Interactive High Fidelity Visualization of Complex Materials on the GPU[☆]

Nuno Silva[a], Luís Paulo Santos[b,*]

[a] *Centro de Computação Gráfica, Campus de Azurém, Guimarães, Portugal*
[b] *Dep. de Informática, Universidade do Minho, Campus de Gualtar, 4710-054 Braga, Portugal*

Abstract

High fidelity interactive rendering is of major importance for footwear designers, since it allows experimenting with virtual prototypes of new products, rather than producing expensive physical mock-ups. This requires capturing the appearance of complex materials by resorting to image based approaches, such as the Bidirectional Texture Function (BTF), to allow subsequent interactive visualization, while still maintaining the capability to edit the materials' appearance. However, interactive global illumination rendering of compressed editable BTFs with ordinary computing resources remains to be demonstrated.

In this paper we demonstrate interactive global illumination by using a GPU ray tracing engine and the Sparse Parametric Mixture Model representation of BTFs, which is particularly well suited for BTF editing. We propose a rendering pipeline and data layout which allow for interactive frame rates and provide a scalability analysis with respect to the scene's complexity. We also include soft shadows from area light sources and approximate global illumination with ambient occlusion by resorting to progressive refinement, which quickly converges to an high quality image while maintaining interactive frame rates by limiting the number of rays shot per frame. Acceptable performance is also demonstrated under dynamic settings, including camera movements, changing lighting conditions and dynamic geometry.

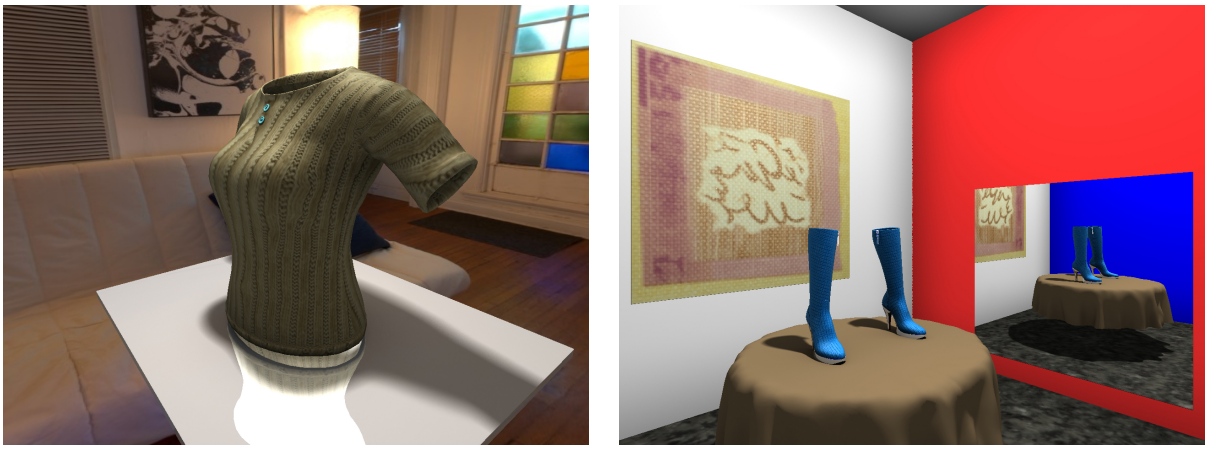
Keywords:
interactive visualization, ray tracing, complex materials, bidirectional texture function, sparse parametric mixture model

1. Introduction

High fidelity interactive rendering is of major importance for product designers, since it allows virtual prototyping new products, rather than producing expensive physical mock-ups. Besides accurate modeling of lighting conditions, light transport and the products'

[☆]Published as: Interactive High Fidelity Visualization of Complex Materials on the GPU; Nuno Silva and Luís Paulo Santos; Computers & Graphics, vol. 37(7), pp. 809–819; November 2013.
http://dx.doi.org/10.1016/j.cag2013.06.006

[*]Corresponding author
Email addresses: nunosilva800@gmail.com (Nuno Silva), psantos@di.uminho.pt (Luís Paulo Santos)



(a) The shirt scene (60 fps)

(b) The Cornell Box (60 fps)

Figure 1: Fully converged images with an area light source and progressive ambient occlusion.

geometry, high fidelity rendering requires simulating the appearance of the wide range of complex materials found on the products designers work with. Interactive rendering requires that the appearance of such materials can be simulated at high frame rates, thus placing a challenging constraint on how these materials are represented and rendered.

A material's appearance depends on how incident radiant flux is scattered when it hits a surface, and varies, among others, according to incoming light and observation directions [1]. Parametric Bidirectional Reflectance Distribution Functions (BRDF) are often used to model a material's appearance, but they cannot simulate many complex non-local lighting phenomena such as self-shadowing, self-occlusion, sub-surface scattering and inter-reflections. Image-based approaches, such as the Bidirectional Texture Function (BTF), can represent non-local phenomena and spatially varying optical properties, which makes rendering using BTFs extremely realistic. The BTF is a 6D function over position, (x, y) and the pair of incident and view directions, $(\theta_i, \phi_i, \theta_o, \phi_o)$ [2]. A material's appearance over a surface is captured by recording a large number of images under different incident lighting and viewing directions. The images are stored in large tables, and rendering involves look-ups within these tables. A single BTF with enough spatial and angular resolution can easily require several gigabytes of storage, hindering interactive rendering. To address this problem several solutions have been proposed, which transform the raw BTF data into a compact and efficiently renderable representation, while minimizing losses on perceived image fidelity. The efficiency of these compressed approaches, together with improvements in BTF acquisition systems and the increasing computational power of GPUs, contributed to the increasing popularity of BTF based rendering systems [1, 3, 4]. However, most of these highly efficient compressed representations do not allow intuitive editing of the BTF data [5, 6, 7], as required by product designers. Interactive global illumination rendering with ordinary computing resources using an intuitively editable representation of the BTF remains an elusive goal.

Wu et al. [8] presented a novel general representation for BTFs, referred to as the Sparse Parametric Mixture Model (SPMM), which is based on a sparse linear combination of basis functions. These basis functions are taken from a dictionary of well known analytical BRDFs, which makes this representation particularly well suited for BTF editing, since the basis functions parameters are well understood by computer graphics and design practitioners. Wu et al. demonstrated that the SPMM achieves a good compromise between rendering performance and image quality by resorting to global illumination light transport models and a parallel CPU ray tracer. Achieved frame rates are, however, far from interactive.

In this paper we demonstrate that high fidelity interactive rendering of complex materials, within the context of cloth and footwear virtual prototyping, is in fact possible. We adapt the SPMM rendering pipeline to the programming model supported by NVIDIA's GPU ray tracing engine - OptiX [9] - and show interactive frame rates with area light sources, specular reflections and diffuse interreflections approximated by ambient occlusion – see figure 1. In order to maintain interactive frame rates both area light sources' visibility and ambient occlusion are progressively accumulated over time. We assess the performance of our approach with respect to some parameters of the SPMM representation, such as the number of basis functions and the number of coefficients used to represent the residual error, and we subjectively compare our visual results with those obtained by Wu et al. [8] with an offline global illumination renderer. We further conduct a performance scalability study on the number of light sources, geometric primitives and SPMMs, and also demonstrate interactive frame rates under dynamic settings, such as moving camera, dynamic geometry and changing lights positions and intensities.

The SPMM is not particularly well suited for image synthesis on the GPU, since it requires scattered non-coalesced memory accesses, which strongly impact on rendering performance. Alternative representations based on factorizing the BTF tensor (or subsets thereof) [5, 6, 7] have the potential to achieve better frame rates, since they only require evaluating dot products. The SPMM representation has, however, the advantage of allowing direct intuitive editing of the BTF. The visualization technique presented in this paper was developed within the context of a research project with the portuguese footwear industry, whose goal is to enable footwear designers to rapidly prototype and visualize new products. The ability to intuitively and readily edit the materials is fundamental to these designers and motivated the choice of this particular representation, together with its high compression ratios and synthesis quality. An additional contribution of this paper to current knowledge is the demonstration that interactive global illumination rendering of editable SPMM-based representations of complex materials is in fact possible using current GPUs.

This paper is organized as follows: the next section discusses related work, section 3 briefly introduces the SPMM representation, whereas section 4 presents the visualizer's rendering pipeline. Results are then analyzed and the paper closes with some concluding remarks and suggestions for future work.

2. Related Work

Most real world materials' appearance can be described at three levels [3, 10]. The macroscale is the large scale geometry of the object, traditionally modeled with explicit representations, such as polygon meshes. The microscale level relates to local interactions of light with a single point of the material's surface, and can be represented using BRDFs. The mesoscale level is in-between these two and comprises various subtle, mostly non-local, lighting effects, such as self-shadowing, self-occlusion, sub-surface scattering and interreflections, which cannot be faithfully represented with the BRDF; instead, image based approaches are used, the most common method being texture mapping.

A material's appearance is usually modeled resorting to three alternative approaches:

- analytical models, such as the BRDF or the Spatially Varying BRDF (SVBRDF) [11]. The latter can be seen as a combination of texture mapping and BRDFs, accounting for materials that have different BRDFs throughout their surfaces. While addressing some of the issues raised at the mesoscale level, they cannot capture non-local lighting phenomena, such as self-shadowing and self-occlusion. The 8-dimensional Bidirectional Subsurface Scattering Reflectance Distribution Function (BSSRDF) can model all these lighting phenomena, but is much too complex to be usable in interactive rendering pipelines, and current capturing systems only measure subsets of this function [1, 3].

- procedural models, requiring the development and parameterization of an algorithm and mathematical functions until the desired appearance is achieved [12]. This is a time consuming task that demands high level of expertise and might still fail to accurately simulate real world materials.

- image based methods, where data is collected from one or more images of the material, the most common method being texture mapping. The BTF is also an image driven approach, requiring a huge number of photographs from a large number of lighting and viewing directions [2]. Image based methods allow new materials to be captured by non-experts by photographing new samples, dispensing with the programming expertise required by procedural approaches.

Since this paper focuses on image based approaches, particularly the BTF, this discussion of related work concentrates on alternative representations of the BTF data, emphasizing those leading to interactive rendering.

The BTF represents both mesoscale and microscale lighting phenomena. It is a hexadimensional function, which, for each wavelength, models the material's appearance based on a point on the surface, $p = (x, y)$, and a pair of incident and viewing directions, $\omega_i = (\theta_i, \phi_i), \omega_o = (\theta_o, \phi_o)$. The raw, non-compressed, BTF data consists on the images representing a dense set of pairs of directions in order to properly sample the hemisphere of light and camera positions. A good quality BTF, such as the ones in the Bonn database [13], encodes $81 * 81$ images for incident and viewing directions, each consisting of 256^2 texels

with three spectral values (RGB). This corresponds to roughly 1.2 GB of raw data for a single BTF, not including High Dynamic Range (HDR) measurements. Achieving interactive visualization rates requires compressing the raw data, exploiting redundancy in an efficient way and allowing fast decompression for real-time rendering.

BTF compression methods are usually based on one of three alternative approaches: linear basis decomposition, probabilistic modeling and analytical reflectance models. Linear basis decomposition is often based on matrix factorization, using techniques such as Principal Component Analysis (PCA), and then maintaining only the largest components. Since the BTF is multi-dimensional, it has to be unfold into a matrix representation prior to factorization. Sattler et al. [13] group data for the same view direction into a matrix and then factorize each of these matrices independently, achieving low compression ratios and up to 10 fps on graphics hardware, including volumetric shadows for point/directional light sources and image based illumination extended with pre-computed visibility maps. Muller et al. [14] cluster the Apparent BRDFs (ABRDF), i.e., the data for all lighting and viewing directions for each spatial point, and then apply PCA over these clusters; they report up to 14 frames per second (fps) on graphics hardware, but the associated illumination model includes only direct lighting from point light sources and ignores light visibility (i.e., shadows). Schneider et al. [15] improve on this algorithm by reparameterizing the associated eigen-BRDFs, and report up to 64 fps with the same illumination model. Suykens et al. [10] report from 50 to 150 fps on graphics hardware using a chained matrix factorization of the raw BTF data and simulating only direct illumination from point light sources without any shadows evaluation. Full matrix factorizations, such as in [16, 17], allow for significant compression ratios. The major limitation with matrix-based approaches is that the high dimensional structure of the BTF data set is not exploited on the compression stage; matrices are two-dimensional and only correlations among columns are being exploited. Havran et al. [6] decompose individual ABRDFs into multidimensional conditional probability density functions, which are then further compressed using multi-level vector quantization. The proposed approach achieves high compression ratios, supports mip-mapping and importance sampling for Monte Carlo based renderers, and the authors report rendering rates of up to 170 fps on the GPU for point based lighting. Schwartz et al. [18] demonstrate that BTFs can be used for interactive progressive photorealistic visualization of cultural heritage artifacts over the web. They use SVD factorization together with a wavelet based compression and stream individual components sorted by order of perceptual relevance to a WebGL visualizer, being able to present high-quality previews of the BTF within a few seconds.

In order to leverage correlations across multiple dimensions, it is possible to directly factorize the BTF tensor representation. Techniques based on tensor product expansion [19] or on N-mode SVD [20, 21, 22] have been proposed. Wang et al. [21] start from the BTF 6th-order tensor and derive a lower rank tensor compressed representation, rather than reducing the data dimensionality. They demonstrate higher compression ratios and superior rendering fidelity compared to alternative approaches. Additionally, they propose an out-of-core technique to handle data sets larger than available memory capacity. However, these approaches still incur high overheads, since random access to a tensor's element requires evaluating long sums. Ruiters et al. [5] propose reducing the number of terms evaluated

during decompression by combining tensor based techniques with a sparse representation similar to local PCA. They apply K-SVD to decompose the tensor into a dictionary and two sparse tensors, achieving simultaneously very high compression ratios and decompression rates larger than those possible applying PCA to the full matrix. Tsai et al. [7] integrate into a single framework the concepts of clustering, sparse representation and tensors. This framework, K-CTA, allows for inter-cluster coherence exploitation by classifying each subtensor into several clusters. The proposed representation is made particularly well suited for synthesis on the GPU by guaranteeing that the number of coefficients on which each subtensor depends is a constant, thus avoiding dynamic branching. Tsai et al. [23] propose a decomposition onto multivariate spherical radial basis functions (SRBFs) coupled with learning of optimized parameterization functions, which provides a data-dependent method for transforming the parameters of a reflectance function into another parametric space, resulting in significantly improved approximation efficiency. They show that in general multivariate SRBFs lead to more efficient rendering performance, while tensor approximation provides a more accurate representation.

Probabilistic modeling approaches are most often based on a combination of displacement filters and Markov random fields, achieving large compression ratios and allowing for arbitrary BTF synthesis. They are however not suited for intuitive material appearance editing [24].

The last group of compression methods represent the BTF data by resorting to analytical reflectance models. McAllister et al. [11] used LaFortune lobes, whereas Ma et al. [25] model average BTF reflectance using the Phong model. In another paper, Ma et al. [26] use a Laplacian pyramid to decompose the raw BTF data into multiple subbands, thus enabling the use of Levels of Detail; each level is further compressed using PCA. The resulting multiresolution representation allows for faster rendering with increasing depth, without compromising on perceived image quality. The authors report up to 30 fps, upper limited by the graphics hardware fill rate, using a local illumination model. The SPMM [8] linearly combines any set of such BRDF analytical models, with the differences between the model and the original data being stored in a per-cluster residual error function (see section 3 for further details). This representation of the BTF data is particularly well-suited for editing of the material's appearance, since the parameters of the basis BRDFs are well understood and might be manipulated directly. It achieves a balanced compromise between compression rate and image quality, and allows for interactive visualization rates, as demonstrated in this paper. The proposed approach is based on a ray tracing engine and can integrate all lighting phenomena supported by geometric optics. Interactive results on graphics hardware are presented for direct lighting with shadows from point and area light sources, specular reflections and diffuse interreflections approximated with ambient occlusion.

Refer to [3, 4] for further details on BTF modeling.

3. The Sparse Parametric Mixture Model

The Sparse Parametric Mixture Model (SPMM) is a compact and intuitively editable representation of generic BTFs, which can be efficiently rendered [8]. Each BTF texel, t, is

approximated as a sparse linear combination of basis functions plus a residual function as illustrated by equation 1, where $t = (u, v)$ identifies the texel in texture coordinates, and (ω_i, ω_o) is the pair of incident and viewing directions; the representation parameters for texel t (identified as a superscript) are the number of basis functions (m^t) and the weight α_j^t of the j^{th} basis function, ρ_j^t; $\epsilon(t, \omega_i, \omega_o)$ is the residual function.

$$BTF(t, \omega_i, \omega_o) = \sum_{j=1}^{m^t} \alpha_j^t \rho_j^t(\omega_i, \omega_o) + \epsilon(t, \omega_i, \omega_o) \tag{1}$$

Each basis function, ρ_j, is a cosine weighted rotated analytical BRDF:

$$\rho_j(\omega_i, \omega_o) = f_j(\kappa_j, R(\omega_i), R(\omega_o))(n_j \cdot \omega_i) \tag{2}$$

$f_j(\kappa_j, \cdot, \cdot)$ is one analytical BRDF with parameters κ_j and R is a rotation to the local frame with normal n_j. The authors use seven analytical BRDFs as candidates for $f_j(\kappa_j, \cdot, \cdot)$, including Lambertian, Oren-Nayar, Blinn-Phong, Ward, Cook-Torrance, Lafortune and Ashikhmin-Shirley, but others can be used. The SPMM can be intuitively edited by direct manipulation of the weights α_j, the BRDFs parameters κ_j and the local frames normals n_j.

In order to fit a BTF raw data to the SPMM, texels are clustered according to their similarity and a dictionary of basis functions ρ_j for each cluster is evaluated from a set of representatives (selected by sub-clustering) using the IPOPT [27] optimization algorithm. The weights α_j are then computed for each texel using the stagewise LASSO algorithm [28]; this algorithm uses the ℓ_1 norm minimization criterion, thus promoting sparsity, i.e., a significant number of weights is zero.

Finally, a residual function, $\epsilon(t, \omega_i, \omega_o)$, is calculated. The residual is evaluated as the average error between the SPMM and the original BTF data for all texels t in each cluster c and is represented by resorting to per cluster PCA compression. It allows representation of non-local phenomena, such as shadowing, masking and interreflections, that would not be captured by the weighted sum of analytical BRDFs. The residual is evaluated according to equation 3:

$$\epsilon(t, \omega_i, \omega_o) = \bar{\epsilon}(c, \omega_i, \omega_o) + \sum_{i}^{N_{PCA}} (\text{coef}_i(t) * b_i(c, \omega_i, \omega_o)) \tag{3}$$

The empirical mean, $\bar{\epsilon}(c, \omega_i, \omega_o)$, and the set of PCA basis functions, $b_i(c, \omega_i, \omega_o)$, are evaluated for each cluster c, whereas the PCA coefficients, $\text{coef}_i(t)$, are evaluated on a per-texel basis. The number of PCA coefficients, N_{PCA}, effectively used impacts both in rendering time and visualization quality, as shown in section 5.2.

4. The Rendering Pipeline

The proposed interactive global illumination renderer runs integrally on the GPU and is based on ray tracing. We used NVIDIA's OptiX [9] as the GPU ray tracing engine,

which transparently handles all fundamental ray casting operations, such as space traversal and ray-geometry intersection, texture coordinates evaluation, etc. OptiX's programming model, together with the GPU architecture, impose some constraints on the program control flow and data structures that are discussed over the next two subsections; these constraints are particularly relevant if performance is critical, such as with the proposed interactive visualizer.

4.1. Illumination Model

Our illumination model includes direct lighting from both point and area light sources, specular reflections and diffuse interreflections approximated with ambient occlusion. Each point light source visibility is trivially evaluated with a single shadow ray; similarly, specular reflections are evaluated with a single ray traced from the shading point along the perfect specular reflection direction. Area light sources require sampling visibility over the respective solid angle, whereas ambient occlusion requires evaluating occlusion over the hemisphere Ω_p centered at the shading point p and with a given radius r [29]. Since these two operations might require a large number of rays in order to deliver an acceptable image quality, interactive frame rates would be compromised. A progressive approach is thus used, where both area light sources and ambient occlusion are progressively refined.

Progressive area light source sampling - direct lighting due to an area light source L subtending a solid angle $\Omega_p(L)$ on a shading point p can be uniformly sampled according to equation 4:

$$\langle L(p \rightarrow \omega_o) \rangle = \frac{\Omega_p(L)}{N} \sum_{k=1}^{N} L_e(y_k \rightarrow \omega_{i,k}) f_r(p, \omega_o \leftrightarrow \omega_{i,k}) \cos(n_p, \omega_{i,k}) V(p, y_k) \tag{4}$$

where y_k is the intersection point of the k^{th} shadow ray with the area light source, $L_e(y_k \rightarrow \omega_{i,k})$ is the radiance emitted from point y_k along the shadow ray direction $\omega_{i,k}$, $f_r(p, \omega_o \leftrightarrow \omega_{i,k})$ is the BRDF at p for the given pair of directions, n_p is the surface normal at p and $V(p, y_k)$ represents the visibility of y_k from p. Assuming that the light source emitted radiance, L_e, is constant independently of y_k and ω_i, then equation 4 can be rewritten as:

$$\langle L(p \rightarrow \omega_o) \rangle = \frac{\Omega_p(L) * L_e}{N} \sum_{k=1}^{N} f_r(p, \omega_o \leftrightarrow \omega_{i,k}) \cos(n_p, \omega_{i,k}) V(p, y_k) \tag{5}$$

Our approach to progressively sample the visibility of the area light source is to shoot a single shadow ray per frame and accumulate the sum present on the right hand term of equation 5. Accumulation is performed on an image plane buffer on a per-pixel basis and multiplied by the term $\frac{\Omega_p(L) * L_e}{N}$ on rendering time, where N represents the number of frames displayed since accumulation initiated and is equal to the number of shadow rays shot per pixel. Since the light source accumulated contribution is maintained on image space, accumulation has to be reset every time the camera moves. It is also reset every time the world's geometry (including the light source) changes, due to alterations on visibility and

relative directions; note, however, that changes on the light source emitted radiance, L_e, do not require the accumulation process to be reinitiated. Progressive sampling the light source allows maintaining interactive frame rates, while including physically based soft shadows on the illumination model.

Progressive ambient occlusion - ambient occlusion $A(p)$ denotes the amount of occlusion a point p receives due to nearby occluders. $A(p)$ is given by

$$A(p) = \frac{1}{\pi} \int_{\Omega_p} V(p, \omega_i, r) \max(\cos(n_p, \omega_i), 0) d\,\omega_i \tag{6}$$

where Ω_p denotes the hemisphere centered on p and $V(p, \omega_i, r)$ indicates whether there is an occlusion from p along direction ω_i up to distance r. This integral can be approximated by resorting to classical Monte Carlo integration using a cosine importance distribution for samples as given by equation 7, where N stands for the number of samples:

$$\langle A(p) \rangle = \frac{1}{N * \pi} \sum_{k=1}^{N} V(p, \omega_{i,k}, r) \tag{7}$$

Our approach to progressive ambient occlusion is identical to the previously described progressive area light source sampling. A single ambient occlusion ray is shot per shading point on each frame and the sum in equation 7 is accumulated on an image plane buffer on a per-pixel basis. This sum is then multiplied by the ambient lighting term and $\frac{1}{N*\pi}$ for writing on the frame buffer. Due to image plane buffering the accumulation process is reinitiated whenever the camera moves and also when the scene's geometry changes due to visibility variations. Changes on lighting conditions will only require reinitiating progressive ambient occlusion if they impact on the approximated ambient lighting term. By shooting a single ray per shading point per frame, global illumination is approximated without compromising on the visualizer interactivity.

4.2. BTF Synthesis

Applying the above described illumination model requires evaluating the BRDF for each shading point and pair of view and incident directions. Whenever a shading point lies on a material mapped with a BTF, the BRDF has to be evaluated from its SPMM representation. This representation is not geared towards optimal GPU performance because it consists on a linear combination of different analytical BRDFs – which results on code divergence due to conditional branching – and relies on multiple table lookups – which leads to multiple non-consecutive low bandwidth memory accesses. Data structures must thus be carefully designed such that memory access overheads do not compromise performance. Current GPU architectures require SPMM data to be stored in the device texture memory, as 1D, 2D or 3D arrays [30, 31], allowing fast random accesses and high bandwidth.

Synthesizing the BRDF value from the SPMM representation, given the texture coordinates and the pair of incident and viewing directions, requires evaluating the linear

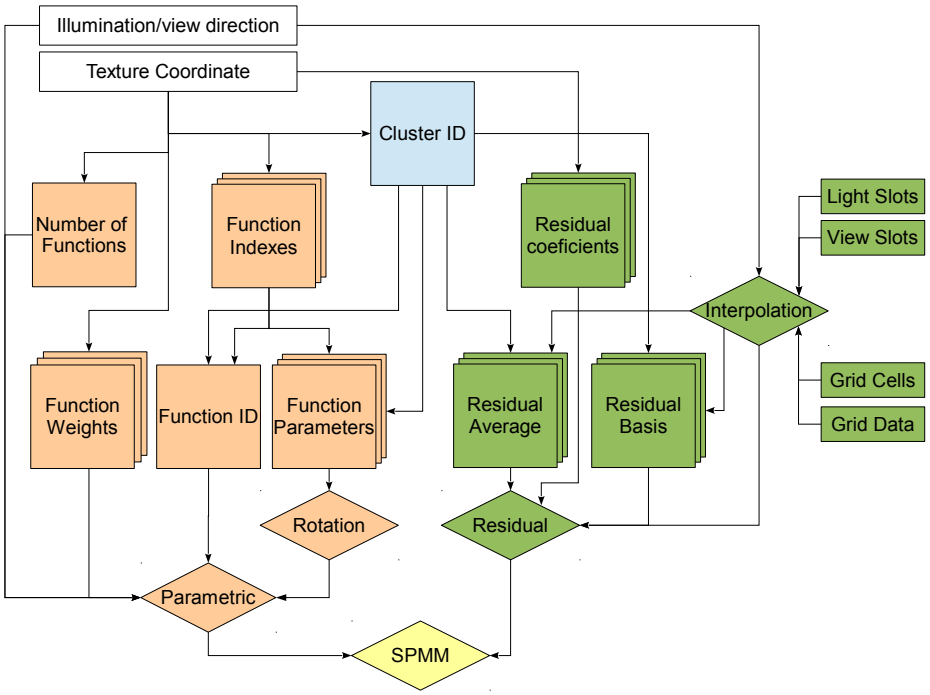


Figure 2: BTF synthesis data flow. Rectangles, squares and stacked squares represent, respectively, 1D, 2D and 3D textures; diamonds represent mathematical operations on data. White boxes represent input, pink boxes represent the evaluation of the linear combination of basis functions, whereas green boxes are related to the evaluation of the residual function.

combination of basis functions and the residual function, as given by equation 1. Figure 2 illustrates the data flow for BTF synthesis.

The pink left-hand part of the diagram represents the evaluation of the linear combination of analytical basis functions. The texel's cluster ID, the number of basis functions (m on equation 1) and the function ID (used together with the function indexes to retrieve ρ_j) are stored as 2D textures of integers. The function indexes are stored as 3D textures of integers, whereas the basis functions weights and parameters (α_j and κ_j respectively on equation 1) are stored on 3D textures of 4 bytes floating point numbers (the minimum resolution allowed by OptiX). Table 1 summarizes all these textures and presents their sizes for the particular case of the wool material.

		Wool	
Name	Format	#elements	bytes
Cluster ID	UNSIGNED BYTE	64K	64K
Function Parameters	FLOAT	71.25K	285K
Function ID	UNSIGNED BYTE	9K	9K
Number of Functions	UNSIGNED BYTE	64K	64K
Function Indexes	UNSIGNED SHORT	720K	1440K
Function Weights	FLOAT	2112K	8448K
Residual Basis	FLOAT	2496K	9984K
Residual Average	FLOAT	615K	2460K
Residual Coefs	FLOAT	256K	1024K
Light/View Slots	FLOAT	243	972
Grid Cells	UNSIGNED INT	324	1296
Grid Data	FLOAT	4824	19296

Table 1: Data structures used to represent the SPMM and figures for the particular case of the Wool material.

Evaluation of the residual function, $\epsilon(p, \omega_i, \omega_o)$, requires an additional interpolation step. This function is represented using clustered-PCA (see equation 3), whose coefficients and mean values were evaluated for all pairs of lighting/view directions along which the BTF was actually sampled; we will refer to these pairs as light/view slots. In rendering time the residual function coefficients for the actual pair of view/incident directions must be interpolated from the closest light/view slots. Note that the analytical basis functions described above do not require interpolation, since they are analytically defined over the entire upper hemisphere.

The direction slots can be interpreted as 3D points over the unit hemisphere and projected onto a set of points on the XY plane by ignoring the Z component. A Delaunay triangulation is applied to these points and the resulting triangles are stored in a texture. Interpolation for a given direction requires determining which of these triangles is intersected by a ray with this direction and origin at the shading point. In order to reduce the number

of such intersections, and given that the Delaunay triangulation is static for a given BTF and triangles are well distributed over the unit circle, a compact regular grid structure with minimal memory requirements [32] is built once and uploaded to the GPU using two 1D textures (rectangles "Grid Cells" and "Grid Data" in figure 2). A ray-triangle intersection test using barycentric coordinates [33] is applied to non-discarded triangles, resulting in the appropriate interpolation weights in case of a hit. This must be done separately for the view and lighting directions, for a total of nine interpolation weights. To further reduce the cost of intersection tests, equation 8 is used, where λ is the vector of barycentric coordinates, r is the ray vector, and T is a matrix formed by the Cartesian coordinates of the triangle vertices. This allows precomputation of the inverse matrix T^{-1} for each triangle, reducing the intersection test on the GPU to a vector subtraction and a matrix-vector multiplication. To maximize data locality, the inverse matrix, the Cartesian coordinates of each vertex and its corresponding ID are packed together, for a total of 16 floats per triangle.

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = T^{-1}(r - v_3) \qquad \lambda_3 = 1 - \lambda_1 - \lambda_2 \tag{8}$$

5. Results

This section presents results achieved with the previously described approach. The performance of the proposed interactive visualizer with respect to some parameters of the SPMM representation, such as the number of basis functions and the number of coefficients used to represent the residual error, is assessed and visual results are subjectively compared with those obtained by Wu et al. [8] with an offline global illumination renderer. Scalability analysis is performed for varying numbers of light sources, geometric primitives and visible SPMMs, followed by a brief discussion of the performance achieved under dynamic settings (dynamic geometry, camera movements and changing lighting conditions).

5.1. Experimental Setup

Experiments were conducted on a workstation equipped with an Intel 2.4GHz quad-core processor, 4GB RAM and a Nvidia GeForce GTX 580 GPU, driver version 301.32. Version 2.5.1 of NVidia's OptiX GPU ray tracing engine was used. Image resolution was set to 512 x 512 pixels (except when otherwise noted), with a single primary ray per pixel. Three different scenes were used, whose face count is presented on table 2. For scalability analysis with respect to the number of geometric primitives the shoe present in the "Shoe" scene was replicated without using instancing, thus effectively increasing storage space. All tests were performed using the SPMM representation of the BTFs in the Bonn Database [13], which have a spatial resolution of 256 x 256 texels and an angular resolution of 81 x 81 directions. The SPMM data sets are the same as used by Wu et al. [8], which the authors kindly made available. These were generated using 32 clusters for the initial k-means and then k=8 for sub-clustering; each BTF texel is approximated using between 3 and 9 basis functions on average, selected among the Lambertian, Oren-Nayar, Blinn-Phong, Ward, Cook-Torrance,

Model	#faces
Cornell Box	104096
Shirt	19050
Shoe	254956
Shoe (x2)	502094
Shoe (x3)	749232
Shoe (x4)	996370

Table 2: Face count for the different scenes

Lafortune and Ashikhmin-Shirley analytical BRDF models, plus a residual function per cluster, which is approximated using 4 PCA coefficients - table 3 further details the average number of basis functions per texel used for each of the BTFs presented in this paper.

When performing quantitative comparisons among images the Root Mean Square Error (RMSE) and the Mean Structural SIMilarity (MSSIM) [34] metrics are used. The latter takes into account luminance, contrast and structure, leveraging the fact that most natural images are highly structured and that the human visual system is highly sensitive to this structure. The MSSIM index ranges from 0 to 1, higher values indicating increased similarity.

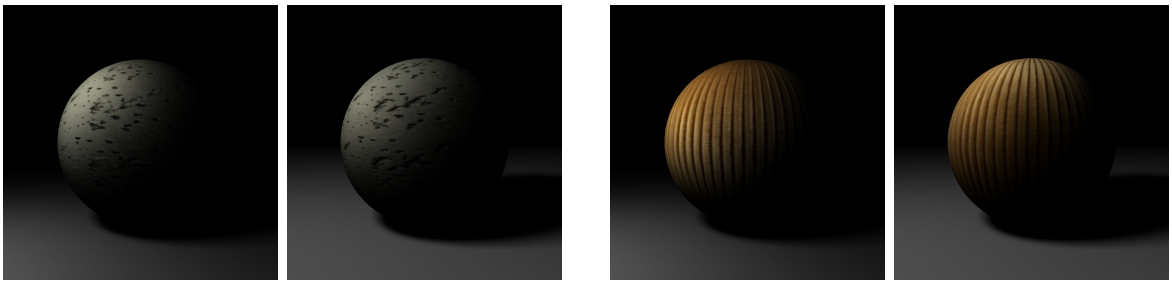
5.2. Performance and Visual Quality Analysis

Figure 3 presents images rendered using Wu et al. [8] offline renderer and the proposed interactive approach, together with the respective rendering times. Indirect reflections are not included on both renderers, i.e., ambient occlusion is switched off on the proposed visualizer, in order to minimize perceived differences. Nevertheless, a quantitative analysis of the perceptual difference between both images, resorting to metrics such as RMSE or MSSIM, is deemed very difficult to perform due to fundamental differences between the renderers, such as stochastic (ours) versus deterministic (Wu's) sampling of the area light sources, among others. Subjective analysis of the rendered images is however enough to demonstrate the quality of the proposed approach, in spite of the huge differences in rendering times, with the proposed interactive approach being approximately 5000 times faster.

In order to understand how do the average number of basis functions per texel and the evaluation of the residual function impact on the visualizer performance, we performed an experiment to get approximate measurements of the time spent on each fundamental operation: ray tracing (RT), evaluation of basis functions (BF) and evaluation of the residual (RES):

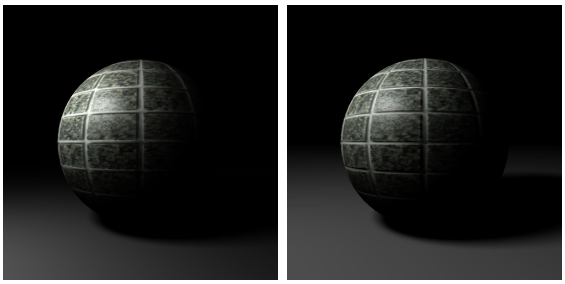
ray tracing (RT) – the time spent ray tracing is measured by rendering the test scene using only Lambertian materials; no SPMM evaluations are performed, but the total number of rays is exactly the same as if complex materials, represented using SPMMs, were present.

basis functions (BF) – the time spent evaluating the basis functions for each SPMM is approximated by measuring the time spent rendering the scene with that SPMM without evaluating the residual function and then subtracting the time spent ray tracing,

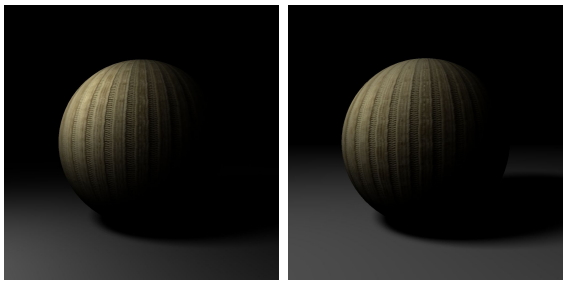


(a) Ceiling (Wu 71.3 secs, ours 12.5 msecs)

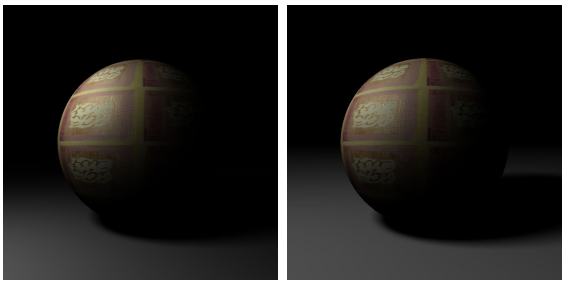
(b) Corduroy (Wu 74.7 secs, ours 15.9 msecs)



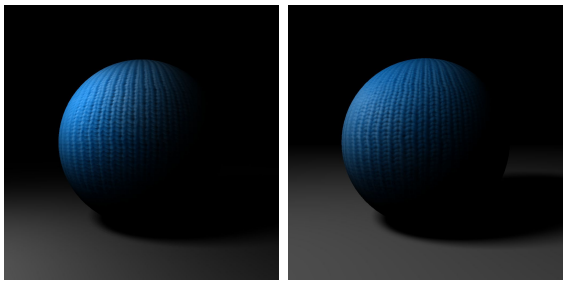
(c) Impala (Wu 74.6 secs, ours 15.6 msecs)



(d) Pulli (Wu 76.4 secs, ours 13.5 msecs)



(e) Wallpaper (Wu 71.7 secs, ours 14.2 msecs)



(f) Wool (Wu 81.7 secs, ours 14.5 msecs)

Figure 3: Comparing images rendered using Wu et al. [8] offline renderer (on the left within each pair) and the proposed interactive approach.

as described on the previous item. Even though the sum of the blue (RT) and red (BF) stacked bars in figure 4 effectively corresponds to the time spent rendering each image (without the residual), the height of the red bar and the values in column BF of table 3 only approximate the time spent evaluating the basis functions; this is because the GPU is now processing a different workload, which most certainly results on reordering computations – OptiX provides no functionality to separately measure the time spent on different regions of the code. Nevertheless, the BF partial timings do correspond to the difference between evaluating the basis functions and performing only ray tracing, thus they correlate to the added effort associated with this evaluation.

residual function (RES) – approximation of the time spent evaluating the residual function is given by the difference between the time spent rendering the scene with the full SPMM (including the residual) and the sum of the RT and BF partial times. It suffers from the same measurement inaccuracy as the BF timings, but remember that the top of the blue (RT), red (BF) and green (RES) stacked bars effectively corresponds to the timing required to fully render each image.

These partial timings are depicted as a function of the number of basis functions in both table 3 and figure 4. The frame rates with and without residual evaluation are also included, which in figure 4 must be read on the secondary vertical axis on the right. Each bar, associated with a particular material represented using a SPMM, is labeled with the respective average number of basis functions per texel. Note that this set of experiments was performed with an image resolution of 1280x962, rather than 512x512 as the remaining reported results.

The frame rate decreases with the number of basis functions, as expected, although not in a strictly monotonic manner. The lack of monotonicity is most probably due to different SPMMs using different mixtures of analytical BRDFs, some requiring more floating point operations to evaluate than others – e.g., compare Lambert with Oren-Nayar or with Ward's anisotropic BRDF. The fraction of rendering time spent evaluating basis functions varies from 26%, with 3.12 basis functions per texel on average, to 37% with an average of 8.71 basis functions per texel. Note, however, that evaluating the full SPMM, including the residual, always dominates rendering times, taking a fraction of 53% of the rendering time for the lightest SPMM to 61% for the heaviest one. The residual evaluation partial timings are harder to explain. These should remain constant, independently of the number of average basis functions, since residual evaluation always requires interpolating (as described in section 4.2) and texture lookups to retrieve the empirical mean and the clustered PCA coefficients and associated vectors (see equation 3). Since the number of PCA coefficients, N_{PCA}, is the same for all SPMMs and all texels (and equal to 4 on these particular data sets), and given that partial times fluctuations do not exhibit any particular trend with respect to the number of basis functions, these fluctuations are supposed to be due to inaccuracies on the measurement process, as described above.

In order to evaluate the impact of the residual evaluation on both the visualizer's performance and the images' quality, a set of experiments were performed, varying the residual evaluation parameters. Taking as a reference the results obtained by evaluating the residual empirical mean and all PCA coefficients (4 for these data sets), these were compared

BTF	#BF	Time (ms) RT	BF	RES	FPS RT+BF	RT+BF+RES
ceiling	3.12	18.9 (47%)	10.4 (26%)	11.1 (27%)	34.0	24.7
floortile	4.41	18.9 (44%)	13.3 (31%)	11.0 (25%)	31.0	23.1
wool	5.23	18.9 (42%)	14.2 (32%)	11.7 (26%)	30.1	22.3
proposte	5.64	18.9 (42%)	13.8 (31%)	12.2 (27%)	30.5	22.2
pulli	5.93	18.9 (45%)	13.4 (32%)	9.4 (23%)	30.9	23.9
wallpaper	6.21	18.9 (42%)	13.5 (30%)	13.1 (28%)	30.8	22.0
corduroy	7.84	18.9 (37%)	17.6 (35%)	14.1 (28%)	27.3	19.7
impala	8.71	18.9 (39%)	18.0 (37%)	11.3 (24%)	27.1	20.7

Table 3: Partial times for rendering the test scene with different average numbers of basis functions. Note that image resolution is 1280x962, thus different from the remaining experiments reported in this paper.

with those obtained by progressively reducing the number of used PCA coefficients down to zero. An additional data point is obtained on this series by also not evaluating the residual empirical mean, which corresponds to no residual evaluation at all and dispenses with the associated interpolation step. Table 4 presents the obtained frame rates (averaged over a 60 seconds run) and corresponding MSSIM and RMSE with respect to full residual evaluation, for the pulli and wool SPMMs. Figure 5 presents images obtained with the pulli material for the different settings. There is clearly a step change in image quality by introducing the residual function, even if only the empirical mean is used - this is noticeable both qualitatively, by comparing figures 5(a) and 5(b), and quantitatively, by comparing the respective MSSIM and RMSE metrics for both materials. Image quality keeps increasing with the number of PCA coefficients, but at a much less perceivable rate, as shown by the MSSIM metric.

Performance decreases as the accuracy of the residual evaluation increases. Note, however, that the performance loss associated with just evaluating the empirical mean ($N_{PCA} = 0$), which requires performing the interpolation step, compared to no evaluation of the residual is not significantly worse than adding an additional PCA coefficient. This shows that the interpolation step described in section 4.2 is quite efficient with an overall cost similar to the texture lookups required to retrieve one PCA coefficient and respective vector. Based on these results, and in order to allow smooth walkthroughs, the residual evaluation is switched off on the interactive visualizer when the camera moves and the residual is evaluated up to a user defined number of PCA coefficients when the camera is stationary. On all remaining experiments the number of PCA coefficients is set to the maximum available on these particular SPMM representations, i.e., 4.

5.3. Scalability Analysis

Figure 6 presents rendering times, in milliseconds, for a single frame while varying different scene configuration parameters. These experiments used the "Shirt", "Cornell Box" (depicted in figure 1) and "Shoe" (depicted in figure 7) scenes, whose face count details are given in table 2. All timings were obtained with ambient occlusion and one area light source,

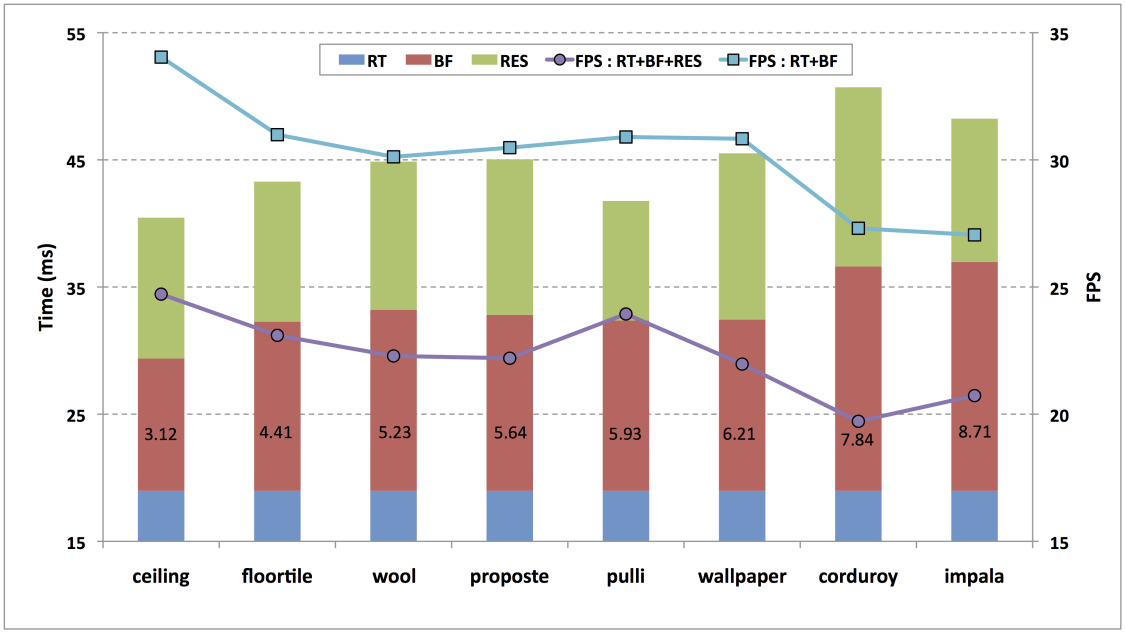
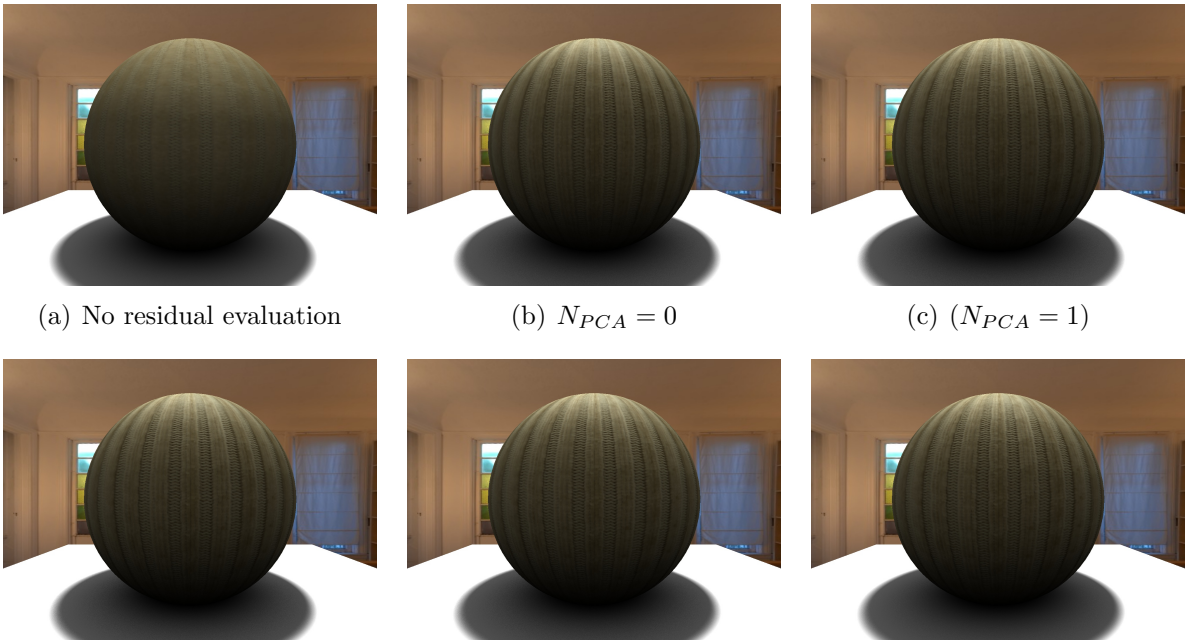


Figure 4: Partial Times for rendering the test scene with different average numbers of basis functions.



(a) No residual evaluation (b) $N_{PCA} = 0$ (c) $(N_{PCA} = 1)$

(d) $N_{PCA} = 2$ (e) $N_{PCA} = 3$ (f) $N_{PCA} = 4$

Figure 5: Test scene using the pulli material rendered with different quality approximations of the residual function.

(a) Pulli

N_{PCA}	FPS	MSSIM	RMSE
no-RES	81.32	0.950	5.687
0	72.91	0.976	2.067
1	66.63	0.981	1.715
2	60.91	0.987	1.352
3	55.52	0.991	1.044
4	50.67	1.000	0.000

(b) Wool

N_{PCA}	FPS	MSSIM	RMSE
no-RES	84.36	0.956	6.181
0	76.17	0.978	1.768
1	70.93	0.986	1.343
2	64.29	0.991	0.999
3	60.52	0.992	0.901
4	56.11	1.000	0.000

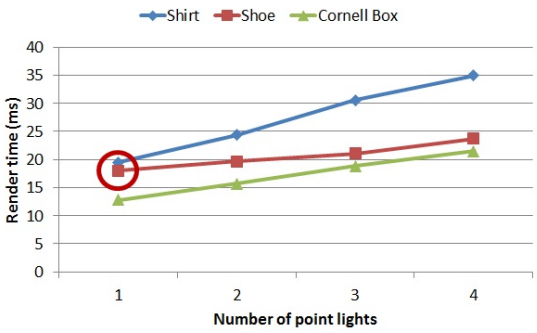
Table 4: Performance, MSSIM and RMSE for different quality approximations of the residual function

except for figure 6(a) where different numbers of point light sources are used.

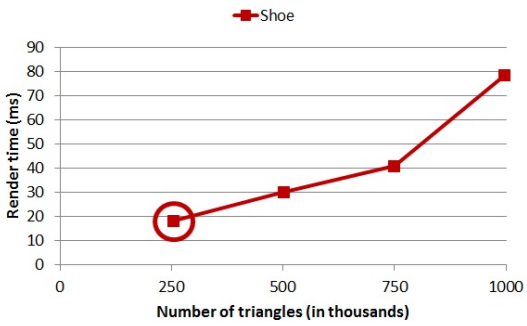
Figure 6(a) shows that rendering times increase linearly with the number of light sources. This is due to the entire evaluation of the data flow illustrated on figure 2. However, it is clear that the identification of the basis functions and respective parameters does not depend on the lighting directions; exploiting this fact in future implementations can improve scalability with the number of lights. Figure 6(b) illustrates rendering time as a function of the number of triangles for the "Shoe" scene. Time scales linearly with the geometry complexity, except when the number of triangles approaches one million. This scalability limit is met because scene data – including SPMMs and acceleration data structures – approaches the maximum OptiX user data allowed on this version. Particularly interesting is the time value highlighted with a red circle for the "Shoe" scene in both figures 6(a) and 6(b); this corresponds exactly to the same setup, except that on the former figure one point light source is being used, whereas on the latter figure progressive sampling of an area light source is being performed. Rendering times are approximately the same (18.00 ms and 17.96 ms, respectively), demonstrating that with the progressive approach interactivity is not impacted, the only cost being progressive refinement of the image quality.

An important aspect of the scalability analysis is understanding how does performance scales with the number of different SPMM textures visible within the image, i.e., the number of different visible materials that are each modeled with one SPMM. Since loading additional SPMMs requires additional storage space on the GPU, it is important to assess how does this scene property impact on rendering performance. Figure 6(c) shows that the rendering time scales sublinearly with respect to this parameter. In fact, it is the number of visible shading points that map onto a SPMM that determines the visualizer performance, as shown in figure 6(d). Note, however, that with 58% of the shading points (152K points out of a total of 512^2) mapped onto a SPMM the frame rate is still 36 fps.

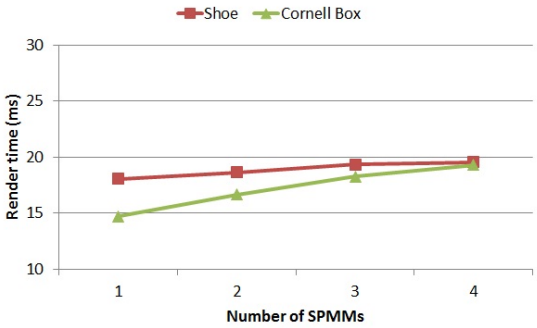
Figure 7 illustrates the results obtained with progressive ambient occlusion and area light source sampling. With no ambient occlusion the frame rate for the shoe scene is around 62 fps, but only points directly lit by the light source are visible (figure 7(a)). Turning ambient occlusion on decreases the frame rate to 45 fps but image quality is much better with this approximation to global illumination (figure 7(b)). The convergence process is noisy (figure 7(c)), but it becomes perceptually negligible after around 20 frames, which for



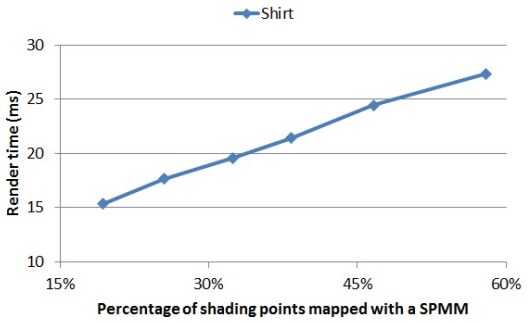
(a) Scalability with the number of point light sources



(b) Scalability with the number of geometric primitives

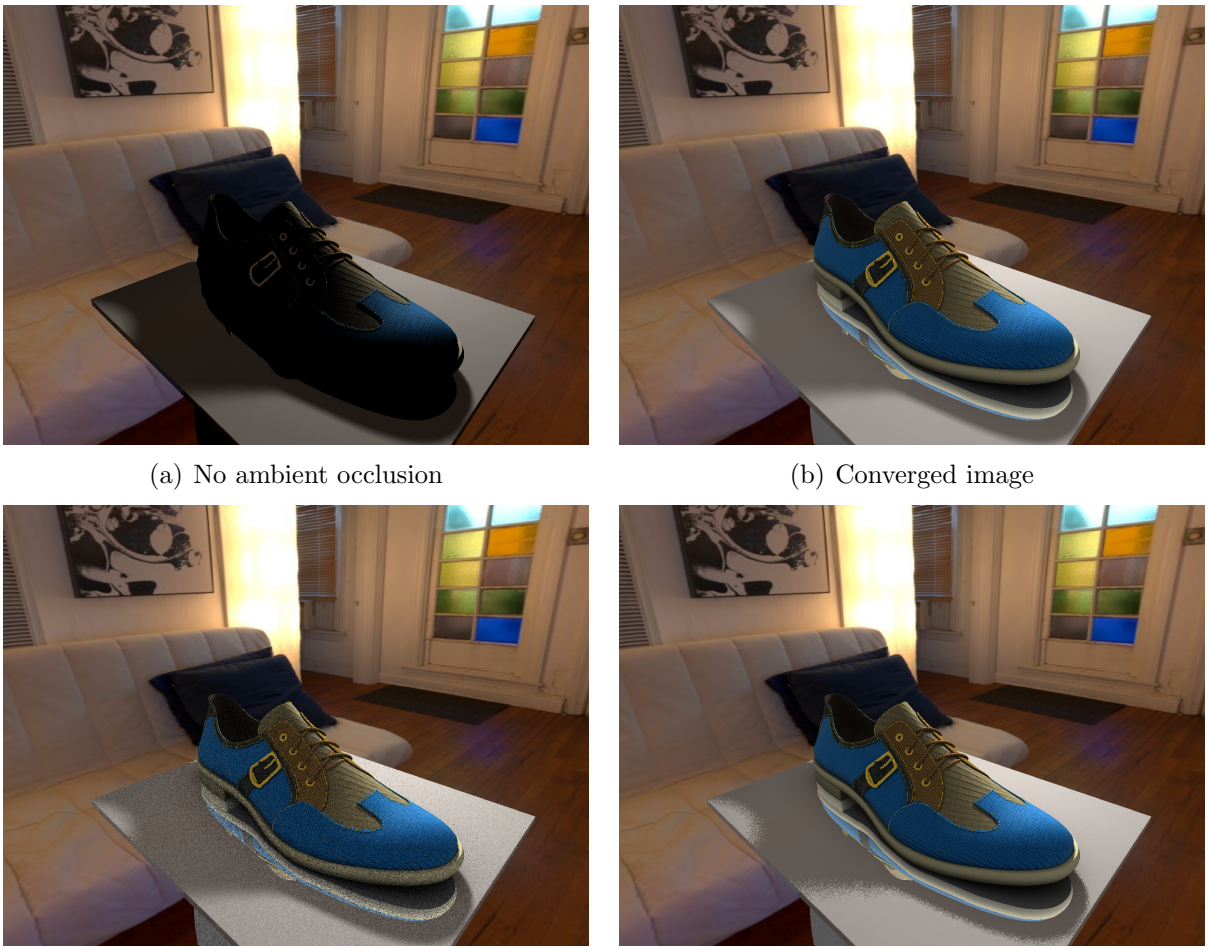


(c) Scalability with the number of visible SPMMs



(d) Scalability with the percentage of shading points that map onto SPMMs

Figure 6: Rendering times, in milliseconds, for a single frame while varying different scene configuration parameters. All timings were obtained with ambient occlusion on and one area light source, except for figure 6(a) where point light sources are used.



(a) No ambient occlusion

(b) Converged image

(c) Ambient occlusion converging

(d) Area light source sampling converging

Figure 7: Convergence of progressive area light sampling and ambient occlusion.

the current frame rate is less than half a second. With respect to progressive sampling of the area light source the results are similar. Figure 7(d) shows one of the first few frames on the convergence process; the penumbra regions are noisy, but noise becomes perceptually negligible in less than half a second. Progressive refinement has thus demonstrated to be a valuable tool to achieve simultaneously both high quality images and interactive frame rates.

5.4. Dynamic Scenes

Supporting camera walkthroughs, dynamic geometry, and changing light sources positions and intensity is straightforward, since, other than progressive area light sources sampling and ambient occlusion, all computations are repeated for each frame. The accompanying video, submitted as supplementary material, demonstrates the rendering performance under dynamic settings for the "Shoe" scene rendered at a resolution of 1280 x 1002 pixels.

Frames rendered with no ambient occlusion and with camera and lights movements achieve a frame rate above 25 fps, whereas under static settings the frame rate is around 38 fps. With progressive ambient occlusion the frame rate is around 27 fps under static settings and 22 fps with light or camera movements. Performance loss is due on both cases to the computation of the new camera position and resetting the progressive sampling accumulation buffers, but minimized due to the fact that the SPMM residual is not evaluated when the camera moves.

Dynamic geometry is also demonstrated, with a significant impact on rendering performance. For the "Shoe (x3)" scene (750K triangles) performance drops from 17 fps to between 3 and 5 fps. Performance loss is now due to the need to rebuild the Bounding Volume Hierarchy used to accelerate ray space traversal. Note that if object instancing was used, instead of replicating the whole shoe geometric description, then better results could be achieved.

6. Concluding Remarks

Interactive visualization of complex materials is of paramount importance for product designers on the footwear industry. Simulating the appearance of such complex materials can be achieved by resorting to compressed image based representations, such as the Bidirectional Texture Function (BTF), but interactive rendering with ordinary computing resources, while still being able to edit the compressed representation, remains to be demonstrated. In this paper we demonstrate that interactive global illumination rendering is indeed possible by resorting to the SPMM representation of BTFs and using a GPU ray tracing engine. The SPMM represents a material's appearance by resorting to a linear combination of well-known analytical BRDFs, which makes it well suited for intuitive appearance editing. The supported illumination model includes direct lighting from point and area light sources, perfect specular reflections and global illumination approximated with ambient occlusion. Maintaining interactive frame rates requires that data structures are carefully laid out on the GPU, particularly by resorting to texture memory to guarantee fast random access and high bandwidth. Area light sources sampling and ambient occlusion might require an arbitrarily large number of rays in order to reduce noise; we propose a progressive approach, which quickly converges to an high quality image while maintaining interactive frame rates by limiting the number of rays shot per frame.

Presented results demonstrated that interactive rendering is in fact possible, but they also demonstrate that evaluating the SPMM for each shading point still dominates rendering times. Further optimizations might be required, which might include precomputing the barycentric weights required to evaluate the residual function and storing them in high precision on an hemicube. The intersection tests associated with the view/lighting directions interpolation step would thus be substituted by texture lookups, eventually resulting on higher frame rates.

The proposed approach is scalable with respect to the number of light sources, geometric primitives and visible SPMMs. Rather than being sensitive to the number of visible SPMMs, performance is sensitive to the number of shading points that map onto SPMMs,

but results show interactive frame rates even for a large percentage of such points. Furthermore, acceptable performance is also demonstrated under dynamic settings, including camera movements, changing lighting conditions and dynamic geometry. The latter presents larger performance penalties due to the need to rebuild the ray space traversal acceleration structure.

For future work we intend to include physically based global illumination by adding progressive photon mapping on the rendering pipeline [35] and support glossy reflections by extending our current progressive sampling technique. Additionally, using displacement mapping techniques would increase image quality by providing a better rendering of the objects' silhouettes and by increasing depth perception [36].

Acknowledgments

Work partially funded by QREN project nbr. 13114 TOPICShoe and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within projectPEst-OE/EEI/UI0752/2011.

The authors would like to thank Professor Don Fussell for the discussions on SPMM algorithm, the University of Bonn for the BTF databases, Professor Hongzhi Wu for the technical support and Mind for supplying the footwear 3D models.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of http://dx.doi.org/10.1016/j.cag.2013.06.006.

Bibliographic References

[1] T. Weyrich, J. Lawrence, H. Lensch, S. Rusinkiewicz, T. Zickler, Principles of appearance acquisition and representation, ACM SIGGRAPH 2008 Classes (2008).

[2] K. Dana, B. van Ginneken, S. Nayar, J. Koenderink, Reflectance and texture of real-world surfaces, ACM Transactions on Graphics 18 (1999) 1–34.

[3] G. Müller, J. Meseth, M. Sattler, R. Sarlette, R. Klein, Acquisition, synthesis, and rendering of bidirectional texture functions, Computer Graphics Forum 24 (2005) 83–109.

[4] J. Filip, M. Haindl, Bidirectional texture function modeling: A state of the art survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 1921–1940.

[5] R. Ruiters, R. Klein, Btf compression via sparse tensor decomposition, Compututer Graphics Forum 28 (4) (2009) 1181–1188.

[6] V. Havran, J. Filip, K. Myszkowski, Bidirectional texture function compression based on multi-level vector quantization, Computer Graphics Forum (2010) 175–190.

[7] Y. Tsai, Z. Shih, K-clustered tensor approximation: a sparse multilinear model for real-time rendering, ACM Transactions on Graphics 31 (3).

[8] H. Wu, J. Dorsey, H. Rushmeier, A sparse parametric mixture model for btf compression, editing and rendering, Computer Graphics Forum 30 (2011) 465–473.

[9] S. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, M. Stich, Optix: a general purpose ray tracing engine, ACM Transactions on Graphics 29 (4).

[10] F. Suykens, K. Berge, A. Lagae, P. Dutré, Interactive rendering with bidirectional texture functions, Computer Graphics Forum 22 (2003) 463–472.
[11] D. McAllister, A. Lastra, W. Heidrich, Efficient rendering of spatial bidirectional reflectance distribution functions, in: ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, 2002, pp. 79–88.
[12] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, S. Worley, Texturing and Modeling: A Procedural Approach, Morgan Kaufmann Publishers Inc., 2002.
[13] M. Sattler, R. Sarlette, R. Klein, Efficient and realistic visualization of cloth, in: Eurographics Symposium on Rendering, 2003.
[14] Müller, J. Meseth, R. Klein, Compression and real-time rendering of measured btfs using local pca, in: Vision, Modeling and Visualisation, 2003, pp. 271—-280.
[15] M. Schneider, Real-time btf rendering, in: Central European Seminar on Computer Graphics for Students, 2004.
[16] M. Koudelka, S. Magda, Acquisition, compression and synthesis of bidirectional texture functions, in: Proc. of 3rd Int. Workshop on Texture Analysis and Synthesis, 2003, pp. 59–64.
[17] X. Liu, Y. Hu, J. Zhang, X. Tong, B. Guo, H.-Y. Shum, Synthesis and rendering of bidirectional texture functions on arbitrary surfaces, IEEE Transactions on Visualization and Computer Graphics 10 (3) (2004) 278–289.
[18] C. Schwartz, R. Ruiters, , M. Weinmann, R. Klein, Webgl streaming and presentation framework for bidirectional texture functions, in: International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), 2011.
[19] R. Furukawa, H. Kawasaki, K. Ikeuchi, M. Sakauchi, Appearance based object modelling using texture database: acquisition, compression and rendering, in: Eurographics Symposium on Rendering, 2002, pp. 257–266.
[20] M. Vasilescu, D. Terzopoulos, Tensortextures: multilinear image-based rendering, ACM Transactions on Graphics 23 (3) (2004) 336–342.
[21] H. Wang, Q. Wu, L. Shi, Y. Yu, N. Ahuja, Out-of-core tensor approximation of multi-dimensional matrices of visual data, ACM Transactions on Graphics 24 (3) (2005) 527—-535.
[22] Q. Wu, T. Xia, C. Chen, H.Lin, H. Wang, Y. Yu, Hierarchical tensor approximation of multi-dimensional visual data, IEEE Trans. on Visualization and Computer Graphics 14 (1) (2008) 186—-199.
[23] Y. Tsai, K. Fang, W. Lin, Z. Shih, Modeling bidirectional texture functions with multivariate spherical radial basis functions, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (7) (2011) 1356–1369.
[24] M. Haindl, J. Filip, Extreme compression and modeling of bidirectional texture functions, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (10) (2007) 1859–1865.
[25] W. Ma, S. Chao, B. Chen, C. Chang, M. Ouhyoung, T. Nishita, An efficient representation of complex materials for real-time rendering, in: ACM Symposium on Virtual Reality Software and Technology, New York, 2004, pp. 150–153.
[26] W. Ma, S. Chao, Y. Tseng, Y. Chuang, C. Chang, B. Chen, M. Ouhyoung, Level-of-detail representation of bidirectional texture functions for real-time rendering, in: Proceedings of the 2005 symposium on Interactive 3D graphics and games, I3D '05, 2005, pp. 187–194.
[27] J. Nocedal, A. Wächter, R. Waltz, Adaptive barrier update strategies for nonlinear interior methods, SIAM Journal on Optimization 19 (4) (2008) 1674—-1693.
[28] P. Zhao, B. Yu, Stagewise lasso, Journal of Machine Learning Research 8 (2007) 2701—-2726.
[29] P. Shanmugam, O. Arikan, Hardware accelerated ambient occlusion techniques on gpus, in: Proceedings of the 2007 symposium on Interactive 3D graphics and games, I3D '07, ACM, 2007, pp. 73–80.
[30] R. Fernando, GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, Pearson Higher Education, 2004.
[31] M. Pharr, R. Fernando, GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation, Addison-Wesley Professional, 2005.
[32] A. Lagae, P. Dutré, Compact, fast and robust grids for ray tracing, Computer Graphics Forum (Pro-

ceedings of the 19th Eurographics Symposium on Rendering) 27 (2008) 1235—-1244.
[33] M. Pharr, G. Humphreys, Physically Based Rendering: From Theory to Implementation, Morgan Kaufmann Publishers Inc., 2004, pp. 125–130.
[34] Z. Wang, C. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment:from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (2004) 600–612.
[35] C. Knaus, M. Zwicker, Progressive photon mapping: A probabilistic approach, ACM Transactions on Graphics 30 (3)).
[36] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, H. Shum, View-dependent displacement mapping, ACM Transactions on Graphics 22 (3) (2003) 334—-339.