LBNL-5954E

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

A Practical Framework for Sharing and Rendering Real-World Bidirectional Scattering Distribution Functions

G. Ward
Anyhere Software

M. Kurt
International Computer Institute

N. Bonneel
Harvard University

Windows and Envelope Materials Group
Building Technology and Urban Systems Department
Environmental Energy Technologies Division

September 2012

DISCLAIMER

A Practical Framework for Sharing and Rendering Real-World Bidirectional Scattering Distribution Functions

Greg Ward[1], Murat Kurt[2], and Nicolas Bonneel[3]

[1]*Anyhere Software,950 Creston Rd., Berkeley CA, USA*
[2]*International Computer Institute, Ege University, Turkey*
[3]*School of Engineering & Applied Sciences,Harvard University, USA*

Abstract

The utilization of real-world materials has been hindered by a lack of standards for sharing and interpreting measured data. This paper presents an XML representation and an Open Source C library to support bidirectional scattering distribution functions (BSDFs) in data-driven lighting simulation and rendering applications.The library provides for the efficient representation, query, and Monte Carlo sampling of arbitrary BSDFs in amodel-free framework. Currently, we support two BSDF data representations: one using a fixed subdivision of thehemisphere, and one with adaptive density. The fixed type has advantages for certain matrix operations, while theadaptive type can more accurately represent highly peaked data. We discuss advanced methods for data-drivenBSDF rendering for both types, including the proxy of detailed geometry to enhance appearance and accuracy.We also present an advanced interpolation method to reduce measured data into these standard representations.We end with our plan for future extensions and sharing of BSDF data.

1.Introduction

Accurate light scattering off surfaces is critical to realism for computer graphics and physically-based rendering. While many different reflectance models have been developed, the world of real materials is too diverse to fit a single mathematical model (Ngan et al., 2005). Multi-lobe BRDFs come closestto generality (LaFortune et al., 1997, Ashikhmin and Shirley, 2000; Yu et al., 2010), but can fail when reflectance behavior does not fit the parameterization. Few models address transmission, and none can handle complex fenestration systems (CFSs) designed to convey daylight inspecialized ways, such as prismatic glazings, holographicfilms, and specular louvers. This is becoming an increasinglyimportant topic in modern building design (Gayeski et al., 2009).

Data-driven BSDFs offer a more general alternative to mathematical models, but they present a number of challenges. First, there is the question of how to parameterize the function – a grid of elevation and azimuth values is simple but inefficient. How do we represent a BSDF with strong, localized peaks? How do we generate importance based Monte Carlo samples quickly and efficiently? How do we cope with the common case of missing data in our measurements? We must address each of these questions before we can claim to have a practical solution.

Our long-term goal is to provide a practical framework for sharing model-free BSDF data between the lighting simulation and graphics communities. To accomplish this, we need an interchange and application standard that is simple, efficient, and transparent. We propose such a standard with a library implemented in C, whose API is more general than current XML representations. This library is flexible enough to be updated with improvements to the underlying file standard, and is already being used by industry.

We begin with a summary of relevant work on data-driven BSDF models, followed by an explanation of our fixed and variable representations, with examples of each. We demonstrate additional data-driven BSDF rendering techniques using a version of the *Radiance* lighting simulation and rendering system

(Larson and Shakespeare,1998). Finally, we present an advanced interpolation method for converting incomplete BSDF measurements to a usable form. We end with a discussion of the method and future directions.

2. Background and RelatedWork

The Bidirectional Scattering Distribution Function (BSDF) is a four-dimensional function akin to the BRDF that describes how incident light over both the front and back hemispheres of a surface is scattered in front of and behind a surface. The BSDF $f_r()$ can be defined in terms of the *rendering equation*(Kajiya, 1986):

$$R(\vec{\omega}_o) = \int_\Omega R(\vec{\omega}_i) f_r(\vec{\omega}_i; \vec{\omega}_o) \, |\vec{\omega}_i \cdot \vec{n}| d\Omega_i. \quad (1)$$

The semicolon between the arguments of $f_r()$means that the two$\vec{\omega}$'s may be swapped without changing the value of the function – hence its bidirectional nature (a.k.a. reciprocity). Essentially, the rendering equation states that the outgoingradiance in the direction $\vec{\omega}_o$equals the integral of incident radiance from all directions multiplied by $f_r()$with a cosinecorrection. The $d\Omega_i$term is common notation for incident solid angle from the direction$\vec{\omega}_i$and would include $\sin\theta_i$ if given as $\theta_i$and ϕ_i.

The unit of the BSDF is$steradians^{-1}$. Since integrating the cosine over the incident hemisphere yields a value of π, a Lambertian reflector has a value of ρ/π for all$\vec{\omega}_o$, $\vec{\omega}_i$on the same side, and zero elsewhere. Similarly, a translucent diffuser has a$f_r()$value of τ/π on the opposite side, where $\rho + \tau \leq 1$ for energy balance. The values for ρmay be different on the front and back sides, but reciprocity implies τmust be the same in either direction.

Eq. (1) may be considered on a per-wavelength basis for color rendering, where the $R()$ and $f_r()$ functions become vector-valued, with spectral components treated independently.Typically, this is done for three components corresponding to some version of red, green, and blue, but may be extended to several spectral dimensions to achieve more accurate color. For the purposes of this paper and without loss of generality, we restrict our discussion to the scalar form of these functions.We further ignore subsurface scattering (i.e., BSSRDFs) (Jensen et al., 2001), polarization, diffraction, and fluorescence. These effects are important for certain classes of materials, but we leave them as subjects for future work.

The main challenge for any data-driven BSDF method is Monte Carlo sampling. Both rejection sampling and BSDF weighted sampling are terribly inefficient for typical, peaked distributions. Researchers have consequently spent much effort developing more practical approaches. These techniques fall into two broad categories: partial model methods and table-based methods.

Table-based methods build a cumulative probability table for each incident (or exiting) direction and look up sample directions with stochastically distributed random variables (Matusik et al, 2003). Such methods are direct and efficient but memory-intensive even for isotropic BRDFs, and become quickly impractical for general(anisotropic) distributions. Storing a single 4-D BRDF with 1-degree angular resolution takes over a gigabyte of memory, and look-up tablesmore than double this requirement. Such methods also placeaburden on the BRDF measurement process, which mustcompletely cover all relevant directions.

Partial model methods attempt to separate each BRDF into a product of lower-dimensional functions that can be fit to measured data (McCool, 2001; Bilgili et al., 2011;Suykens et al., 2003). This approach stands somewhere between table-based methods and analytical BRDF models (Cook and Torrance, 1981; He et al.; 1991; War92,LaFortune et al, 1997; Edwards et al. 2006). Similar to most full BRDF models, partial models have a small memory footprint and can be readily sampled with a series of modest one dimensional (1-D) lookups (Lawrence et al.,2004). Although factorization can in principle be used to match any distribution, the modeler must choose an appropriate parameterization, and this changes from one class of materials to another. Partial modeling thus alleviates but does not eliminate the troubling restriction to a subclass of real-world materials. A partial model is still a model.

Since our goal is to represent arbitrary and therefore model-free BSDFs, we adopt a table-based approach. Our challenge is two-fold: reducing memory requirements during rendering, and filling in missing measurement data. Our methods for reducing memory requirements are explained in the following sections.

The second challenge is more difficult, as data filling implies the presence of some model or smoothness behavior, and for this we rely on some recent work in this area (Bonneel et. al., 2011).We present our measurement interpolation method towards the end of our paper.

3. Fixed and Variable Resolution Data Standards

Our BSDF data representation builds upon the eXtensible Markup Language (XML) standard created by Mitchell et al. to support LBNL's WINDOW 6 program, the industry standard for calculating radiative transfer through CFSs (Mitchellet al., 2008). The WINDOW 6 matrix representation was created to facilitate combining window elements as layers, which requires reflectance as well as transmittance data for each layer. We adopt the existing format and introduce our own adaptive density representation.

Most BSDF files contain extensive data values for front and back reflection as well as transmission. They might also contain information describing geometric detail for the system, as would be useful in the case of venetian blinds. Such detail permits a rendering program to rely on the measured BSDF for indirect light transfers, while using the geometry for direct views and computing visibility. To represent geometric detail, we interpolate the Materials and Geometry Format (MGF), a compact description designed for physically-based rendering (Ward, 2012). Use of this information is optional and facilitated by a standard parser that reduces MGF entities to a form that is easy for a rendering system to digest. We demonstrate how this data may be used in Section 4. (Examples of our XML file structure and MGF subformat are given in the supplemental material.)

Once we have a BSDF loaded, we have found the following four queries useful for rendering, independentof the underlying representation:

(a) Get a BSDF value for a pair of directions

(b) Get the directional hemispherical scattering for a given incident direction

(c) Get the projected solid angle sample size for one or two directions

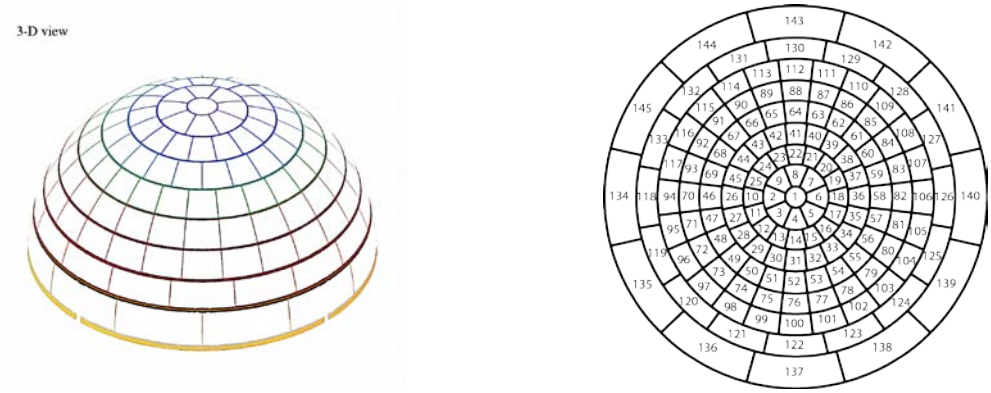
(d) Get a PDF-derived importance direction for the given incident direction

The first query (a) is a simple look-up into our stored BSDF. The second query (b) integrates all the requested BSDF components at the given incident vector, yielding the total energy transfer for that direction. The third query (c) is helpful for determining how closely to sample the BSDF in a particular region. Given two directions, it returns the min. and/or max. projected solid angle linking the two in our representation. Given only one direction, it returns the min. and/or max. sample size over all other directions (front and back). The final query (d) is the most crucial for Monte Carlo rendering, since it generates importance samples according to the probability distribution function. The following subsections describe our two representations and techniques for answering query (d).

3.1. Matrix BSDFs

The matrix BSDF type is thus named because it uses a fixednumber of incident and exiting directions to form a rectangular matrix of values. Thenumbers of incident and exiting directions are often equal, since square matrices can be more easily composed into a single matrix that includes the effects of interreflections between layers (Klems, 1994). For this reason also, the WINDOW 6 designers chose a mapping between matrix position and hemisphere direction that results in a diagonal matrix when transmission or reflection is purely specular. The Klems coordinate system, as it is called, forms the basis for mapping between directions and matrix entries.

The Klems hemispherical basis is shown in Figure 1(a). The patches have roughly equal projected solid angles. The patch indexing scheme for placing hemisphere values in a matrix row or column is shown in Figure 1(b) and the exact angle definitions are given in Table 1. The orientation of this hemisphere changes depending on whether the ray is incident or exiting on the front or back side in order tomaintain the aforementioned diagonal relationship in specular systems.



(a) (b)

Figure 1. (a) The Klems hemispherical coordinate system for matrix BSDFs. (b) The ordering of Klems patches in matrix rows or columns.

Average θ	Maximum θ	Number of $\phi's$
0	5	1
10	15	8
20	25	16
30	35	20
40	45	24
50	55	24
60	65	24
70	75	16
82.5	90	12

Table 1. Definition of Klems basis angles in degrees. The first ϕ value at each latitude is centered at0^o.

Using the Klems basis for our matrix BSDF, we have exactly 145×145 or 21025 values for each component. For a BRDF, we may only have a single reflection component, but for a transmissive system, we should have three components: transmission, front reflection and back reflection. Traditional WINDOW 6 files contain redundant front and back transmission for a total of four components. A matrix BSDF file is therefore in the range of 500 -1000 Kbytes uncompressed, and a loaded BSDF takes about 256 Kbytes of memory using 32-bit floats.

Answering our sampling query (d) for the Klems matrix representation is not particularly difficult:

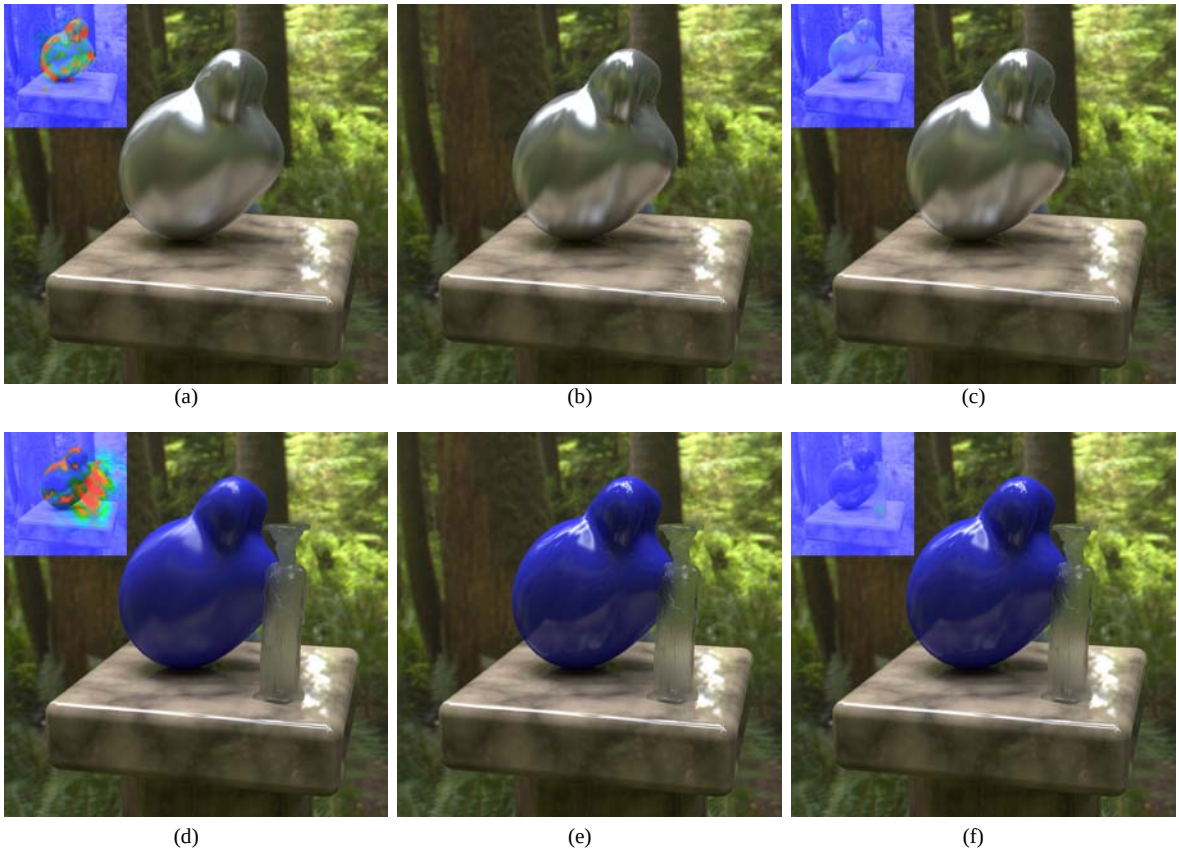
(1) For each new incident patch (row), we build a cumulative table from the exiting patches (columns) in Klems index order. Tables are cached and reused for the same incident patches.

(2) The final entry in this column table is the total directional hemispherical scattering for the given incident direction, and we multiply the input$[0, 1]$ random variable for the sample ray by this total.

(3) We look up the entry closest to this product in our table, and generate a ray in the direction corresponding to the associated patch.

Effectively, we are combining our two angular dimensions into a single 1-D cumulative table, similar to Monte Carlo inversion process. The net result is that smaller random variable inputs will tend to send rays out patches with smaller index numbers (near the normal) and larger random inputs sent out patches with larger indexes (near the horizon). More rays will of course be sent out patches with larger BSDF values, which is the essential character of importance sampling. Further randomization is performed to avoid sending all the rays for a particular patch in the same direction.

An example rendering of a matrix BSDF representation of a Ward-Geisler-Moroder-Dür model (Geisler-Moroder and Dür, 2010) is shownin Figure 2(a). The data-driven model shows the effects of matrix sampling, particularly in the highlights. Figure 2(d) shows a rendering of what should be a much smoother surface with sharper reflections. This demonstrates the main limitation of a matrix representation, which is resolution. Sampling a BSDF every 10or 15 degrees is inadequate for many rendering applications. Increasing the matrix resolution would help, but a uniform partition of the hemisphere at the resolution required to capture such peaks would take gigabytes of storage and be very inefficient. This is where we may prefer a more adaptive solution.



(a) (b) (c)

(d) (e) (f)

Figure 2. (a) Matrix-based BRDF rendering of anisotropic Ward model. The sharpness of the highlights is degraded by thelack of resolution in the 145×145 matrix representation. (b) Reference image. (c) Tensor tree BRDF rendering of the samemodel. (d) Matrix representation of isotropic Ward model on sculpture and statuette. (e) Reference image. (f) High-resolutiontensor-tree representation. Insets show color-coded maps which represent a probability that an average observer will notice adifference between the reference image and rendered image (Mantiuk et al., 2011).

3.2. Tensor Tree BSDFs

A tensor tree represents the sharp peaks in a BSDF by subdividing adaptively in different regions of the distribution. Sample density is coordinated between input and output directions, since higher density required in one implies that higher density is required in the other.

Consider a mirror-like surface. At an incident angle of $(\theta, \phi) = (45^o, 0^o)$(there will be a strong peak in the outgoingdirection near $(\theta, \phi) = (45^o, 180^o)$, which must be recordedat high density. We could subdivide the exiting hemisphereadaptively using a geodesic hierarchy or similar, but whatabout the incident hemisphere? We cannot partition it independently,because every incident direction will correspond to a peak *somewhere* in the outgoing hemisphere, implyingthat the same high resolution is needed *everywhere* for incidence.A low-resolution incident hemisphere of adaptivelypartitioned exiting hemispheres would have us jumping fromone narrow output direction to another, with predictable results.Reversing incident

and exiting hemispheres would nothelp – they must be subdivided together. The solution is tointerleave the input and output hemispheres and employ treesubdivision on the whole distribution.

Tensors are a direct extension of vectors and matrices. Avector is a rank-1 tensor, and a matrix is a rank-2 tensor. Arank-3 tensor is therefore a stack of matrices, and so on tohigher dimensions (Lebedev and Cloud, 2003). In our application, tensors providea way to keep closely associated BSDF values near eachother for convenient subdivision.

Sticking with dimensions we can visualize, an isotropic BSDF can be represented as a rank-3 tensor. Since the outgoing distribution does not change with rotation, we only need one matrix per incident theta, covering all the outgoing directions for that incidence. Columns and rows in each matrix correspond in this case to the two degrees of freedom on the exiting hemisphere. The actual dimensions of the matrix and even the number of theta values need not be decided in advance – this is where our tree structure takes over.

Figure 3 shows a familiar octree hierarchy, where each subdivided node contains eight children corresponding to the voxels of that subtree. If all the leaf voxels were the same size, our uniform three dimensional (3-D) grid would correspond to a cubic tensor. We could fill this tensor with a set of isotropic BSDF values. From this starting point, we can adaptively prune our tree. We collapse all siblingleaf voxels that are within some epsilon of each other into their parent, assigning an average value. Applying this recursively to the entire tree, we end up with a hierarchy that is dense where the BSDF changes rapidly and sparse where it changes slowly, similar to wavelet-based decompositions. The tree representation directly allows us to look up any value quickly and efficiently from the original BSDF. Indeed, the tree itself becomes a new aggregate data type that transcends the original tensor because it has no maximum density limit. A mirror would be represented by a high density track of large-valued voxels along a line through our cube, surrounded by more sparsely represented regions requiring little storage.

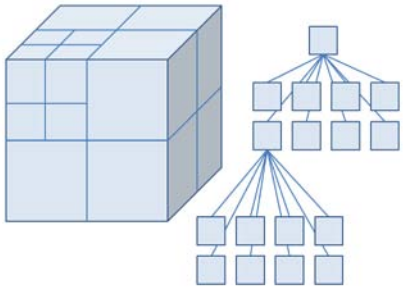


Figure 3. An octree hierarchy used to define a rank-3 tensor tree for an isotropic BSDF.

In the more general case of anisotropic BSDFs, our tensor is rank-4 and encoded in a hextree. The two dimensions of the exiting hemisphere are enveloped in two more dimensions for the incident hemisphere. Figure 2(c) shows our example scene with the BSDF data recorded as a rank-4 tensor tree. The maximum angular resolution in this rendering is less than 1^o, compared to 10^o in the original matrix representation. The total size of the tree is considerably larger due to its higher density, but it only takes 5% of the memory a fully populated tensor requires and $1/3$ the time to render. This particular BSDF loads into 30 Mbytes of memory. The differences are even more pronounced in the smooth, isotropic case shown in Figure 3(f). For the blue material, we resolved our tensor tree to an angle of 0.25^o. This surfacetook 170 Mbytes of RAM during rendering, and the roughertranslucent surface took 5 Mbytes.

We have yet to address the important question of how we map dimensions in our tensor tree to dimensions on each hemisphere. After all, one domain is square and the other is not. We need to preserve neighborhoods for stratified sampling, and it would be optimal if areas were maintained as well. Specifically, we would like each entry at a given level in our tree to represent the same projected solid angle on our hemisphere. Fortunately, just such a mapping was developed by Shirley and Chiu (Shirley and Chiu,1997) and is shown in Figure 4:

$$(x', y') = \begin{cases} \left(x\cos\left(\frac{\pi y}{4x}\right), x\sin\left(\frac{\pi y}{4x}\right)\right), & if\ |x| > |y| \\ \left(y\cos\left(\frac{\pi}{2} - \frac{\pi x}{4y}\right), y\sin\left(\frac{\pi}{2} - \frac{\pi x}{4y}\right)\right), & otherwise \end{cases} \quad (2)$$

where $(x, y)$and (x', y') are the two-dimensional (2-D) point coordinates on the square and the disk. The mapped regions are distorted between a square and a disk so that locality and area properties are preserved for stratified importance sampling.

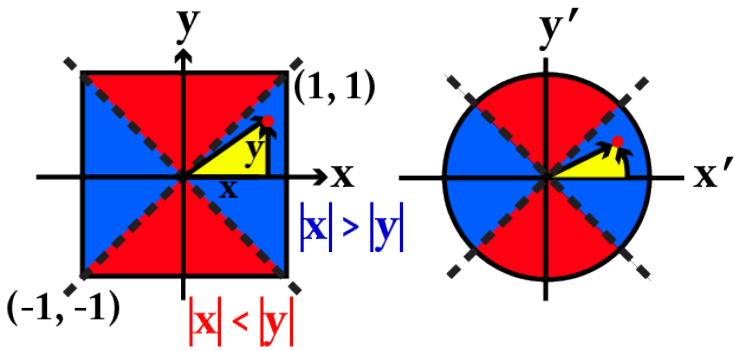


Figure 4. Shirley and Chiu's area-preserving map between a square and a disk (Shirley and Chiu, 1997).

Similar to the matrix representation described in the previous section, the only difficult query to implement for tensor trees is the ray sampling call (d). For stratified MC sampling, we need to sort our leaf nodes at a given incident directioninto a 1-D sequence that preserves locality. We employ theHilbert traversal shown in Figure 5, which maximizes locality while keeping a direct relationship to our octree/hextreebranching (Gotsman and Lindenbaum, 1996). Working in a square slice corresponding to the exiting hemisphere for the queried incident vector, we order our cumulative table along the Hilbert path. This path traverses the $[0,1]^2$ square with a $[0,1]$ line segment such that any given fraction of the line segment covers the same fraction of the area. This is the key to converting our 2-D sampling domain into a 1-D cumulative table based on projected area. Larger leaf nodes will skip further along the path, but we never have the problem of re-entering a node since the Hilbert curve respects the quadtree's recursive boundaries.

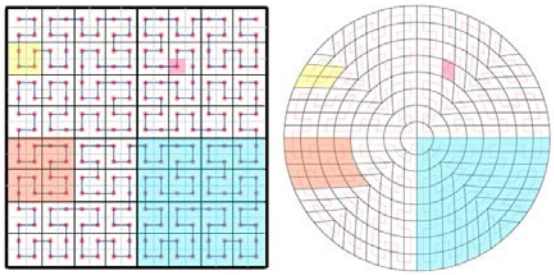


Figure 5. A Hilbert 2-D space-filling curve drawn to the 4thlevel. Different regions illustrate a hypothetical tensor tree's subdivision of exiting directions where samples might go.

Our stratified sampling method (d) for tensor trees is summarized as follows:

(1) We project our incident vector onto a disk by discarding the z0-coordinate and map this position to the unit square using Shirley and Chiu's formula.

(2) For a previously unvisited branch of our octree or hextree, we build a cumulative table along a level-16 Hilbert path in the associated exiting quadtree. Tables are cached and reused for the same incident branches.

(3) The final entry in this column table is the total directional hemispherical scattering for the given incident direction, and we multiply the input$[0,1]$ random variable for the sample ray by this total.

(4) We look up the Hilbert curve position corresponding to this product in our table, and this gives us a 2-D coordinate in our unit square. We further jitter the position to avoid using the leaf nodes' center every time.

(5) We map this position back to a unit disk using Shirley and Chiu's mapping, then to a unit vector by adding the appropriate z'-component (i.e., $z' = \pm\sqrt{1 - x'^2 + y'^2}$). This is our generated ray.

This is very similar to the method used for the matrix representation with the addition of a Shirley-Chiu mapping on either end. Some additional work rotating vectors is required for isotropic distributions, but this is straightforward. Overall, the only part that takes significant time is the building of the cumulative tables, which is why we cache and reuse these data. In a running program, our tables may use as much memory as the BSDF itself. Least recently used replacement could be applied if this ever becomes an issue.

The method (d) described above and the one given in Section 3.1 for matrix BSDFs are not themselvesstratified, but they preserve stratification in the controlling input variable. If the routine is given a sequence of uncorrelated control values, then the generated importance samples will be uncorrelated, also. However, inputs near each other in the 0-1 range produce rays that are near each other in the normalized samplingdomain. Similarly, well-separated inputs tend to generate well-separated ray samples. Stratifying the control input thus produces stratified output rays, which is an important technique for reducing variance in the result. Adaptive sampling also makes use of this property, but should be applied in stages to avoid bias (Kirk and Arvo, 1991).

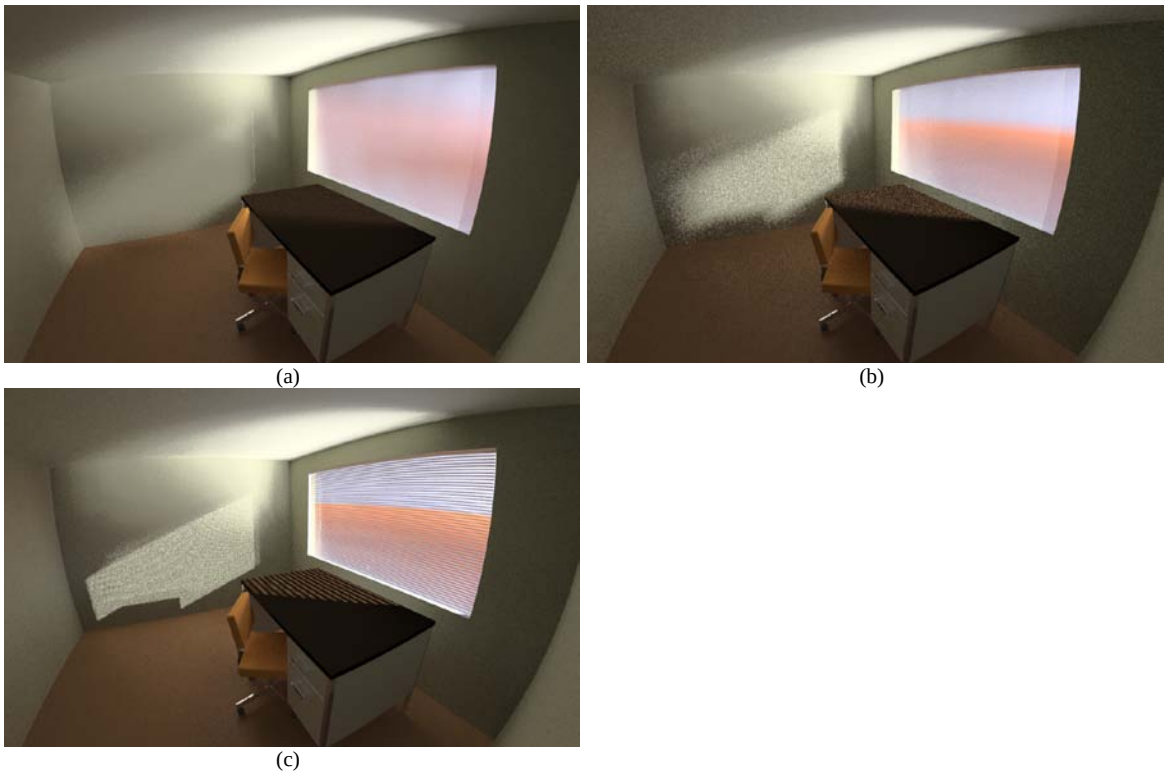
4. Rendering in Radiance

The *Radiance* lighting simulation system is used directly or indirectly by the majority of the daylight modeling community(Galasiu and Reinhart, 2008). Its freely available source facilitates additions and experimentation, and improves transparency for researchers and advanced users (Larson and Shakespeare, 1998). *Radiance* provides an excellent platform for testing our BSDF representation and library interface.

Although most BSDF research focuses on different aspects of reflection, we are particularly interested in transmission through complex fenestration systems due to their importance for daylighting applications.Such systems are often designed to have unusual behaviors when it comes to light transfer (Thanachareonkit and Scartezzini, 2010). These behaviors are not fit by existing models, and consequently have a greater need for data-driven methods. Since a CFS controls how light enters a space, accurate simulation is critical to predicting interior light levels and distributions. Because such systems tend to be found on or near windows, how they affect the view and to what degree they induce glare are also important concerns.

4.1. BSDF Proxy Method

One of the most challenging aspects of a CFS is rendering its appearance. While aggregate behavior as described by a BSDF is a useful way to model illumination, spatial variations determine what a window looks like to an occupant, and may impart a pattern to a solar beam component as well. Figure 6(a) shows a BSDF simulation of a venetian blind CFS with curved, half-specular louvers from an interior perspective using a Klems matrix representation. At 10^o resolution, direct pass-through cannot be represented accurately, casting light blobs rather than beams. Figure 6(b) shows a BSDF simulation which uses a tensor tree to achieve roughly 3^o resolution, and the sun patch begins to take shape, although the edges are still blurry. We also get the vague sense of a horizon out the window, although we cannot see the slats of the blinds. In fact, increasing BSDF resolution would never allow us to see the slats, because the BSDF represents only angular data, not spatial (X-Y) patterns. One solution is to introduce a bidirectional texture function (BTF) to the data (Dana et al. 2002). This would certainly solve the problem, but would take additional storage and put a burden on the measurement and rendering processes.



(a) (b)

(c)

Figure 6. (a) A venetian blind system rendered from a 145145 Klems matrix. (b) The same scene rendered using a tensortree representation with 3 the resolution of the Klems data. (c) The scene using a BSDF surface as a proxy for detailed blindsgeometry. We can now see the blinds in full detail as well as the striped shadows cast on the interior surfaces.

We take an alternate approach, where the BSDF is used as a *proxy* for detailed geometry. We use the geometry for direct views and shadow testing, and the measured BSDF to characterize light scattered by the system.

Figure 7 diagrams our proxy behavior during the ray tracing process. View rays pass through the two proxies and interact with the underlying geometry, as do rays sent to test for light sources and shadows. Indirect sampling rays interact only with the proxy surfaces, and transmitted rays are displaced to bypass the detailed geometry and compute contributions from the opposite side. (Indirect rays originating inside the sandwich are blind to the proxy geometry, and care is taken to avoid double-counting unscattered transmission from light sources). This is very similar in practice to the light source "imposter" method used by *Radiance* and other renderers (Ward, 1994;Shirley et al., 1996).

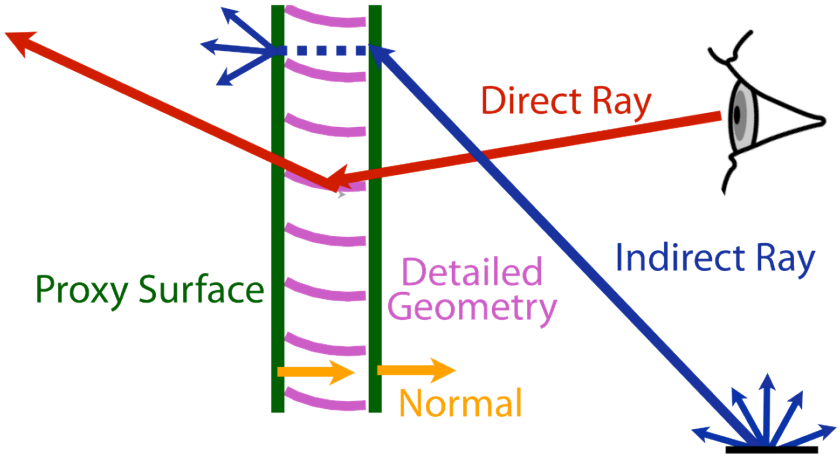


Figure 7. Proxy surfaces handle indirect contributions. Detailedgeometry is used for direct viewing and source testing.

The proxy approach proves an acceptable compromise for rendering, as shown in Figure 6(c). Indirect contributions are smoothly integrated because sample rays do not see the intensity variations of the blinds themselves, relying on the measured BSDF data that is spatially averaged. Achieving a similar level of quality without the benefit of a data-driven BSDF takes 5 times as long. (See the supplemental material.) This leverages the accuracy of our measurements while preserving the appearance of a full geometric model.

4.2. BSDF Simulation

It is often the case that a CFS or fabric has geometric detail whose scale is too large to be averaged by existing goniophotometers. The usual method for obtaining a BSDF for such compound surfaces is to individually measure the materials that make up the system, threads, diffuse paints, specular coatings or translucent materials, etc., and then create a geometric model that matches the compound surface as closely as possible (Westin, 1992;Zhao, 2011). The combined model is then handed to a ray tracing simulation, and its overall behavior is characterized.

This approach has a number of important advantages. Besides avoiding the need for a warehouse-sized goniophotometer, we can simulate scattering wherever we like on thecombined system, thus avoiding the interpolation requirement discussed in the following section. Using our virtual goniophotometer, we can precisely control the area of measurement and boundary conditions, which is very important for repeating systems with large elements. Moreover, we can test the quality of our representations by simulating reference systems at different resolutions, using both matrices and tensor trees, isotropic and anisotropic, and comparing simulations to originals, as we have done in this paper.

All of the synthetic BSDFs shown in this paper were computed by the *Radiance***genBSDF** program, which has been validated against commercial optical ray tracing software (McNeil et al. 2012). We have enhanced this program to produce our tensor tree data as well as the original Klems matrix data, taking fullest advantage of multicore processors. It can still take hours to characterize a CFS such as the tensor tree version of the half-specular blinds in Figure 6. This is nevertheless a savings over the time it would take for full gonio measurement of an anisotropic system, even if it were possible using existing devices.

5. Measuring and Interpolating BSDFs

Measuring BSDFs often results in unreliable, missing, or sparse data samples. This can be due to the goniophotometer blocking the light source in the normal front scattering condition, or high measurement noise for low BSDF values (Matusik et al, 2003). This problem is even more pronounced on anisotropic

BSDFs, where the full 4-D domain needs to be sampled, hence resulting in a *curse of dimensionality*. For instance, Matusik et al.'s (Matusik et al, 2003) anisotropic measured BRDF dataset is not suitable for direct rendering, or fitting to analytical models (red-velvet and purple satin in (Ngan et al., 2005)). In addition, we represent BSDFs with a particular sampling pattern, which can be different from the initial measurement locations. In such conditions, interpolation is necessary to fill in the gaps.

While linear interpolation can give plausible results when data points are sufficiently close, it can fail to capture large variations or follow highlight movement in the BSDF. We address this problem using a generalized 3-way displacement interpolation method akin to (Bonneel et. al., 2011).

5.1. Displacement interpolation for ghosting BSDF values

In (Bonneel et. al., 2011), a mass transport method is used to interpolate between two different probability distributions f_0 and f_1. An intermediate function f_α is found such that the energy used to move distribution f_0 toward f_1 is minimized, and so the transport is stopped at an intermediate time $0 \leq \alpha \leq 1$. This has successfully been used to interpolate between any two *different* BRDFs. However, we are interested in finding an interpolation *inside* a single given BSDF. While the former only requires two functions to interpolate from, the latter requires at least three functions as we will demonstrate.

Similarly to (Bonneel et. al., 2011), we consider the probability distribution defined on the sphere and parameterizedby the incident direction $p_{\vec{\omega}_i}(\vec{\omega}_o) = \frac{\log(1+f_r(\vec{\omega}_i;\vec{\omega}_o))}{\int_{\Omega_o}\log(1+f_r(\vec{\omega}_i;\vec{\omega}_o))d\vec{\omega}_o}$. In the tensor based representation, both $\vec{\omega}_i$and $\vec{\omega}_o$ lie on a particular grid. Hence, we are looking for $p_{\vec{\omega}_i}(\vec{\omega}_o)$ given $k \times l$ measured values$p_{\vec{\omega}_i^k}(\vec{\omega}_o^l)$. The continuous Radial Basis Function (RBF) gaussian-based representation naturally allows to obtain the valueof $p_{\vec{\omega}_i^k}(\vec{\omega}_o)$ for arbitrary $\vec{\omega}_o$. The remaining challenge is to retrieve the probability distribution $p_{\vec{\omega}_i}$ given fixed distributions$p_{\vec{\omega}_i^k}$. This challenge is addressed by considering a displacement interpolation between three different $p_{\vec{\omega}_i^k}$distributions, located at the vertices of a spherical Delaunay triangulation of the directions$\vec{\omega}_i^k$.

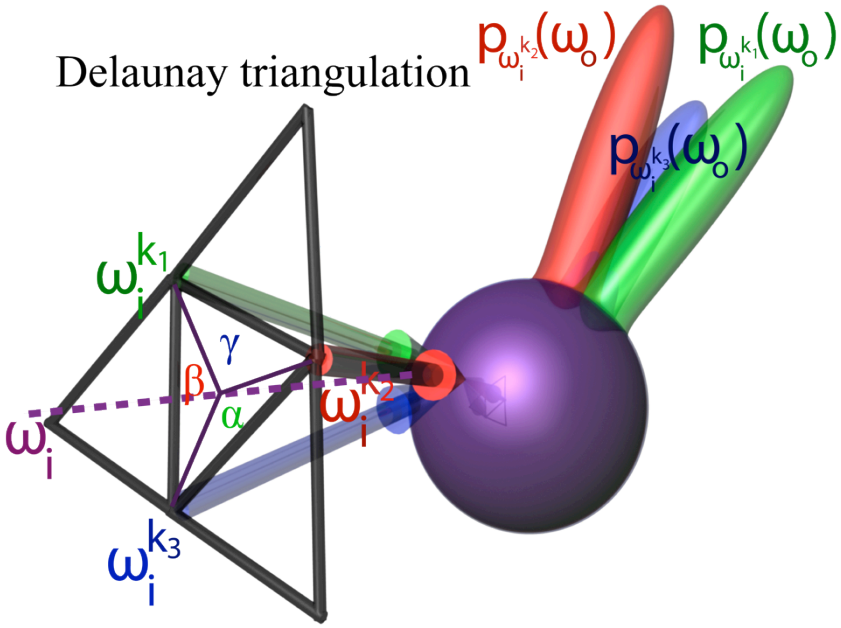


Figure 8. Interpolating between 3 probability distributions, defined at the vertices of a Delaunay triangulation.

The set of measured incoming directions $\vec{\omega}_i^k$ is first triangulated using a spherical Delaunay triangulation (see Figure 8). For each incoming vector $\vec{\omega}_i$ of the tensor grid, we compute its barycentric coordinates α, β, γ with respect to its surrounding triangle with vertices $\vec{\omega}_i^{k_1}, \vec{\omega}_i^{k_2}, \vec{\omega}_i^{k_3}$. Using this triangle, a three-way displacement interpolation could be solved with a third order unknown tensor (Sharma 1977). However, solving for and storing such a dense tensor would be prohibitive, as the memory usage would grow as $O(n^3)$ which is intractable in typical conditions where $n \approx 1000$ gaussian RBFs. We instead developed a two-stage, two-way approximation. The barycentric coefficients are used to interpolate between the probability distributions by applying pairwise displacement interpolations. Specifically, we first decompose $p_{\vec{\omega}_i^{k_1}}, p_{\vec{\omega}_i^{k_2}}$ and $p_{\vec{\omega}_i^{k_3}}$ in a set of positive Gaussian RBFs. We then advect the gaussians using the mass transport formulation in a pairwise fashion: first advecting $p_{\vec{\omega}_i^{k_1}}$ toward $p_{\vec{\omega}_i^{k_2}}$ at a time$t = 1 - \frac{\alpha}{\alpha+\beta}$, then advecting the resulting gaussians toward $p_{\vec{\omega}_i^{k_3}}$ at a time$t' = \gamma$. This scheme is not order-independant since the space of these probability distributions has non-zero curvature (Lott, 2008). However, we found no visual differences permuting the indices of the triangle. The gaussians are finally summed up to reconstruct the interpolated function $p_{\vec{\omega}_i}$, evaluated at $\vec{\omega}_o$. An interpolation example is shown in Figure 9.

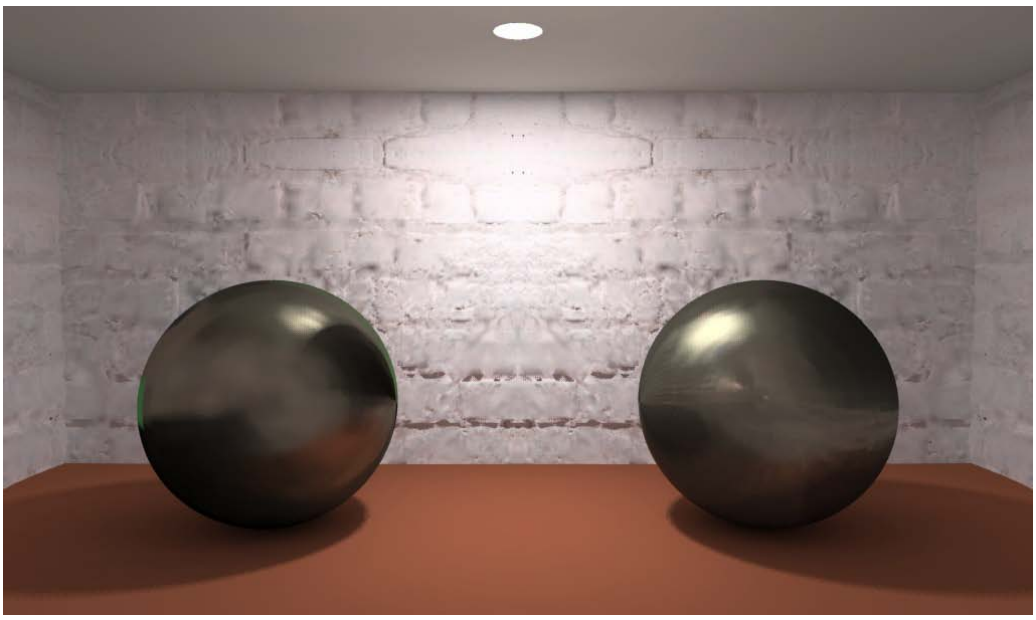


Figure 9. The measured anisotropic brushed-aluminum BRDF (Matusik et al, 2003) (left) is not suitable for direct renderingdue to missing data (in black) and measurement noise (greenhalo). Displacement interpolation fills-in the missing parts and smoothes the measurement noise (right).

6. Discussion and FutureWork

Figure 10 demonstrates many of the features of our BSDF representation and rendering method. The scene contains isotropic and anisotropic data interpolated from measurements as well as BSDFs derived from simulation. The bronze head, flowerpot and silver goblet materials were all converted from MERL isotropic measurements (Matusik et al, 2003). Our XML files for recording these BRDFs were on average $1/10th$ the size of the original binary data. The aluminum mural and velvet tablecloth were interpolated from MERL anisotropic data (Matusik et al, 2003). The sheer curtain was simulated using genBSDF on a loose-weavefabric model. The complex fenestration consisting of the window, frame, and vertical blinds was modeled as a unit and a proxy surface was created just in front of it, which substantially lowers noise during the interreflection calculation without affecting the view. Less than 500 Mbytes of RAM were needed during rendering, which took 52 hours on a 12-core machine.

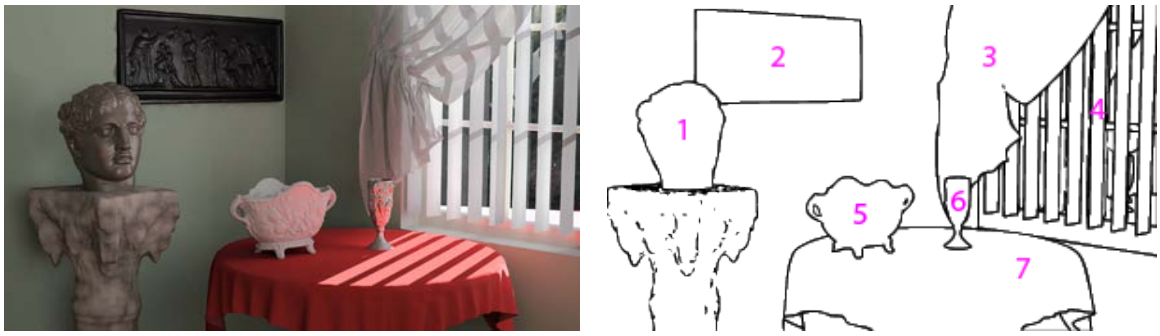


Figure 10. An image rendered using multiple data-driven BSDFs. The head model (1) uses the alum-bronze material from the MERL dataset (Matusik et al, 2003). Similarly, the mural (2) uses the anisotropic brushed-aluminum material from MERL. The sheer curtain (3) uses a BSDF simulated from a detailed fabric model. The window itself (4) has a proxy BSDF (also simulated) that efficiently redirects light from the outside without affecting the appearance of the window itself. The flowerpot (5) uses the MERL alumina-oxide data, the goblet (6) uses the silver-metallic-paint material, and the tablecloth (7) uses the red-velvet anisotropic measurements (Matusik et al, 2003).

Our efforts have focused on two BSDF representations, the Klems matrix and the tensor tree. The Klems matrix is convenient for combining window layers and performing annual simulations, while the tensor tree provides adaptable density for highly peaked distributions. Other representations could provide benefits beyond these, perhaps using wavelets or spherical harmonics (Claustres et. al, 2002;Sillion 1991). Our goal was not to find the ultimate BSDF representation, but to provide a framework with basic functionality that insulates our rendering application from the implementation details. Nevertheless, we get plausible results when representing MERL anisotropic data (Matusik et al, 2003) which are neither proper for direct rendering nor analytic modeling (Ngan et al., 2005). Improvements in storage and computational efficiency will always be welcome, and we hope to incorporate alternative representations in the future.

There are presently limitations to the use of proxy geometry as we have defined it. The main assumption is that the BSDF geometry fits neatly between two planar surfaces, and light leakage from the sides is negligible or blocked by a window frame or an equivalent surround. If the data-driven surface is reflecting only, then the detail geometry must live behind a single, planar surface. It may be possible to extend the proxy concept to substantially non-planar situations, but this would require a more sophisticated, parametric representation of a geometric texture. We consider this an interesting area for future work.

An important enhancement we plan to make in the near future is the addition of spectral data. The API was designed with this in mind, and our library can associate either colors or spectra with individual BSDF components. Unfortunately, the current XML files provide only CIE Y channel and thermal data. Adding other spectral sensitivities to the format and to our physical measurement devices is not particularly difficult. The simplest method is to employ source and sensor-corrected, colorimetric X, Y and Z filters in the goniophotometer, whose output is convertible to any defined RGB space. Further in the future, we may wish to add bandpass filters for multispectral measurements. In some cases, it will be more efficient to represent the specular peak with fewer color channels than the rest of the BSDF. For example, our API could support a monochrome tensor tree for specular peaks combined with a seven spectral component Klems matrix component and a 5-nm wavelength increment Lambertian component. The XML specification would have to be extended somewhat, but a compliant application would simply link to the new library to benefit from these additions.

One of our principal goals is to foster sharing of BSDF data. By providing a standard format and library, we have taken the first steps, but there is more we can do. We intend to set up a wiki site for BSDF data, where researchers and practitioners can upload and download data with full descriptions. To seed this site, we plan to convert the MERL dataset to our format with the authors' permission (Matusik et al, 2003). Independently, the Lawrence Berkeley National Laboratory continues to add to their Complex Glazing Data Base, which uses the matrix version of the XML standard (Optics, 2012). Other researchers have previously shared their measurements (Marschner et al., 1999), and we look forward to assisting anyone who wishes to contribute data to the repository.

Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, State and Community Programs, Office of Building Research and Standards of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 and by the California Energy Commission through its Public Interest Energy Research (PIER) Program on behalf of the citizens of California. This work was also supported by NSF grant IIS 1110955.

References

Ashikhmin M., Shirley P.: An anisotropic Phong BRDF model. Journal Graphics Tools 5, 2 (2000), 25–32.

Bilgili A., Öztürk A., Kurt M.: A general brdf representation based on tensor decomposition. Computer Graphics Forum 30, 8 (2011), 2427–2439.

Bonneel N., Van De Panne M., Paris S., Heidrich W.: Displacement interpolation using lagrangian mass transport. ACM TOG 30, 6 (2011), 158:1–158:12. (Proc. SIGGRAPH Asia ’11).

Claustres L., Boucher Y., Paulin M.: Spectral brdf modeling using wavelets. In Proc. of SPIE (2002), SPIE, pp. 33–43.

Cook R. L., Torrance K. E.: A reflectance model for computer graphics. Computer Graphics 15, 3 (1981), 307–316. (Proc. SIGGRAPH ’81).

Dana K. J., Van Ginneken B., Nayar S. K., Koenderink J. J.: Reflectance and texture of real-world surfaces. ACM TOG 18, 1 (Jan. 1999), 1–34.

Edwards D., Boulos S., Johnson J., Shirley P., Ashikhmin M., Stark M., Wyman C.: The halfway vector disk for brdf modeling. ACM TOG 25, 1 (2006), 1–18.

Gotsman C., Lindenbaum M.: On the metric properties of discrete space-filling curves. IEEE Transactions on Images Processing 5, 5 (1996), 794–797.

Geisler-Moroder D., Dür A.: A new ward brdf model with bounded albedo. Computer Graphics Forum 29, 4 (2010), 1391–1398. (Proc. Eurographics Symp. Rendering).

Galasiu A., Reinhart C.: Current daylighting design practice: A survey. Building Research & Information 36, 2 (2008), 159–174.

Gayeski N., Stokes E., Andersen M.: Using digital cameras as quasi-spectral radiometers to study complex fenestration systems. Lighting Research and Technology 41, 1 (2009), 7–25.

He X. D., Torrance K. E., Sillion F. X., Greenberg D. P.: A comprehensive physical model for light reflection. Computer Graphics 25, 4 (1991), 175–186. (Proc. SIGGRAPH ’91).

Jensen H. W., Marschner S. R., Levoy M., Hanrahan P.: A practical model for subsurface light transport. In Proc. SIGGRAPH ’01 (2001), ACM, pp. 511–518.

Kirk D., Arvo J.: Unbiased sampling techniques for image synthesis. Computer Graphics 25, 4 (July 1991), 153–156.

Kajiya J. T.: The rendering equation. Computer Graphics 20, 4 (1986), 143–150. (Proc. SIGGRAPH ’86).

Klems J.: A new method for predicting the solar heat gain of complex fenestration systems. ASHRAE Transactions 100, 1 (1994), 1065–1086.

Lebedev L., Cloud M.: Tensor analysis. World Scientific Pub., 2003.

LaFortune E. P., Foo S.-C., Torrance K. E., Greenberg D. P.: Non-linear approximation of reflectance functions. In Proc. SIGGRAPH ’97 (1997), pp. 117–126.

Lott J.: Some geometric calculations on Wasserstein space. Commun. Math. Phys. 277, 2 (2008), 423–437.

Lawrence J., Rusinkiewicz S., Ramamoorthi R.: Efficient BRDF importance sampling using a factored representation. ACM TOG 23, 3 (2004), 496–505. (Proc. SIGGRAPH ’04).

Larson G. W., Shakespeare R.: Rendering with Radiance: the art and science of lighting visualization. Morgan Kaufmann Publishers Inc., 1998. 2,

McCool M. D., Ang J., Ahmad A.: Homomorphic factorization of BRDFs for high-performance rendering. In Proc. SIGGRAPH ’01 (2001), ACM, pp. 171–178.

McNeil A., Jonsson C., Applefeld E., Ward G., Lee E.: A validation of a ray-tracing tool used to generate bi-directional scattering distribution functions for complex fenestration systems. Submitted to Energy and Buildings (2012).

Mitchell R., Kohler C., Klems J., Rubin M., Arasteh D., Huizenga C., Yu T., Curcija D.: Window 6.2/ Therm 6.2 Research Version User Manual. Tech. Rep. LBNL- 941, Lawrence Berkeley National Laboratory, 2008.

Mantiuk R., Kim K. J., Rempel A. G., Heidrich W.: HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. ACM TOG 30, 4 (2011), 40:1–40:14. (Proc. SIGGRAPH ’11).

Matusik W., Pfister H., Brand M., Mcmillan L.: A data-driven reflectance model. ACM TOG 22, 3 (2003), 759–769. (Proc. SIGGRAPH ’03).

Marschner S. R., Westin S. H., Lafortune E. P., Torrance K. E., Greenberg D. P.: Image-based brdf measurement including human skin. In Proc. of the EG Workshop on Rendering (1999), pp. 131–144.

Ngan A., Durand F., Matusik W.: Experimental analysis of brdf models. In Proc. of Eurographics Symposium on Rendering (2005), pp. 117–126.

Optics W.: The complex glazing data base, April 2012. http://windowoptics.lbl.gov/data/cgdb.

Sillion F. X., Arvo J. R., Westin S. H., Greenberg D. P.: A global illumination solution for general reflectance distributions. Computer Graphics 25, 4 (July 1991), 187–196.

Shirley P., Chiu K.: A low distortion map between disk and square. Journal Graphics Tools 2, 3 (1997), 45–52.

Sharma J. K.: Extensions and special cases of transportation problem: A survey. Digital library of India (1977), 928–940.

Suykens F., Vom Berge K., Lagae A., Dutré P.: Interactive rendering with bidirectional texture functions. Computer Graphics Forum 22, 3 (2003), 463–472.

Shirley P., Wang C., Zimmerman K.: Monte carlo techniques for direct lighting calculations. ACM TOG 15, 1 (Jan. 1996), 1–36.

Thanachareonkit A., Scartezzini J.-L.: Modelling complex fenestration systems using physical and virtual models. Solar Energy 84 (2010), 563–586.

Ward G. J.: Measuring and modeling anisotropic reflection. Computer Graphics 26, 2 (1992), 265–272. (Proc. SIGGRAPH ’92).

Ward G. J.: The radiance lighting simulation and rendering system. In Proc. SIGGRAPH ’94 (1994), ACM, pp. 459– 472.

Ward G.: The materials and geometry format, April 2012. http://radsite.lbl.gov/mgf/mgfdoc.pdf.

Westin S. H., Arvo J. R., Torrance K. E.: Predicting reflectance functions from complex surfaces. Computer Graphics 26, 2 (July 1992), 255–264.

Yu C., Seo Y., Lee S. W.: Global optimization for estimating a brdf with multiple specular lobes. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010), IEEE Computer Society, pp. 319–326.

Zhao S., Jakob W., Marschner S., Bala K.: Building volumetric appearance models of fabric using micro ct imaging. ACM TOG 30, 4 (Aug. 2011), 44:1–44:10. (Proc. SIGGRAPH ’11).

Supplemental material

1.The XML Format

A simple BSDF example of our XML format is provided in Figure 11. Although this is a legal BSDF file, it is not particularly interesting in the sense that it contains only a single reflection value, describing a perfect 100% diffuser. The detail geometry of a venetian blind is presented in Figure 12. This information is interpolated in an XML file that would also contain data for the BSDF reflection and transmission values for thousands of angles. These data might be measured or calculated, or some combination of the two. The complete definition of this XML format as well as the MGF description of geometry and materials will be provided on our website at the time of publication.

```
<?xml version="1.0" encoding="UTF-8"?>
<WindowElement xmlns="http://windows.lbl.gov"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://windows.lbl.gov/BSDF-v1.4.xsd">
<Optical>
<Layer>
  <Material>
      <Name>Perfect Diffuser</Name>
      <Manufacturer>ACME Surfaces</Manufacturer>
      <Thickness unit="Meter">0.150</Thickness>
      <Width unit="Meter">1.000</Width>
      <Height unit="Meter">1.000</Height>
  </Material>
  <DataDefinition>
      <IncidentDataStructure>TensorTree3</IncidentDataStructure>
  </DataDefinition>
  <WavelengthData>
      <LayerNumber>System</LayerNumber>
      <Wavelength unit="Integral">Visible</Wavelength>
      <SourceSpectrum>CIE Illuminant D65 1nm.ssp</SourceSpectrum>
      <DetectorSpectrum>ASTM E308 1931 Y.dsp</DetectorSpectrum>
      <WavelengthDataBlock>
          <WavelengthDataDirection>Reflection Back</WavelengthDataDirection>
          <AngleBasis>LBNL/Shirley-Chiu</AngleBasis>
          <ScatteringDataType>BRDF</ScatteringDataType>
          <ScatteringData>{ 0.318309886 }</ScatteringData>
      </WavelengthDataBlock>
  </WavelengthData>
</Layer>
</Optical>
</WindowElement>
```

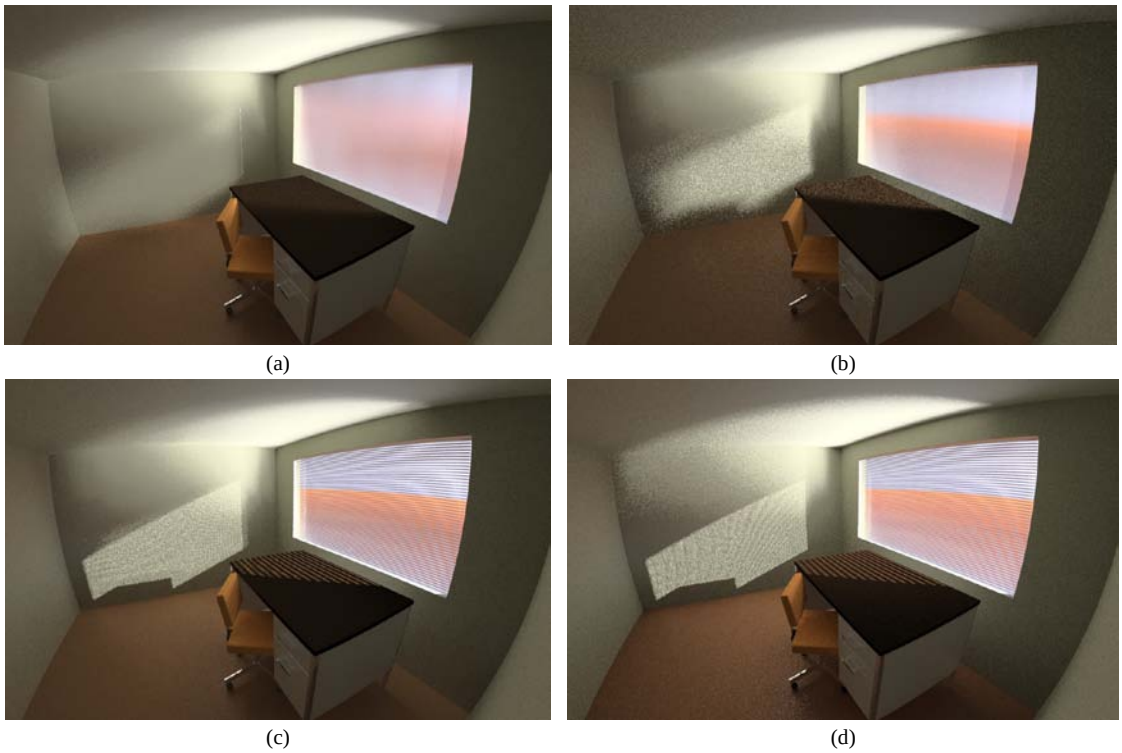
Figure 11. Simple but complete BSDF example XML file.

```
<Geometry format="MGF" unit="Meter">
# Y—axis points "up",
# Z—axis into room,
# right—handed coordinates
m WhitePlastic =
        rd .7
        rs .02  0
        sides  2
o VenetianBlinds
xf —rx —60 —a 67 —t 0 .03 0
      o Slat
       v v1 =
              p —2 0 0
       v v2 =
              p 2 0 0
       v v3 =
              p 2 0  .04
       v v4 =
              p —2 0 .04
       f v1 v2 v3 v4
      o
xf
o
</Geometry>
```

Figure 12. Inline MGF description of venetian blinds with 67 white slats.

2. BSDF Proxy Method

In Figure 13, we show one of the advantages of our data-driven BSDF representation, where it is used as a proxy for detailed geometry.



(a) (b)

(c) (d)

Figure 13. (a) A venetian blind system rendered from a 145 × 145 Klems matrix. (b) The same scene rendered using a tensor tree representation with 3 × the resolution of the Klems data. (c) The scene using a BSDF surface as a proxy for detailed blinds geometry.

We can now see the blinds in full detail as well as the striped shadows cast on the interior surfaces. (d) The same scene rendered without the benefit of a data-driven BSDF takes 5 × as long.

In Figure 13(c), our data-driven BSDF model is used as proxy for detailed blinds geometry. This provides 5 × speedup when compared to Figure 13(d) where a similar quality scene rendered without using our data-driven BSDF representation.

3. Annual Simulations

An important goal of the BSDF library is to insulate the caller from the underlying data representation, but there are times when the application needs more intimate access. One such case is the three-phase formulation of the daylight coefficient method for annual simulation. Briefly, this method breaks the daylight simulation problem down into three flux transfer problems:

(1) Light transfer from the sky (rather sections of the sky) to the building exterior.

(2) Light transfer through windows and skylights (e.g., via a CFS).

(3) Light transfer from inside window/skylight surfaces to the interior.

Each of these transfers is represented by a matrix. The exterior matrix is specified as $\mathbf{D}_w$ for "daylight." The window matrix is $\mathbf{T}_w$ for "transmission," and the interior matrix is $\mathbf{V}_w$ for "view." They each have a suffix because they generally apply to one window at a time, and the contributions from all windows must be summed together in the end. In the simple case where only a single window is present, we can express a vector of interior measurements i as:

$$\mathbf{i} = \mathbf{VTDs}. \tag{3}$$

The input vector s is the set of sky patch values for a particular time of day and year at the location being simulated. $\mathbf{D}$ can be thought of as the matrix of form factors between sky directions and window insolation directions, which includes the effects of external geometry. The $\mathbf{V}$ matrix contains integration factors between the light entering from the window in different directions and the desired set of simulation values, including the effects of interior surface interactions. The result vector $\mathbf{i}$ can be almost anything, depending on the computation of V. It might be a set of illuminance values, or pixel colors in an image. The beauty of this equation is that once $\mathbf{V}$ and $\mathbf{D}$ are determined using a ray tracing or radiosity method, calculating new $\mathbf{i}$ vectors for different sky conditions s is a quick matrix multiplication. If the transmission matrix $\mathbf{T}$ is fixed, then $\mathbf{VTD}$ may be combined into a single rectangular matrix to simplify this further. However, leaving it in the original form permits us to swap out $\mathbf{T}$ matrices to simulate different control settings for an operable CFS such as a venetian blind (e.g., slat angle).

The $\mathbf{T}$ matrix is of course our fixed resolution BSDF representation, which is critical to making this all work. A variable resolution BSDF does not permit a matrix formulation. This also argues for providing an application access to the underlying representation, which we do through the use of implementation headers that extend our API.

Figure 14(a) illustrates the exterior daylight matrix $\mathbf{D}$ that links sky patches to incident directions on an exterior window. We have assigned random colors to the sky regions, which may interact with the ground plane and other building surfaces on their way to the incident window directions corresponding to the Klems basis of the BSDF matrix, which we have outlined and superimposed on a hemispherical view.

Similarly, Figure 14(b) illustrates the interior view matrix $\mathbf{V}$, used in this case to generate a fisheye image from the back of the space. Colored patches corresponding to Klems window directions are projected onto the walls and reflected indirectly to other surfaces making up this view. Sampling in this figure and the

previous one was turned off to make the patch boundaries clear, but of course these directions are better-mixed in our actual calculation of $\mathbf{V}$.

(a) (b) (c)

Figure 14. (a) View from exterior window circled on the left, showing Klems basis (outlines) and sky subdivisions (colored patches) on the right. The daylight matrix$\mathbf{D}$ corresponds to these patch coefficients. (b) View matrix used to generate an image based on the output of our selected window. Direction patches are colored randomly to show their contribution to this view. (c) A single timestep out of thousands of images computed for the solar year. This represents 3pm on a clear solar equinox withvenetian blinds having curved, half-specular louvers set to horizontal.

Once we have computed $\mathbf{D}$ and $\mathbf{V}$, we can select the window behavior encoded in the BSDF $\mathbf{T}$ and multiply it all together using Eq. (3). This allows us to generate a sequence of images corresponding to the movement of light throughout the day and year, and each time step takes a few seconds compute (following a few hours of precalculation). An example result is shown in Figure 14(c). We have sped this up even further by combining all the sky vectors into an annual matrix and using a GPU to compute 8760 images for the whole year in the time it takes to write the information to disk.

4. BSDF Software Library

We have implemented the following queries for matrix and tensor tree sampling in an ANSI-C library with a fixed API:

(a) Get a BSDF value for a pair of directions

(b) Get the directional hemispherical scattering for a given incident direction

(c) Get the projected solid angle sample size for one or two directions

(d) Get a PDF-derived importance direction for the given incident direction

In the interests of space, we glossed over some important details about how BSDF data is made available to the application. The caller may in practice treat the BSDF as a single entity, or may access it as components. These components include transmission, front reflection and back reflection, each further divided into Lambertian and non-Lambertian components, any of which may be zero. The library further supports multiple non-Lambertian components, which may provide for more efficient spectral representations in the future (e.g., uncolored superficial reflection and colored body reflection).

In the simplest use of our BSDF library, the application makes a single call at each surface evaluation to generate a sample ray direction and weight using method (d). The luminance (Y) channel of the weight would always be the same, although the spectrum might change for different importance samples. Multiple rays could be generated to reduce variance, and their weights adjusted by $1/N$. The rays would then be evaluated in the usual recursive fashion.

One refinement to this simple method that greatly reduces noise is to sample light sources independently, evaluating both Lambertian and non-Lambertian components taken from the BSDF. Method (a) would be called to evaluate the BSDF in the known light source directions during shadow testing. The light sources would have to be ignored in subsequent sampling with (d) to avoid over-counting their contributions.

As well as treating light sources independently, *Radiance* relies on a separate calculation of diffuse interreflection. We extract this component from our importance sampling, and break our calls down further by non-Lambertian component. This ensures that every shading evaluation generates at leastone sample ray per component, fitting with the behavior of other material types in *Radiance*. Testing the design of our BSDF library in a practical rendering context led to some important refinements to the API as well as improvements in our implementation, as one would expect.

New representations as well as extensions to existing types may be added in the future without altering the interface, which is more general than the existing XML specification (figure 13). The library supports loading and caching BSDF data and simple vector operations for surface reorientation.

We employ a plug-in interface for each BSDF representation, using a callback structure with an associated set of required methods. The client data specific to each loaded BSDF is defined in a separate header file for that type. Applications that need access to the underlying data simply include the BSDF type's header, applying the accessors defined therein. The solution is neat and efficient, although changing existing types may require updates to type-specific calls.

5. Fitting Results

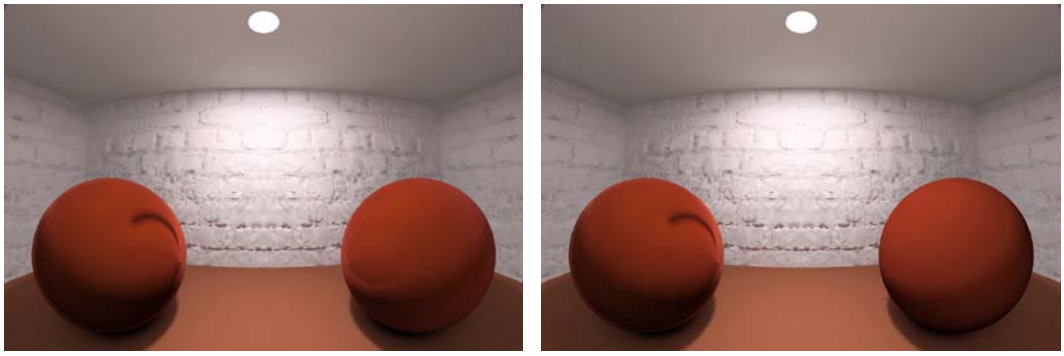
Portions of Matusik et al.'s (Matusik et al, 2003) anisotropic measured BRDF dataset is not suitable for direct rendering or fitting to analytical models due to unreliable, missing and noisy measurements (Ngan et al., 2005).We address this problem using a generalized 3-way displacement interpolation method akin to (Bonneel et. al., 2011).

As seen in Figure 15(a), our interpolation method reproduces the anisotropic purple satin measurements accurately with a rank-4 tensor tree. Both highlight regions and grazing angles match the original surface. In contrast, the Ward anisotropic BRDF model (Ward, 1992) has difficulty fitting this BRDF, as shown in Figure 15(b). (To estimate the parameters of Ward model, we use Ngan et al.'s (Ngan et al., 2005) fitting procedure.)

Similarly, Figure 16(a) shows our displacement interpolation method applied to Matusik et al.'s anisotropicdata for red velvet, which fills in missing regions in the measurements (e.g., black streak). Again, the Ward

(a) (b)

Figure 15. (a) The measured anisotropic purple satin BRDF (Matusik et al, 2003) (left) is interpolated into a rank-4 tensor tree and rendered using our method (right). (b) The Ward BRDF model (right) does not fit the data nearly as well.



(a) (b)

Figure 16. (a) The measured anisotropic red velvet BRDF (Matusik et al, 2003) (left) is not suitable for direct rendering due to missing data (black streak). Displacement interpolation fills in the missing data and smoothes the measurement noise (right). (b) The Ward BRDF model (right) does not represent the measured anisotropic red velvet as well, reverting to Lambertian reflection in this case.

BRDF model in Figure 16(b) cannot represent the measured data very well, and the closest fit is purely Lambertian.