically validated metric for bidirectional texture data reduction. *ACM Trans. Graph. 27*, 5 (2008), 1–11. 2

[FH09] FILIP J., HAINDL M.: Bidirectional texture function modeling: A state of the art survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence 31* (2009), 1921–1940. 2

[GMSK09] GUTHE M., MÜLLER G., SCHNEIDER M., KLEIN R.: BTF-CIELab: A perceptual difference measure for quality assessment and compression of BTFs. *Computer Graphics Forum 28*, 1 (2009), 101–113. 2

[HFM10] HAVRAN V., FILIP J., MYSZKOWSKI K.: Bidirectional texture function compression based on the multilevel vector quantization. *Computer Graphics Forum (Proc. of Eurographics 2010) 27*, 1 (Jan 2010), 175–190. 2, 5, 6

[KBD07] KAUTZ J., BOULOS S., DURAND F.: Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph. 26*, 3 (2007), 53. 2, 8

[LBAD*06] LAWRENCE J., BEN-ARTZI A., DECORO C., MATUSIK W., PFISTER H., RAMAMOORTHI R., RUSINKIEWICZ S.: Inverse shade trees for non-parametric material representation and editing. In *Proc. of SIGGRAPH '06* (2006), ACM, pp. 735–745. 2, 3

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proc. of SIGGRAPH '97* (1997), ACM, pp. 117–126. 4

[LH74] LAWSON C. L., HANSON R. J.: *Solving least squares problems*, 3 ed. Prentice-Hall, 1974. 4

[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 *Proc. of I3D '05* (2005), ACM, pp. 187–194. 2

[MG09] MENZEL N., GUTHE M.: g-BRDFs: An intuitive and editable BTF representation. *Computer Graphics Forum 28*, 8 (2009), 2189–2200. 1, 2, 3

[MMK03] MÜLLER G., MESETH J., KLEIN R.: Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003* (Nov. 2003), pp. 271–280. 4

[MMS*05] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis and rendering of bidirectional texture functions. *Computer Graphics Forum 24*, 1 (Mar. 2005), 83–109. 1, 2, 3

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental analysis of BRDF models. In *Proc. of the Eurographics Symposium on Rendering* (2005), Eurographics Association, pp. 117–226. 3

[NWW08] NOCEDAL J., WÄCHTER A., WALTZ R. A.: Adaptive barrier update strategies for nonlinear interior methods. *SIAM J. on Optimization 19*, 4 (2008), 1674–1693. 4

[RK09] RUITERS R., KLEIN R.: BTF compression via sparse tensor decomposition. *Computer Graphics Forum 28*, 4 (2009), 1181–1188. 2

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Proc. of Eurographics Workshop on Rendering '03* (2003), Eurographics Association, pp. 167–177. 5

[WBS*04] WANG Z., BOVIK A. C., SHEIKH H. R., MEMBER S., SIMONCELLI E. P., MEMBER S.: Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Processing 13* (2004), 600–612. 5

[XWT*09] XU K., WANG J., TONG X., HU S.-M., GUO B.: Edit propagation on bidirectional texture functions. *Computer Graphics Forum 28*, 7 (2009), 1871–1877. 2

[ZY07] ZHAO P., YU B.: Stagewise Lasso. *J. Mach. Learn. Res. 8* (2007), 2701–2726. 4