Simulated Bidirectional Texture Functions with Silhouette Details

Mohamed Yessine Yengui * Pierre Poulin

LIGUM, Dept. I.R.O., Université de Montréal

ABSTRACT

The representation of material appearance requires an understanding of the underlying structures of real surfaces, light-material interaction, and human visual system. The Bidirectional Texture Function (BTF) describes real-world materials as a spatial variation of reflectance, which depends on view and light directions. Real BTFs integrate all optical phenomena occurring in a complex material, such as self-occlusions, interreflections, subsurface scattering, etc., independently of the mesoscopic surface geometry.

In this paper, we revisit BTF simulation to improve the modeling of surface appearance. In the recent years, computer graphics has achieved very good levels of image realism on geometrical appearance of 3D scenes. It is therefore logical to think that using this technology to simulate visual effects at the level of the mesoscopic geometry should provide even more realistic simulated BTFs. Our ultimate goal here is thus to produce material appearance as rich and as similar as those in reality, but relying more on the intuition and skills of artists, and on the rendering capacity of today's computer graphics.

We have designed a virtual parallel-projection / directional incident illumination framework that exploits rendering coherency in order to produce, in reasonable rendering times and with good compression ratios, BTFs of complex mesoscopic geometry, and this, even at grazing angles. Our current framework can simulate efficiently local interreflections effects within mesoscopic structures, as well as effects due to transparency, silhouettes, and surface curvatures. Our general simulation framework should also prove extensible to several other visual phenomena.

Index Terms: BTF, BRDF, aBRDF, simulation, surface appearance, compression, mesoscopic geometry, silhouette.

1 INTRODUCTION

The appearance of a surface is intricately related to how the surface reflects incident light. For homogeneous opaque surfaces without surface details that are individually visible, the bidirectional reflection distribution function (BRDF) encodes in a single 4D function the reflection at a given point for all combinations of directions of observation and incident illumination. For heterogeneous surfaces, the BRDF varies over the surface. A basic bidirectional texture function (BTF) encodes in 6D this spatial variation of surface reflection into a 2D texture, i.e., one 2D texture for every pair of observation and incident illumination directions (see Figure 2). A BTF thus captures all visual effects due to fine-scale self-shadowing, self-occlusions, interreflections, subsurface scattering, etc., that can affect or come from different points on the surface. A 4D BRDF thus extracted from one BTF texel can violate some BRDF properties, since light transfers can come from different points on the surface. It is referred to as an apparent BRDF (aBRDF) [27]. In this paper, we will refer to aBRDFs as BRDFs, for simplicity.

*e-mail: { yenguimy, poulin } @iro.umontreal.ca

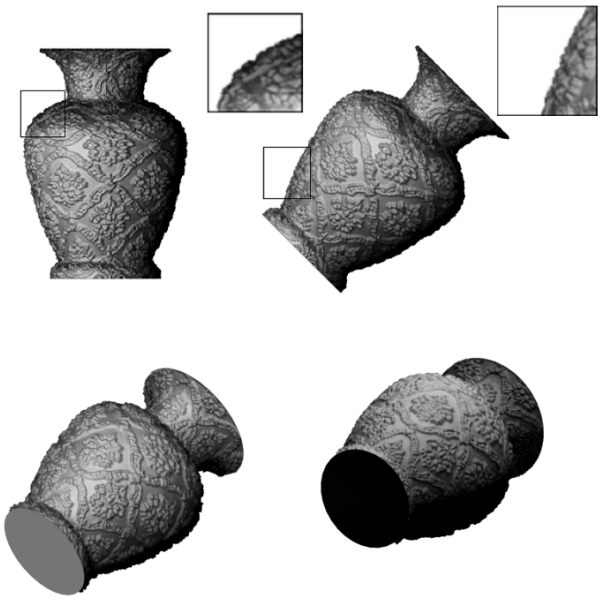

Figure 1: A vase with a ceiling pattern BTF, with proper shading, shadows, silhouettes, and light interreflections.

In spite of the capacity of a BTF to better reproduce the appearance of heterogeneous surfaces, major difficulties impede its full adoption by the graphics community. In fact, BTF acquisition is a long and difficult process that results in especially large datasets. It is therefore essential to optimize the acquisition process and to develop strong compression schemes for the resulting data.

Real-world BTF acquisition constitutes a rather delicate operation that requires the respect of several constraints of measure. The incident illumination must be uniform over the entire sample, the camera image must be registered properly to its sample texel, the measured surface properties must not change over time, etc. Nevertheless, even though major improvements have contributed to their technical quality, these physical measure devices often content themselves with small textures taken at low sampling resolutions of the light and view directions. They have also difficulties at grazing angles to capture the reflectance properties of materials or to guarantee uniform incident illumination.

On the synthetic front, computer graphics has greatly progressed to render efficiently images of high realism. By integrating the concept of hierarchies of models, computer graphics realizations can thus be exploited to simulate sophisticated and realistic light transport at the level of the surface details themselves, and use the results as higher-quality precomputed BTFs. Because we can control much more the environment than in real capture systems, several difficulties are therefore lifted or reduced. The benefits are that artists can better exploit this traditional rendering pipeline, and thus produce much more realistic surface details, tuned to their desires. The artists will therefore model the meso-structure scene like they

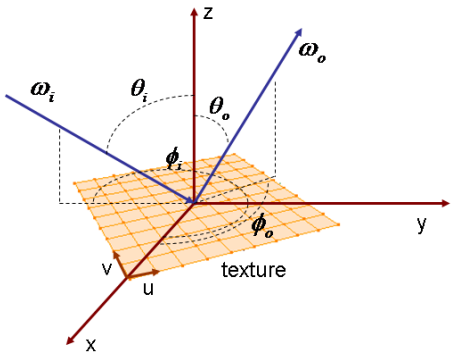

Figure 2: The different variables involved in a BTF representation, with its infinitely distant observation and illumination directions over a 2D texture.

would do for regular 3D scene, except that no light sources will be needed. The light source will come from the BTF simulation.

The general goal of our work is to conceive a framework for efficient BTF simulation, that can support several illumination phenomena, in order to create through simulation more realistic appearances for complex meso-structures on surfaces. Within this framework, this paper has three contributions that, although somewhat related to other work, are put together to allow us to achieve our goal. First, key to our approach is the use of a virtual parallel-projection camera / light simulation, thus enabling fine resolutions and reaching more grazing angles without the need for special registration. Second, to efficiently compute light interreflections, we propose a network of points in order to reuse common precomputed paths to evaluate the indirect illumination under a bidirectional path tracing approach. This frees us from the very high computational costs of entirely recomputing global illumination effects over a BTF. Finally, a BTF is always precomputed as a flat surface, but is often mapped onto curved surfaces. Our third contribution takes into account curvature, and adds very important transparency and silhouettes effects to the BTF-mapped surface with a companion 4D visibility map function (VMF) for a given BTF.

The rest of the paper is organized as follows. In Section 2, we present related work on BRDF/BTF measurements and simulations, and on compression schemes for the huge associated datasets. In Section 3, we detail each step of our algorithm, including the representation of BTFs, the efficient rendering, and the compression. Results are discussed in Section 4, before concluding.

2 Previous Work

2.1 BRDF/BTF Measurement

A number of physical BTF measurement systems [18, 9, 3] have been proposed with the aim to improve BTF measured quality. Differences between these devices are related to the type of measuring components and the degrees of freedom associated to each of these components. For some systems, the camera moves and the light source is fixed, whereas it is the opposite for others.

The technical improvement of BTF measurement systems still attracts the interests of several researchers. Numerous problems need to be resolved to reach this objective. They are mainly related to four challenges: (1) The limited number of the measured materials. Some materials cannot be captured with the existing systems, because they are too large, or too small. (2) The low resolution of some BTF sampling, resulting in numerous optical phenomena being under-sampled or simply missed. (3) The gigantic storage requirements for the datasets. (4) The precision of the measures and the stability of the systems still remain to be improved.

There have been huge advances in physical acquisition systems during the past two decades, including the types of phenomena captured, the precision, the compression schemes, etc., but the larger the data, the more anecdotal its availability.

To show the typical resolutions for current available BTF databases, we have found 61 materials from CUReT [4] (205 view/light directions of image resolution less than 640×480), six materials from Bonn [26] (81×81 view/light directions of image resolution of 256×256), six materials from UTIA [8] (81×81 view/light directions of image resolution of 512×512), and ten materials from UIUC [12] (about 90×120 view/light directions of image resolution of 480×360). These databases provide interesting data given the difficulties of scanning real materials, but for high-quality BTFs, much higher resolutions are needed. This is one additional reason why we opted for a synthetic framework to simulate BTFs. Finally, because the goal of our method is not to improve on the acquisition process, we do not discuss its related issues any further.

2.2 BRDF/BTF Simulation

Cabral et al. [2] were the first ones to propose a BRDF simulation technique. They approximate the BRDF over angular hemispherical bins by determining light reflected by each triangle from a mesoscopic geometry defined as a height field. Horizon mapping [16] is used to more efficiently approximate the proportion of light coming from the incident direction that is reflected by the material in the observed direction. The main drawback of this method is that it considers only direct illumination, is limited to height-field surfaces, and could not really exploit GPU in these early days. This is in fact partly what motivated us to look more into BTF simulations.

Westin et al. [33] simulate complete light reflection off the surface of an arbitrary micro-geometry structure. It can handle most illumination phenomena, since light transport is computed by Monte Carlo Path Tracing (MCPT). Unfortunately, it requires huge simulation times to sufficiently lower the inherent noise of the simulation, since MCPT is not well adapted to exploit path coherency. This is another aspect that we try to improve.

Suykens et al. [27] were the first ones to synthesize BTFs by passing a modeled meso-structure through a global illumination renderer (*RenderPark*). The synthesized BTFs (three models are shown in the paper) are generated at a fairly low angular resolution (e.g., 16×8).

Müller et al. [20] use a procedural method to define the material's micro-structure and then to enrich it with a BTF measured from real materials. They use a meso-structure constrained synthesis to combine the measured and the edited micro-geometries.

2.3 BTF/BRDF Compression

A BTF can be interpreted as a texture of BRDFs (actually aBRDFs). It is therefore possible to use BRDF compression methods on each BRDF. Some techniques [11, 17, 27, 13] have approximated the BRDF with a finite product of lower-dimension functions. Thus, a 4D BRDF can be approximated by a product of 2D or 1D functions. These functions are discretized into 2D textures, allowing for real-time illumination of a geometrical model, thanks to the GPU support of textures.

Other methods [22, 7] suggest to adjust the measured BRDF with an analytical model. This operation consists in determining the analytical model parameters in the least-squares sense. Pacanowski et al. [23] propose a fitting method of a measured BRDF material to the more general rational polynomials. Even though these methods provide very good compression rates, they require long computation times for the optimization operation to convergence. Furthermore, BRDF models are specific to a certain material type and are defined in a precise angular domain.

A third group of BRDF compression methods represents a BRDF by means of linear combination of basis functions, such as wavelets and spherical harmonics. Cabral et al. [2] and Westin at al. [33] use spherical harmonics to encode their BRDF in the form of a matrix of coefficients of its projection on a finite spherical harmonic basis function.

Sattler et al. [26] subdivide the BTF images into separate subsets based on the viewing direction, and then apply a principal component analysis (PCA) method. Koudelka et al. [12] use the singular value decomposition (SVD) technique; they determine k descriptive BRDFs to be a new basis of the BTF data. Müller et al. [19] observe that the BTF varies linearly in local zones and they combine methods of vector quantization and PCA on these local zones. Havran et al. [10] develop a method of BTF compression based on the hierarchical vector quantization. Ruiters and Klein [25] apply sparse representation based on K-SVD to each dimension separately, allowing for improved compression ratios. More recently, Tsai and Shih [28] bridge with K-CTA the gap between sparse representation of K-SVD and clustered tensor approximation (CTA). Their algorithm is quite flexible to represent high-dimensional datasets, exploits inter-cluster coherence, and they demonstrate its advantages on BTF data as well as view-dependent occlusion textures.

We will be using non-negative factorization from Lawrence et al. [13], because it provides good compression ratios, although not the current best ones, and it has other advantages. We will discuss it in more details in Section 3.4.

For more details on the BRDF/BTF compression, we suggest to visit the survey of Filip and Haindl [6].

2.4 Details at Silhouettes

Wang et al. [31] introduce a 5D structure called view-dependent displacement mapping (VDM). The VDM depends on the texture (u,v) resolution, the (θ,ϕ) viewing direction, and the curvature variation. The major drawback of this technique is that it can be applied only for height-field surface geometry. Wang et al. [32] propose a generalization of VDM, also as a 5D structure. It is computed at points on a regular 3D grid of a given meso-structure. It depends on position (x,y,z) and viewing direction (θ,ϕ) of the points. Wang et al. [30] have measured 4D structures to add silhouette details to a measured BTF-mapped surface. It is performed by rendering the front surface and the corresponding back surfaces and testing whether the distance between them is greater than the sum of the precomputed displacement. The major drawback of this method is that it neglects surface curvature variations.

Magda and Kriegman [15] separate shading from mesoscopic geometry, and encode the BTF into a stack of layers to represent the volumetric texture. They estimate from a captured real BTF a normal, an attenuation factor, a diffuse contribution, and parameters of a parametric reflectance model for each texel of each layer. The representation is suitable for real-time GPU display, but silhouettes may appear as superposed thin layers. While vertical panels between layers are proposed to bound the occlusion due to each volumetric texel, the modified shading on these panels is not discussed. No light interreflections are also assumed.

3 Our BTF Simulation Approach

If no storage is allowed, a very naive construction of a BTF at a 1° angular resolution would lead to (90×360) Z-buffer renderings $\times$ (90×360) shadow maps resulting in $1,049,760,000$ images, almost seven months at 60 fps (frames per second). This clearly calls for optimizations and approximations. Fortunately, synthetic BTFs are highly coherent.

In the case of BTF simulation, our goal is to achieve real-time rendering in the presence of multiple optical phenomena. Consequently, it remains necessary to precompute all possible common treatments for different light/view configurations in one BTF. Furthermore, it is compulsory to introduce some optimizations into the visibility computation within the BTF geometry, as well as into the mutual light transfers. With the aim of simplifying the problem, we propose to share the same set of direction vectors for the camera and for the light.

3.1 Parallel Camera / Directional Light

A standard physical camera used for the BTF capture operates an optical system (e.g., a telephoto lens) to make all incoming image rays parallel to each other. For various orientations of the camera over the BTF to be measured, the size of this flat texture decreases as a $\cos\theta$ factor according to angle θ between the normal of the texture plane and the view camera direction. Points at the center of every pixel thus change with every camera position. It is therefore necessary to suppose that locally, points around a certain zone observed within a camera pixel are sufficiently homogeneous. In spite of all these efforts, at oblique angles, this assumption does not hold true.

To reduce some of the problems associated to a physical camera, our capture system uses a virtual camera based on the parallel projection. The principle of this camera consists in intersecting the same positions of points located on the upper planar surface above the BTF, and this, from any camera position. We name this upper plane the *reference plane*; it is located horizontally, above the meso-geometry, normally set at the meso-geometry maximal height. The texels of the BTF are located on this plane.

Unlike the physical measurement system, our virtual device of simulation does not require a post-treatment for mutual registration. This treatment is made implicitly. In fact, for a *planar* meso-structure, points of direct intersections are identical for all the observation directions. They are different if the meso-geometry lies underneath the reference plane.

3.2 Simulation Approach

This section describes BTF construction, using hardware rendering for visibility determination, variance shadow maps for shadowing, and an adaptation of bidirectional path tracing for global illumination effects.

In global illumination, each ray intersecting point x of the meso-geometry has to evaluate the following equation:

$$L(x,\theta_o,\phi_o)=\int_{\Omega+} f_r(x,\theta_i,\phi_i,\theta_o,\phi_o)L_i(x,\theta_i,\phi_i)V(x,\theta_i,\phi_i)\cos\theta_i d\omega_i$$

where (θ_o,ϕ_o) are the polar coordinates of the view direction, (θ_i,ϕ_i) are those of the light direction, and f_r is the BRDF, all evaluated at point x. This equation describes the proportion of radiance reflected with respect to the flow of received radiance. L_i is the incident radiance from the ω_i direction. The binary visibility function $V(x,\theta_i,\phi_i)$ is efficiently computed with a variance shadow map [5]. Thus, for each intersected point, we simply project it on the precomputed shadow map for the corresponding light direction.

For several types of meso-geometry, global illumination effects can alter significantly the appearance of surfaces from different directions of viewing and lighting. However computing global illumination for any type of reflection under many light/view directions can prove a major undertaking. We have chosen to adapt the Bidirectional Path Tracing algorithm [29], and the rendering equation resolution is performed in two passes. First we generate points over the meso-geometry, and build an interconnection network between mutually visible points. For every camera direction, we ray-cast the meso-geometry, and store the direct hit points. Then, we connect the direct hit points to the network of points. The first connection gives the first bounce in the precomputed path. The next bounces are implicitly encoded in the network. Visibility/shadow map is computed from the direct intersection points. A visibility factor

(some form of ambient occlusion) is estimated for every point (direct point or network point), by shooting rays from its position. It is used to weight the reflected radiance L evaluated at a point. In the second pass, we evaluate the reflected radiance L for the precomputed paths. For each point (direct hit or network), light source visibility is determined using the corresponding variance shadow map. Russian roulette is used to determine whether the radiance ray evaluation is stopped before reaching a maximum number of bounces. The next section gives more details about our rendering algorithm.

3.2.1 Precomputations

Meso-geometry Sampling: We sample the 3D meso-geometry with a method proposed by Nehab and Shilane [21]. First a kd-tree bounding the meso-geometry is built. The resolution of the kd-tree controls the number of points to distribute over the surface, with one point per leaf node. The position of a sample point inside every leaf node of the kd-tree is refined, such that it lies on the meso-geometry and remains closer to the center of the node. Finally a minimal distance between points is enforced by deleting points too close to each other. In our description, we refer to the set of these points as the network points.

Network Interconnections: We use a standard ray tracer to interconnect the mutually visible network points. For every network point, we link it to at most one mutually visible network point. This is accomplished by sampling directions, finding nearby network points in this direction, and selecting one network point according to an importance sampling based on their form factors (normals and distance). The selected network point is retested for visibility. The links are naturally enclosed within the convex hull of the meso-geometry. They can interconnect network of points that lies in cavities. Several sets of connected points can result from this algorithm, the sets being disconnected from each other.

In this step, we also compute a visibility factor α for each network point x, as

$$\alpha(x) = \int_{\Omega+} V(x,\omega) d\omega \qquad (1)$$

where $V(x,\omega)$ is a binary function that equals 1 if x is occluded in direction ω, and 0 otherwise. It will be used later in the rendering pass to weight the radiance L evaluated on this network point. Typically, we use 256 directions.

Precomputing Paths for Each Sampling Direction and the Variance Shadow Map: Both the camera and the light source use parallel rays (parallel projection camera and directional light); they can therefore share the same structure. In this preprocessing step, we compute the direct hit points for each (θ,ϕ) direction. For each point, we store its triangle index and barycentric coordinates, as well as its material index. The barycentric coordinates are used to retrieve the local reference frame (normal and tangent vectors). Furthermore, in this step, a variance shadow map [5] is computed for each direction. The shadow value for a given image pixel is defined as the distance between the primary hit point (point p in Figure 3) and its projection on the reference plane (point q).

Light source visibility for each point of the meso-structure can be approximated by projecting it in the corresponding variance shadow map.

For every direct hit point, we link it to N_1 visible network points. We use $N_1 = 4$ or 6 in our simulations. A visibility factor α is also computed for every direct hit point.

In this phase, we precompute the described visibility data for a dense sampling of directions.

3.2.2 Rendering Pass

Visibility Query: The visibility test (Figure 4) is made as follows. A precomputed hit point p is viewed from direction ω_o. To determine if it is in shadow, its depth value in the associated light direction shadow map is computed. The corresponding pixel position r and the depth value q have been precomputed. p is visible if its distance to r is smaller than the distance of q to r. The variance shadow map reduces aliasing problems related to shadow map discretization. This method improves the standard shadow scheme by representing a distribution of depths at each texel. For details, we invite the readers to see the original variance shadow map paper [5].

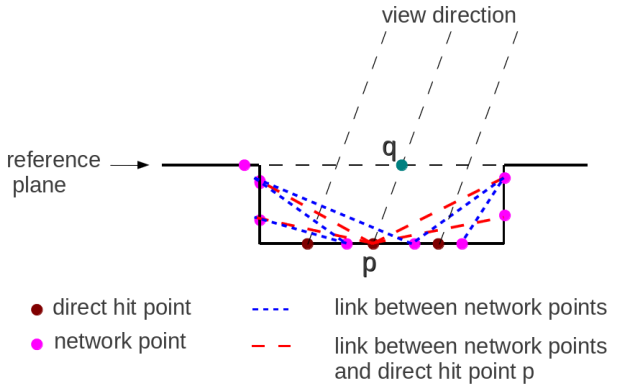

Figure 3: The precomputed data consist of primary hit points, a network of points with links to the direct points, and a variance shadow map.

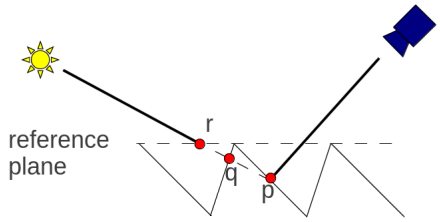

Figure 4: Visibility test.

For each pixel, the radiance is estimated using the precomputed paths in the network of points. These paths are implicitly initiated by linking the primary hit points to N_1 micro-geometry network points. Multi-bounce light interreflections are possible, considering each set of connected points can have several points. The light reflection at a contact point depends on the proprieties of the underlying material. The visibility of the light source from a point in the scene is determined by using the procedure described in the previous paragraph.

Global illumination is developed as follows, where index $i-1$ corresponds to the number of interreflections, and $\overrightarrow{y_{i-1}y_i}$ corresponds to the material reflection between two successive points (y_{i-1}, y_i) in one connected set of a network of points:

$$\begin{aligned}
L &= L_{dir} + L_{ind} \\
L_{ind} &= L_{ind_1} + L_{ind_2} + \cdots + L_{ind_i} \\
L_{ind_1} &= \frac{\alpha_1}{N_1} \sum_{i=1}^{N_1} \rho(\overrightarrow{\omega_o}, \overrightarrow{xy_1}) L_1(y_1) \\
L_{ind_2} &= \frac{\alpha_1 \times \alpha_2}{N_1} \sum_{i=1}^{N_1} \rho(\overrightarrow{\omega_o}, \overrightarrow{xy_1}) \rho(\overrightarrow{xy_1}, \overrightarrow{y_1y_2}) L_2(y_2) \\
L_{ind_i} &= \frac{\alpha_1 \times \cdots \times \alpha_i}{N_1} \sum_{i=1}^{N_1} \rho(\overrightarrow{\omega_o}, \overrightarrow{xy_1}) \cdots \rho(\overrightarrow{y_{i-2}y_{i-1}}, \overrightarrow{y_{i-1}y_i}) L_i(y_i).
\end{aligned}$$

Typically, we use paths of lengths 2 to 4, although longer paths are possible for meso-geometry prone to many interreflections.

Figure 5 shows changes of ceiling appearance for different view/light directions, computed with our proposed algorithm.

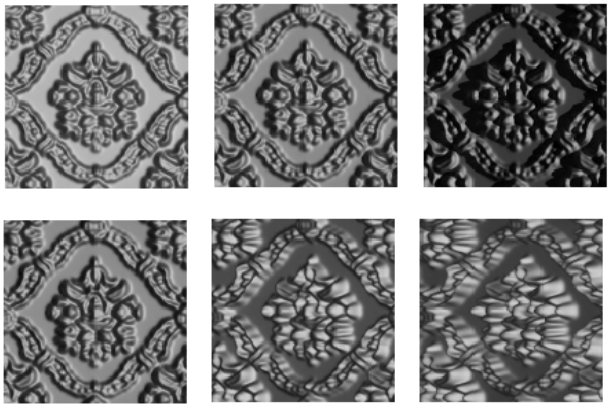

Figure 5: A few images used to construct the ceiling BTF, taken from different (top) light directions and (bottom) view directions.

3.3 Adding Silhouettes to BTF Rendering

Rendering a BTF mapped to any surface provides richer visual effects than standard texture mapping, because it can include effects such as self-shadowing, interreflections, mutual occlusions, etc. However, it does not reproduce detailed displaced silhouettes nor does it take into account occlusions due to curvature variations. This is because a BTF is an image-based technique without geometry. It encodes all the effects caused by the meso-structure without explicitly modeling the underlying geometry.

To solve this problem, we augment the BTF with a 4D visibility map function (VMF). This function is none other than our precomputed variance shadow map. It is defined as $VMF(u,v,\theta,\phi)$, where (u,v) are the texture coordinates and (θ,ϕ) the spherical coordinates of the view direction. It corresponds to the distance from p to q in Figure 3.

Computing the Texture Coordinates and View Direction: We begin by extruding the basis mesh of the surface, on which the BTF is mapped. Each triangle vertex is extruded along its interpolated surface normal direction to generate a volumetric mesh (defined by triangle slabs, forming a one-layer mesh). We try to avoid self-intersecting extrusions, but this remains an open problem for complex intertwined geometry. To render each image pixel, we need to ray march through the extruded meso-structure until we identify an intersection with the surface details or reach its base (if it is opaque). The marching proceeds simultaneously in object space and texture space. However, one needs to remember that we do not have access anymore to the meso-geometry in a BTF.

An example of the process is illustrated in Figure 6. For the pixel ray, we determine its entry point p_1 in the volumetric mesh and its corresponding texture coordinates t_1, as well as its exit points p_2 and t_2 of this extruded triangle slab in both spaces. The segment (t_1,t_2) gives the view direction $V(\theta,\phi)$ in texture space. On the reference plane, the entry point t_1 is identical to the view direction t_0, which encodes the VMF distance $VMF(t_0,\theta,\phi)$, defining the sampled shortest distance to the (now absent) meso-geometry. As $VMF(t_0,\theta,\phi) > (t_2 - t_0)$, we assume there are no intersections in this extruded triangle slab, and the ray continues to the next triangle slab, i.e., here defined as segments (p_2,p_3) and (t_2,t_3). In texture space, (t_2,t_3) has its VMF entry located on the reference

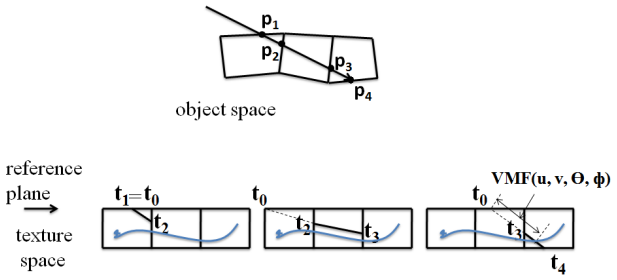

Figure 6: Top: Ray intersection with the extruded surface in object space. Bottom: The corresponding ray segments in volumetric texture space. Note that the surface in light blue is given as an indication, as no meso-geometry is present at this stage in the BTF representation.

plane in the new coordinates t_0, where it is accessed. Here again, no intersections are found as $VMF(t_0,\theta,\phi) > (t_3 - t_0)$, so the ray continues. Finally for segments (p_3,p_4) and (t_3,t_4), and their associated new t_0, $VMF(t_0,\theta,\phi) \leq (t_4 - t_0)$ and we consider there is an intersection with the meso-geometry.

Computing the Light Direction: Up to now, we have determined texture coordinates (u,v), view direction $\omega_o(\theta,\phi) = V(\theta,\phi)$, and intersection distance from the VMF associated data. We can transfer the associated intersection point in object space (and world space) to find out the light direction from this intersection. To compute the light direction $\omega_i(\theta,\phi)$ in texture space, we trace in object space a ray from the intersection point up to the light direction, until we escape the volumetric mesh. The same segments in object space can be recomputed in texture space as with the view segments of the previous paragraph. If no segments are occluded, we use the corresponding ω_i from the slab where the view intersection occurred. This is our default light direction. Otherwise, we update the light direction from the slab where an intersection occurs, going from inside to outside. Finally, the shading value of the current ray is given directly from the $BTF(u,v,\omega_o,\omega_i)$.

3.4 BTF Compression

We need to compress the huge amount of data resulting from the BTF rendering step, in the order of GBs, to a much smaller size, in the order of MBs, which can be efficiently used later in a rendering application. We apply the factorization method from Lawrence et al. [13] to each individual texel (aBRDF) computed by our BTF simulation. The method has the advantage to be applicable on any BRDF parameterization, and thanks to its non-negative factors, the method can provide an importance sampling function for the factored BRDFs, which is very useful when integrated in a rendering system based on sampling.

Recall the following from the original method. Non-Negative Matrix Factorization (NMF) as proposed by Lee and Seung [14] decomposes the matrix of the BRDF parameterized in the $(\vec{\omega_o}, \vec{\omega_i})$ basis, ensuring that no entries result in negative values. The factorization problem can be mathematically formulated as follows: Given a data matrix $V_{n,m}$ of dimensions (n,m), with every matrix element $V[i,j] \geq 0$, and a positive integer $k \leq \min(n,m)$, we compute two data matrices $W_{n,k}$ and $H_{k,m}$ such that $V \approx WH$ with the goal of minimizing the difference between V and WH.

The proposed factorization is quite similar to the one from Lawrence et al. [13], with the difference that we use the Singular Value Decomposition (SVD) method to automatically determine the value for k. Mathematically, the formula used is the following:

Meso-geometry	Direction resolution	Texture resolution	Samples per texel	Shadow map resolution	Number of network points	Precomputation time (in mins)	Rendering time (in mins)
ceiling (900k)	16×32	128×128	6×6	768×768	2,521,853	70 min	130 min
braiding (200k)	16×32	64×64	6×6	384×384	3,623,758	13 min	36 min
colored bump (350k)	20×40	32×32	8×8	256×256	2,855,710	15 min	45 min
leather (820k)	16×32	128×128	6×6	768×768	2,043,534	56 min	145 min
brick (860k)	16×32	128×128	6×6	768×768	1,933,122	58 min	142 min

Table 1: Statistics to build a BTF and its precomputed data: Number of directions (view or light), BTF texture resolution in the number of BRDFs (texels), samples ray traced per texel, shadow map resolution, and timings for precomputation and rendering stages.

$$\rho(\omega_o, \omega_i) = \sum_{j=1}^{J} F_j(\omega_o) G_j(\omega_i). \tag{2}$$

Intuitively, BRDFs that exhibit high variations over the light/view directions need more factors than low-variation BRDF data. Using the SVD, we compute the singular value $\{\lambda_i\}_{i=1,\ldots,\min(n,m)}$ of matrix $V_{n,m}$. Parameter k is defined as the smallest integer value that satisfies the following inequality:

$$\left(\sum_{i=1}^{k} \lambda_i\right) / \left(\sum_{i=1}^{\min(n,m)} \lambda_i\right) \geq \beta, \qquad \text{for } \beta \in [0,1]. \tag{3}$$

We have used $V_{n,m}$ singular vectors to initialize the $W_{n,k}$ and $H_{k,m}$ matrices for the factorization algorithm, as described in [1].

At this point, suppose that every BRDF (texel of a BTF) is factorized into k factors depending on (θ_o, ϕ_o), called H factor, and k factors depending on (θ_i, ϕ_i), called W factor. To exploit the repetitive feature of the underlying BRDFs that could come from very similar texels of a BTF, and therefore, their H and W factors, we simply apply the standard PCA algorithm on each of the factor sets.

4 Results and Discussions

We analyze in this section the computational behavior of our BTF simulation application. Performance measurements were computed on an Intel i7 920 2.80 GHz with 12 GB of CPU memory and an Nvidia GTX 480 GPU.

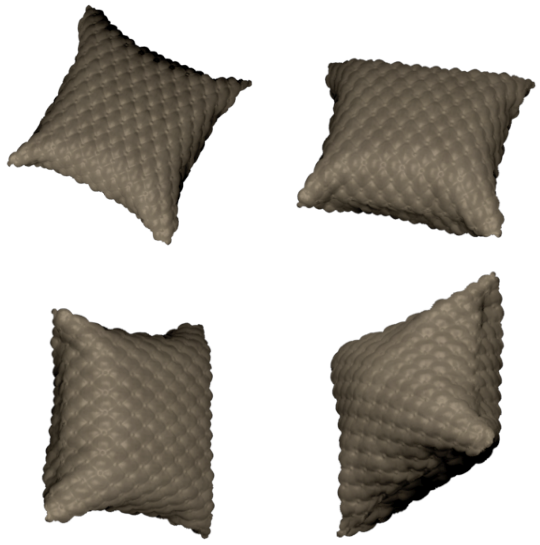

Figure 7: A pillow with a leather BTF.

We use the Nvidia GPU's Optix library for all precomputed data and for the proposed rendering algorithm. Table 1 gives the direction sampling rate of the meso-structure upper hemisphere, the resolution of each BTF, precomputing times, and data sizes for constructing our BTFs. Precomputations include meso-geometry sampling, visibility factors, point network interconnections, and path connections. Renderings in precomputations include producing each BTF image, including global illumination effects. Note that final rendering in the 3D scene is performed from within seconds (for quality rendering) to multiple times per second (for interactive rendering with ray tracing on GPU).

We have noticed that computing times depend basically on the number of polygons in the meso-geometry and on the interreflections simulated on the meso-structure surface. Whereas, data sizes depend only on the parameters resolution.

All VMF data are sampled under 16×32 viewing directions with a texture of resolution (128×128). The complete size of the VMF structure is about 16 MB. Given this smaller data size, we have not compressed it, even though this would be quite easy to do.

Compression takes about 3 hours for a BTF with a texture of resolution 64×64, and 12 hours for 128×128. In our simulated BTFs, parameter β ranges between 0.7 and 0.8. With our BTF simulation application, we are able to synthesize our own BTFs with finer sampling directions resolution ($16 \times 32 = 512$, $20 \times 40 = 800$ directions) compared to the Bonn University [18] real measurement system (81 directions) and the simulated approach of Suykens et al. [27] (50 directions, up to 128). Note that it is still possible to increase our BTF resolutions, but we have to invest more time, especially in the compression stages.

Meso-geometry	Data orig.:compress.		Ratio	RMSE
ceiling (900k)	48.4 GB	147 MB	1:330	0.073
braiding (200k)	12.1 GB	50 MB	1:240	0.062
colored bump (350k)	7.5 GB	40 MB	1:190	0.068
brick (860k) (direct illumination)	48.4 GB	130 MB	1:372	0.071
brick (860k) (global illumination)	48.4 GB	108 MB	1:448	0.065
leather (820k)	48.4 GB	150 MB	1:323	0.076

Table 2: Simulated BTFs and achieved compression ratios.

All our BTF-mapped surfaces are rendered with PBRT [24] (physically-based rendering) ray tracer, with 16 rays per pixel. As image rendering time is concerned, the pillow with braiding (Figure 12) with 1 ray per pixel takes less than 2 seconds. This scales representatively for the rendering of all our images. While not yet completed, we believe that an efficient hardware rendering implementation should result in real-time display of a surface with one BTF.

A vase with BTF taken from a ceiling pattern is shown in Figure 1, and a leather has been applied on a pillow (Figure 7) and a chair (Figure 8). These images show correct direct illumination and shadows, silhouettes, and interreflections under different lighting and view directions.

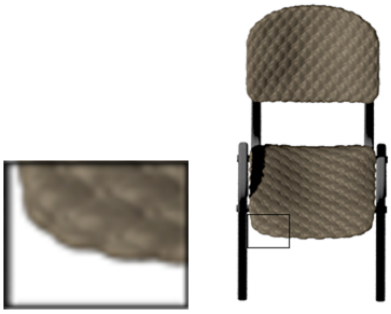

Figure 8: A chair with a leather BTF.

Figure 9 compares the effect of adding indirect illumination on the BTF appearance of bricks. It appears mainly between the bricks, where the interreflections of light "soften" the shadows. Figure 12 shows that using our VMF structure, we are also able to simulate BTFs that exhibit transparency effects, without the need of an extra alpha channel in the BTF data.

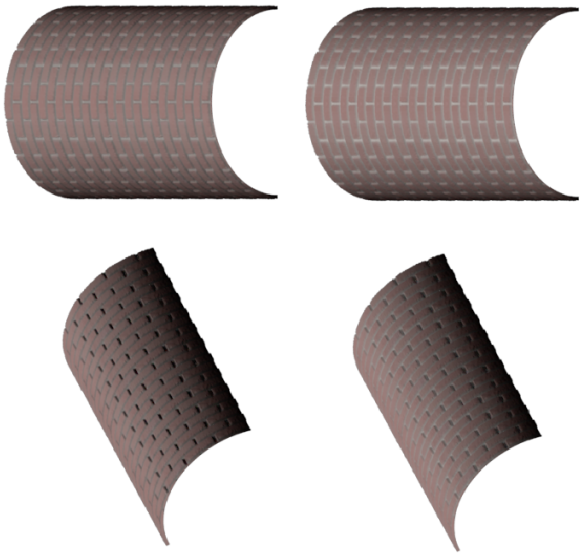

Figure 9: A surface with a BTF made of bricks. Left: under local illumination. Right: under global illumination. The space between bricks appears brighter.

We compare in Figure 10 a compressed BTF of colored bumps to an uncompressed version of the same BTF. We can notice that distortions are more visible at the shadows, especially at grazing angles due to high discontinuity at these angles. The softer shadows however tend to be visually less objectionable.

To further validate our results, we have implemented a Monte Carlo Path Tracer to render ground-truth images with the meso-geometry itself. We can see in Figure 11, with the ground truth on the left column and our BTF mapping method on the center column, that our BTFs can efficiently approximate the ground-truth solution. The distortion in the surface appearance is mainly related to effects due to compression and BTF discretization. The difference in the computing times between the ground truth and our method is about a factor of 8 to 12 times. Rendering a BTF mapped over a surface takes about 30 to 60 seconds (using PBRT), compared to 300 to 700 seconds required for rendering the actual instantiated meso-

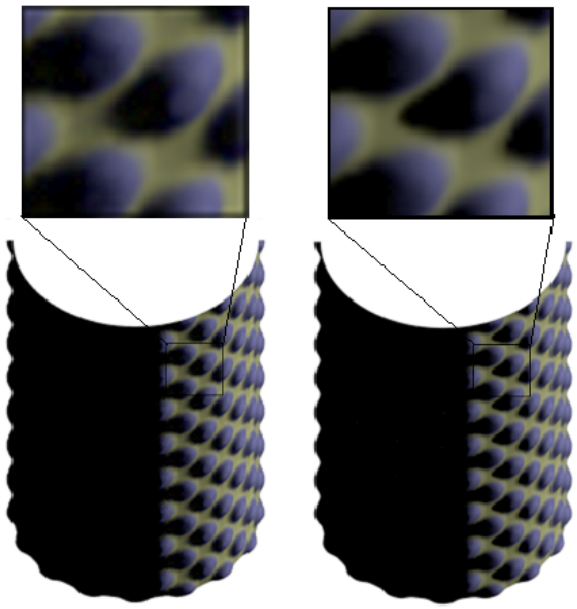

Figure 10: BTF of colored bumps that are (left) compressed and (right) not compressed.

geometry (our ground truth) to get a similar high-quality image. In Figure 11 (right), we show how different the shading appears when the light direction in the BTF is not traversed across all deformed texels, but computed directly where the the view ray intersects the surface geometry (slab).

Figure 13 depicts an example of a BTF that cannot be reduced to a height field. The underlying meso-geometry is composed of an array of spheres, with red spheres located at a height of twice their radius, and green spheres at three times their radius. We have simulated the BTF with global illumination. The BTF and the meso-geometry are both applied over a section of a large cylinder, to exhibit effects due to slight curvature and silhouettes. For the BTFs in the left column, we used diffuse and specular coefficients of 0.7 and 0.3, respectively. For the third BTF, we switched them to 0.3 and 0.7, respectively. Our simulated BTFs are compared with ground truths in Figure 13 right. We are able to effectively generate correct silhouettes, highlights, self-shadows, and subtle indirect illumination effects. Some small artifacts are basically related to the VMF/BTF discretization and compression. Some differences are also due to our improper tone mapping.

5 Conclusion

In this paper, we have introduced a pipeline to simulate complex material appearance described with a 6D bidirectional texture function (BTF). The artist only needs to generate a meso-geometry, like he would do to model normal 3D scenes. The simulated BTF can then be applied on any parameterized surface in a 3D scene, and the BTF provides the resulting complex surface appearance. We have decomposed our application in three principal parts:

1. Precomputations: We precompute all possible common treatments between BTF images. This includes an interconnection network for light transfer within the BTF meso-geometry that is used by all the possible view/light directions. Other than determining the direct hit points, we precompute a variance shadow map for each sampled direction to approximate the light source visibility in the rendering step.

2. Rendering: Using the precomputed data, we approximate global illumination with an adaptation of bidirectional path tracing. Although our solution is biased, we obtain good results that simulate several visual phenomena.

3. Silhouettes: Using a precomputed visibility map function (VMF), we render proper shaded silhouettes with BTF-mapped surfaces and can take into account curvature surface variations.

The results correspond to our expectations for efficient construction of high-quality BTFs that can then be rendered in high-quality images or in interactive applications. This gives more flexibility to artists to increase the quality of their realistic surfaces.

With this first step achieved, we believe that there are other difficulties that need to be resolved. Our quality BTFs are still measured in tens to a few hundreds of MB, but we believe other BRDF compression schemes could be used, e.g., [23], which should prove stronger, but at the cost of longer compression times. A framework for automatically dividing the BTF in subspaces manageable in memory should also alleviate the current maxima limits for certain BTF dimensions. Finally, other simulation dimensions, including time variations and subsurface scattering, should be analyzed to benefit from their own high coherencies.

Just like BRDFs, which are becoming more and more frequent in today's applications, BTFs form a powerful and flexible description of material appearance. We believe they will become more popular as their difficulties are pushed further away.

Acknowledgements

The authors thank the anonymous reviewers for their constructive comments. Mohamed Yessine Yengui acknowledges a fellowship from the government of Tunisia. This work was partly supported by NSERC.

References

[1] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recogn.*, 41(4):1350–1362, Apr. 2008.

[2] B. Cabral, N. Max, and R. Springmeyer. Bidirectional reflection functions from surface bump maps. In *Proc. SIGGRAPH '87*, pages 273–281, 1987.

[3] K. J. Dana. BRDF/BTF measurement device. In *IEEE Intl. Conf. on Computer Vision*, volume 2, pages 460–466, 2001.

[4] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, 1999.

[5] W. Donnelly and A. Lauritzen. Variance shadow maps. In *Proc. Symposium on Interactive 3D Graphics and Games*, pages 161–165, 2006.

[6] J. Filip and M. Haindl. Bidirectional texture function modeling: A state of the art survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):1921–1940, 2009.

[7] A. Forés, S. N. Pattanaik, C. Bosch, and X. Pueyo. BRDFLab: A general system for designing BRDFs. In *Proceedings CEIG'09*. Eurographics, 2009.

[8] M. Haindl, J. Filip, and R. Vacra. Digital material appearance: the curse of tera-bytes. *ERCIM News*, 90:49–50, 2012.

[9] J. Y. Han and K. Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. In *SIGGRAPH '03*, pages 741–748, 2003.

[10] V. Havran, J. Filip, and K. Myszkowski. Bidirectional texture function compression based on multi-level vector quantization. *Computer Graphics Forum*, 29(1):175–190, Jan. 2010.

[11] J. Kautz and M. D. McCool. Interactive rendering with arbitrary BRDFs using separable approximations. In *Proc. Eurographics Workshop on Rendering '99*, pages 247–260, 1999.

[12] M. L. Koudelka, S. Magda, P. N. Belhumeur, and D. J. Kriegman. Acquisition, compression, and synthesis of bidirectional texture functions. In *ICCV '03 Workshop on Texture Analysis and Synthesis*, pages 59–64, 2003.

[13] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient BRDF importance sampling using a factored representation. In *Proc. SIGGRAPH '04*, pages 496–505, 2004.

[14] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press, 2000.

[15] S. Magda and D. Kriegman. Reconstruction of volumetric surface textures for real-time rendering. In *Eurographics Symposium on Rendering*, pages 19–29, 2006.

[16] N. L. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, 1988.

[17] M. D. McCool, J. Ang, and A. Ahmad. Homomorphic factorization of BRDFs for high-performance rendering. In *Proc. SIGGRAPH '01*, pages 171–178, 2001.

[18] G. Müller, G. H. Bendels, and R. Klein. Rapid synchronous acquisition of geometry and BTF for cultural heritage artefacts. In *International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, pages 13–20, Nov. 2005.

[19] G. Müller, J. Meseth, and R. Klein. Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation*, pages 271–280, Nov. 2003.

[20] G. Müller, R. Sarlette, and R. Klein. Procedural editing of bidirectional texture functions. In *Eurographics Symposium on Rendering 2007*, June 2007.

[21] D. Nehab and P. Shilane. Stratified point sampling of 3D models. In *Eurographics Symposium on Point-based Graphics*, pages 49–56, June 2004.

[22] A. Ngan, F. Durand, and W. Matusik. Experimental analysis of BRDF models. In *Proc. Eurographics Symposium on Rendering*, pages 117–226, 2005.

[23] R. Pacanowski, O. S. Celis, C. Schlick, X. Granier, P. Poulin, and A. A. M. Cuyt. Rational BRDF. *IEEE Trans. Vis. Comput. Graph.*, 18(11):1824–1835, 2012.

[24] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[25] R. Ruiters and R. Klein. BTF compression via sparse tensor decomposition. In *Eurographics Symposium on Rendering 2009*, pages 1181–1188, 2009.

[26] M. Sattler, R. Sarlette, and R. Klein. Efficient and realistic visualization of cloth. In *Eurographics Symposium on Rendering*, June 2003.

[27] F. Suykens, K. V. Berge, A. Lagae, and P. Dutré. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum (Proc. Eurographics)*, 22(3):463–472, 2003.

[28] Y.-T. Tsai and Z.-C. Shih. K-clustered tensor approximation: A sparse multilinear model for real-time rendring. *ACM Trans. Graph.*, 31(3):19:1–17, 2012.

[29] E. Veach and L. J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proc. SIGGRAPH '95*, pages 419–428, 1995.

[30] J. Wang, X. Tong, J. Snyder, Y. Chen, B. Guo, and H.-Y. Shum. Capturing and rendering geometry details for BTF mapped surfaces. *The Visual Computer*, 21(8-10):559–568, 2005.

[31] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. *ACM Trans. Graph.*, 22(3):334–339, July 2003.

[32] X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. Generalized displacement maps. In *Proc. Eurographics Symposium on Rendering*, pages 227–233, 2004.

[33] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. In *Proc. SIGGRAPH '92*, pages 255–264, 1992.

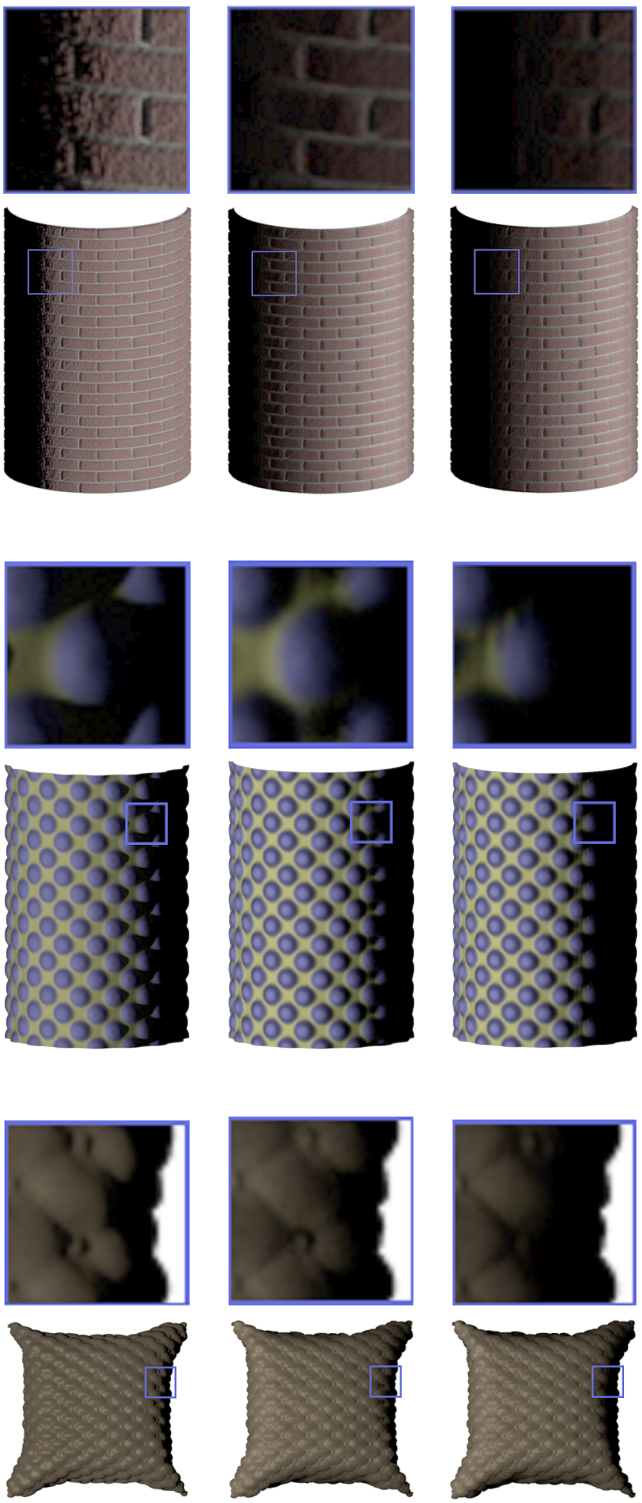

Figure 11: Our BTF mapping method closely matches ground truth computed with Monte Carlo Path Tracing. Left: ground truth. Center: BTF. Right: BTF without deformed texel traversal.

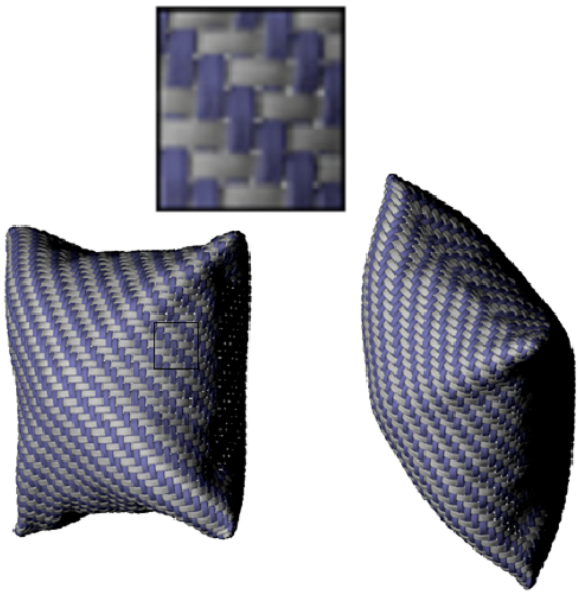

Figure 12: Pillow with a BTF made of braids. Note how transparency is properly simulated, given the pillow is empty and backface culling is enabled, in order to increase transparency effects.

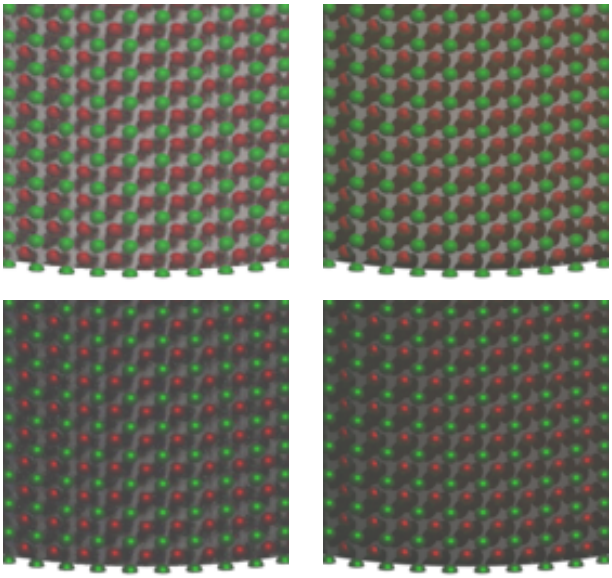

Figure 13: A slightly curved surface with a BTF made of an array of spheres. Left: BTF under global illumination. Right: Ground truth. Bottom: BTF and ground truth with higher specular reflection.