Computing and applying variable-resolution data for bidirectional scattering distribution functions

G. Ward

Anyhere Software, 950 Creston Road, Berkeley, CA, 94708 USA

Abstract

We introduce the *Tensor Tree* representation for bidirectional scattering distribution functions (BSDFs), a variable-resolution data structure designed to minimize storage and computation. We offer a method for reducing full-resolution BSDF data into this format, and relate techniques for the efficient generation of uniformly weighted Monte Carlo samples. The Tensor Tree BSDF representation has been tested and incorporated into the 4.1 release of LBNL's *Radiance* lighting simulation and rendering system, with additional methods for modeling the appearance of complex fenestration systems. Finally, an independent software library is described for third-party developers who wish to support BSDF data outside of *Radiance*, and recommendations are made for future development directions.

Keywords:

1. Background

The bidirectional scattering distribution function (BSDF) describes how incident light reflects off or transmits through a surface. Mathematically, a BSDF is a *probability density function* (PDF) describing the likelihood that a photon incident from one direction will scatter in another direction. This function, usually denoted as $f_r()$, plays a central role in the integral equation that relates exiting radiance to incident radiance:

$$L_e\left(\theta_e,\phi_e\right) = \iint L_i\left(\theta_i,\phi_i\right) f_r\left(\theta_i,\phi_i;\theta_e,\phi_e\right) \left|\cos\theta_i\right| \sin\theta_i \, d\theta_i \, d\phi_i \tag{1}$$

The semicolon between incident and exiting directions indicates that incident and exiting directions may be swapped without affecting the result. That is to say, the BSDF obeys *reciprocity* with respect to energy direction. Also, like any PDF, the value of $f_r()$ is unbounded but must be positive everywhere and integrate to 1.0 or less over the exiting sphere to maintain energy balance for any given incident direction.

For a perfectly Lambertian surface with reflectance ρ, $f_r()$ will equal ρ/π for all directions on the same side as the incident light, and zero everywhere on the reverse side. For a pane of polished glass, $f_r()$ will be a Dirac δ function (i.e., a singularity) in the transmitted and reflected directions, and zero everywhere else. Although the BSDF may take on a theoretically infinite value in such cases, it will always integrate to something finite, and in practice, we can never measure the BSDF exactly when it is perfectly specular.

The BSDF as we have defined it is a four-dimensional scalar function. In fact, $f_r()$ may also be a function of wavelength, polarization, and a number of other variables we are choosing to ignore for the sake of simplifying our treatment. In our case, we are not measuring $f_r()$ as a function of these other dimensions, but they should be kept in mind as future areas to explore.

One pair of dimensions we are ignoring is the translational position in the plane of the surface. The "*r*" in $f_r()$ actually stands for "relocatable," and assumes a certain degree of constancy for the BSDF with respect to surface position. This was deemed a useful approximation, since we are averaging subsurface scattering effects as if all interactions happen at the outer surface boundary, when in fact they do not [Nicodemus77]. This is even more significant for transmission, and in effect says we are interested only in the aggregate behavior of a transmitting system, not the precise path photons follow to penetrate the 3-dimensional medium involved.

It is precisely the relocatable aspect of the BSDF that makes measurement problematic for most complex fenestration systems (CFS). Averaging a large enough sample area that the behavior of a louvered or otherwise periodic system is uniform, and putting the photometer and light source far enough away that the collimation is well controlled over such an area, are difficult challenges. A more common approach is to carefully measure the materials that make up a CFS, and separately measure the geometric configuration. From these, a model is constructed and ray-tracing may be used to simulate the system.

This two-step process for modeling CFS has a few important advantages. First, any size and shape system may be characterized, provided there is some way to define the input and output apertures. Second, the periodicity can be matched in the simulation with a nearly infinite extent for better accuracy. Third, the BSDF itself may be calculated at any resolution, time and computational resources permitting, all the way down to grazing angles.

The main disadvantage of the two-step approach is the difficulty in disassembling a CFS to get exact material and geometry measurements. Even small errors in the geometric model, something as seemingly unimportant as the sharpness of an edge, can make a large difference to the results. The original CAD files from the maker are a good starting point, but tolerances and other details about manufacturing process should be considered as well.

1.1. Fixed-resolution BSDF Data in WINDOW 6

LBNL's WINDOW 6 program maintains a library of fixed-resolution BSDF data for various window systems. There are currently three different coordinate system resolutions. The most commonly used "full resolution Klems basis" specifies 145 incoming and 145 outgoing directions, as shown in Figure 1. The hemisphere cells are numbered 1 (at the zenith) through 145 (near the horizon). Including the zenith, which only has a single cell, there are 9 theta (latitude) bands, and for each of these, a different number of azimuth divisions. The cells are arrayed to maintain roughly equal projected solid angle, though they do not maintain a uniform aspect ratio. (E.g., cells on the horizon are 30° from their neighbors compared to 10° spacing around the zenith.)

One of the more interesting aspects of the Klems basis is its use of coordinate axis reversal for convenient back-front symmetry in a matrix representation. Without this exchange of left-handed and right-handed coordinates, a pane of clear glazing would not give us a diagonal matrix. With axis reversal, both the transmitted and the reflected matrices are diagonal for a simple, specular system. Moreover, this allows WINDOW 6 to characterize layered systems using a special form of matrix concatenation that accounts for interreflected components [Klems94a]. There are of course limits to the accuracy of such methods when systems contain potentially interacting periodicities, and these are discussed in [Klems94b].

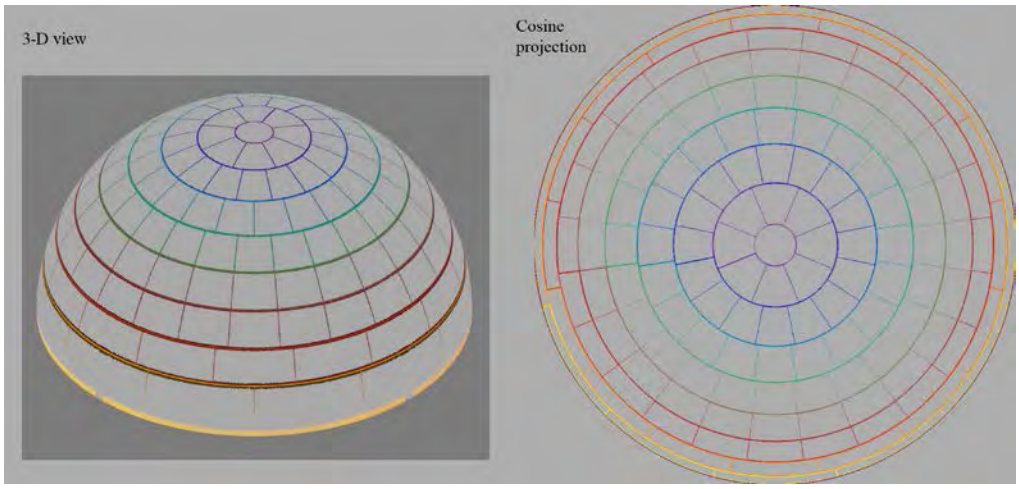

Figure 1. *The full Klems basis for dividing the hemisphere, shown in a perspective projection on the left, and a cosine projection on the right.*

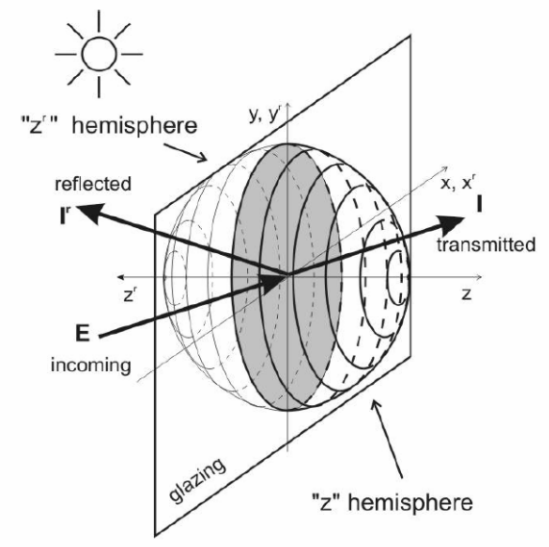

Figure 2. *The Klems X-coordinate axis is reversed on the interior versus exterior sides of the system to simplify a matrix representation.*

The BSDF data is contained in an eXtensible Markup Language (XML) file designed by Mitchell et al., which also defines the coordinate basis and related data [Mitchell08]. The XML prologue is human-readable and designed to be self-explanatory, although the BSDF data blocks themselves are rather large and difficult to parse visually. There are two important restrictions to the BSDF format as defined and used by WINDOW 6. First, only square matrix data is supported. That is, the number of incoming and outgoing directions must be equal. Second, only one coordinate basis is supported in each file. This may be one of the Klems bases, or any similar basis that uses separate theta latitudes and a fixed number of phi cells within each. These restrictions are designed to facilitate the extended matrix concatenation mentioned earlier. However, layering systems with different BSDF resolutions still requires resampling within WINDOW 6.

Such restrictions are unnecessary in *Radiance*, and were lifted in its implementation of matrix BSDFs. Besides supporting multiple angle bases and non-square matrices, a “Geometry” tag was added to the XML specification to store a detailed model of the CFS to improve rendered appearance and shadows.

1.2. Existing Support for Fixed-resolution BSDFs in Radiance

Up until 2011, WINDOW 6 BSDF files were understood by just two *Radiance* programs, **mkillum** and **dctimestep**. The *Radiance* 4.0 **mkillum** utility can read the BTDF associated with a window or CFS and compute that window's light output for a particular illumination condition, improving the speed and quality of renderings. The **dctimestep** utility is employed in the inner loop of a three-phase daylight coefficient calculation, where it takes the BTDF associated with a particular CFS configuration and multiplies it against the sky vector and form factor matrices to arrive at a vector of illuminance or luminance values [Ward11].

The operation of **dctimestep** is simple. It reads in three matrices and a sky vector (produced by **genskyvec**) and concatenates them to obtain a result vector. The XML file containing the BTDF matrix must be converted to a form factor by multiplying each entry against the projected solid angle for the associated Klems patch, which is straightforward. Most of the time in fact is spent reading the input files.

The behavior of the 4.0 version of **mkillum** is considerably more complicated. In the simplest case, a set of hemispherical ray samples are distributed over the Klems basis directions for a thin window element. The results are then passed through the BTDF and used to determine the distribution of light into the room. This distribution is written out as a data file that any of the other *Radiance* programs may use for rendering. When the XML file includes a "Geometry" tag with a geometric description of the CFS, **mkillum** separates the "beam direct" component from the other components, also converting the Materials and Geometry Format (MGF) in the XML file into a *Radiance* scene description. This model may then cast complex shadows into the space during rendering, such as the stripes from a Venetian blind.

Neither of these utilities reads or uses the reflectance portion of a WINDOW 6 BSDF, and the 4.0 version of **mkillum** was unable to account for light reflected off the interior of a complex fenestration system when described by an XML file. This shortcoming has been remedied in *Radiance* 4.1 by eliminating direct use of BSDF data in **mkillum** in favor of the new *BSDF* material type described in Section 3. However, there is way to directly account for changes in the BRDF of a CFS using **dctimestep**, although interior reflectances may exhibit an average behavior so they are not completely neglected.

A third *Radiance* program, **genBSDF**, may be used to produce a WINDOW 6 XML file from a detailed model of CFS geometry and materials. The 4.0 version of **genBSDF** only computes the transmitted (BTDF) component, and only for the full-resolution Klems basis. Also, it cannot read or use other BSDF data in its simulation.

Besides the inability of *Radiance* 4.0 to account for reflected components, there were issues with the limited angular resolution of WINDOW 6 BSDF data. Particularly in the study of glare conditions, the resolution of the window output is critical to making accurate predictions. In some cases, the limited resolution of the default Klems basis may also cause sizable errors in the calculation of workplane illuminance.

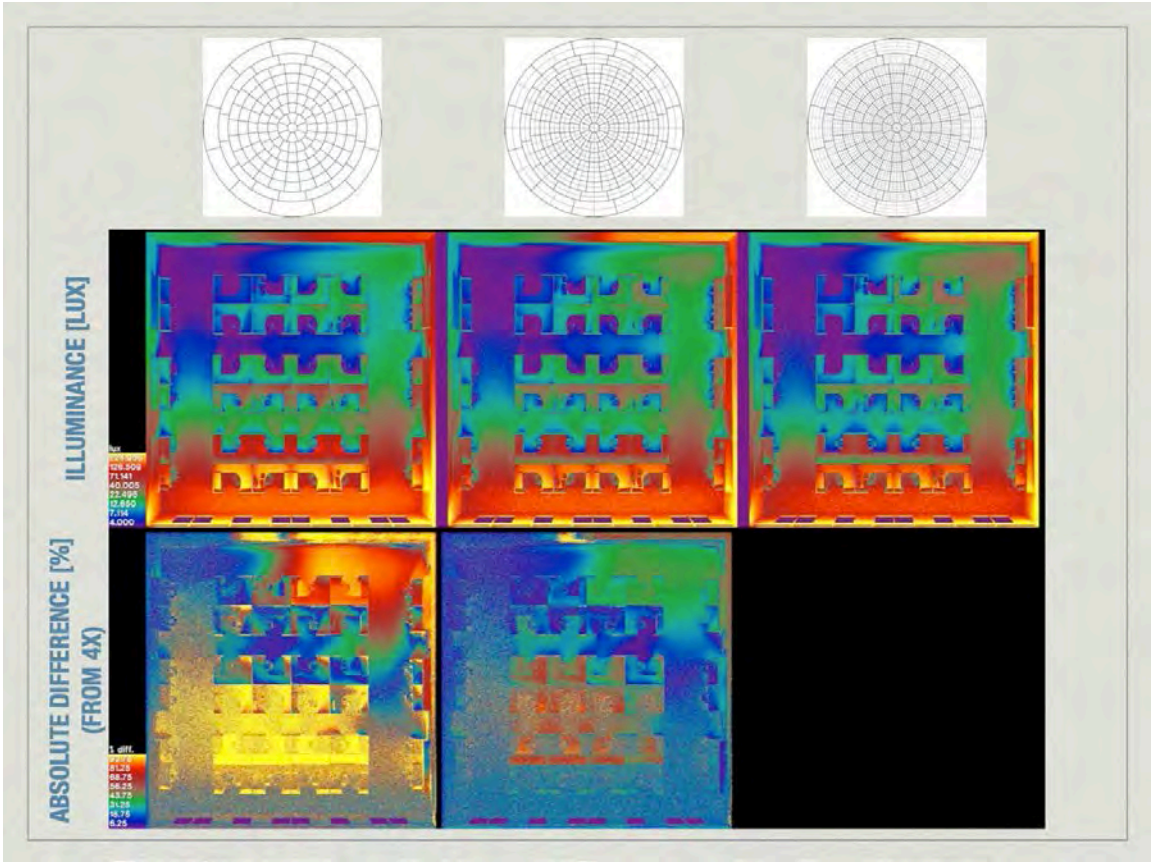

Figure 3. *Comparison between default "full-resolution" Klems basis and further subdivisions to increase accuracy.*

Figure 3 shows a few results from a study by Andrew McNeil, where he compared the full-resolution Klems basis to two "super-resolution" versions, one with 4 times as many hemispherical divisions (580 rather than 145) and one with 16 times as many (2320) [McNeil11]. These data were generated by a modified version of the *Radiance* **genBSDF** program based on a geometric model of a particular CFS with shaped, specular louvers. The hemispherical subdivisions are shown along the top row, with results for each basis below. The error plots show over 100% errors on the work-plane at the back of the room due to the poor angular resolution of the default basis and its inability to accurately place the beam component. This worst-case performance from the study is peculiar to the selected model, orientation and time, but demonstrates how BSDF resolution occasionally matters even to work-plane illuminance calculations.

With 580 directions for incoming and outgoing vectors, the errors in the example above were still greater than 30%. The higher density of 2320 directions entails a BSDF matrix with 5.4 million entries in an XML file that is over 50 Mbytes long. Much of this distribution is close to uniform – higher angular resolution is only needed in the specular directions. If we could introduce some method to resolve the peaks without requiring higher resolution everywhere, we could attain equivalent accuracy in a fraction of the memory. This is the basic idea behind the variable-resolution BSDF.

1.3. Variable-resolution BSDF Requirements

The main requirement for a variable-resolution BSDF representation is that it captures the peaks in a distribution while compressing smooth regions. While a number of schemes could be used to achieve this goal, we wish to avoid methods that either: (a) make the interpretation of the recorded data ambiguous, or (b) limit our ability to generate samples efficiently. We therefore prefer a BSDF data structure that is simple and can be queried directly.

This rules out most transform-based compression schemes such as DCT and wavelets, which require processing thousands of coefficients to evaluate the function at a single point.

To minimize our memory and disk footprint, we need a mechanism for scaling correlated incoming and outgoing specular directions at the same time. Otherwise, we would have to record all incoming directions at the peak resolution, even if the outgoing directions had variable resolution (or vice versa).

Given our goal to apply BSDFs in a ray-tracing context, the representation must be efficient for Monte Carlo sampling. This implies that we can quickly build a table from the cumulative distribution for a given incident direction, invert this table to use it as a sample generator.

We can further optimize our sampling methods and data requirements if we offer a specialized representation for isotropic materials. An isotropic BSDF is rotationally invariant, permitting us to eliminate one of the four dimensions. Since our goal is to record the distribution at the highest possible resolution, such simplification could save us considerable storage space.

2. The Tensor Tree BSDF Representation

The *tensor* is a generalization of vectors and matrices. A *rank-1* tensor is a vector, a*rank-2 tensor* is a matrix, and a *rank-3 tensor* can be thought of as a stack of matrices. We could use a rank-3 tensor to represent an isotropic BSDF and a rank-4 tensor for an anisotropic distribution. But instead of assigning dimensions (row lengths) to our tensors, we apply a binary tree subdivision that allows our resolution to vary locally.

Figure 4 shows an octree as might be used to store the rank-3 tensor of an isotropic BSDF. For incoming and outgoing angles where the distribution is uniform, large blocks can approximate the BSDF. In regions where there is greater variance, smaller blocks are required and the tree is subdivided. In effect, we have a representation that stands in for a rank-3 tensor of unlimited size in each dimension. Similarly, an anisotropic BSDF is represented by a hextree, which we subdivide into 16 branches wherever higher resolution is needed.

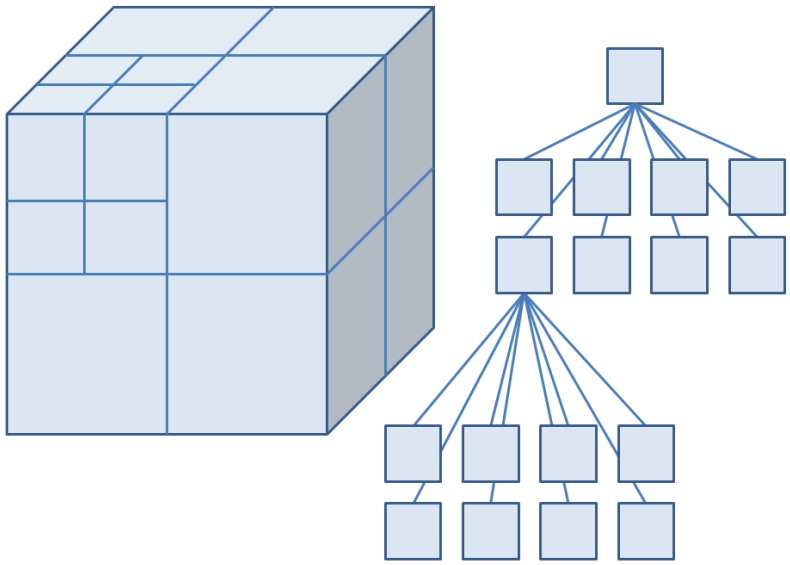

Figure 4. *An octree representation showing how resolution can vary locally in order to capture our desired BSDF sample density.*

2.1. Shirley-Chiu Disk to Square Mapping

Tensor trees as described are a convenient way to satisfy our main requirements of direct queries and scaling input and output resolutions simultaneously. However, they have a basic property that makes them awkward as a BSDF representation: they are rectangular. The BSDF incoming and outgoing directions are best defined on a flat disk projection of the hemisphere, since this eliminates the cosine term. We therefore need a way to map between a disk and a square while preserving local area. Such a mapping was devised by Shirley and Chiu [Shirley-Chiu98], and is shown in Figure 5. A BSDF query is resolved by projecting the incident vector direction onto a disk, then mapping to our square domain using the Shirley-Chiu mapping. The same is done for the exiting direction, after which we can look up the BSDF value in our rank-4 tensor tree. Such a query method is direct, simple, and efficient.

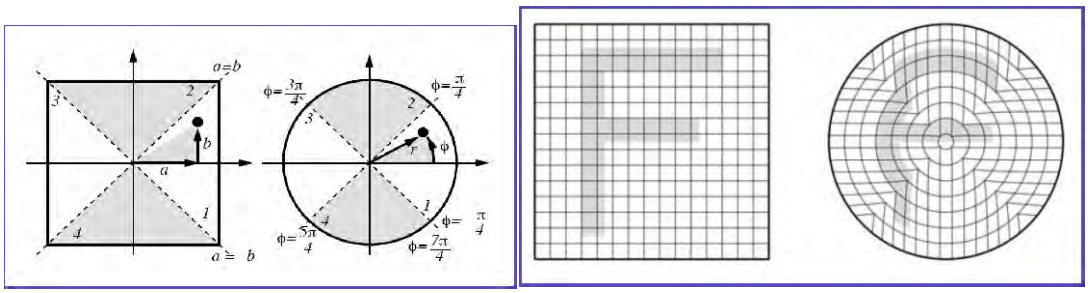

Figure 5. *Shirley and Chiu's area-preserving map between a square and a disk.*

2.2. Isotropic versus Anisotropic BSDFs

The anisotropic case is simpler than the isotropic variant. With two incoming and two outgoing angles, we employ the Shirley-Chiu disk↔square mapping to go into and come out of a rank-4 tensor tree representation as just described. In the isotropic case, we need first to rotate our incident angle to the measurement plane, then look up the two outgoing directions in a rank-3 tensor tree. The outgoing directions are then mapped to a disk that must then be rotated backwards by the same angle as we rotated the incoming vector.

2.3. Additions to the WINDOW 6 XML Format

Radiance 4.1 introduces new identifiers to the WINDOW 6 XML standard to support variable resolution BSDFs. The existing tag, <IncidentDataStructure>, gets two new types, "TensorTree3" and "TensorTree4," corresponding to isotropic and anisotropic data, respectively. The <AngleBasis> tag has the new type "LBNL/Shirley-Chiu" to indicate the disk↔square mapping rather than the Klems angle basis. Finally, the BSDF data itself is recorded using curly braces to differentiate subtree and leaf data. The ordering of branch blocks for anisotropic data is:

{
branch-X_i-Y_i-X_e-Y_e
branch+X_i-Y_i-X_e-Y_e
branch-X_i+Y_i-X_e-Y_e
branch+X_i+Y_i-X_e-Y_e
branch-X_i-Y_i+X_e-Y_e
…
branch+X_i+Y_i+X_e+Y_e
}

where -X_i stands for the lower incident X-coordinate, and +Y_e stands for the upper exiting Y-coordinate, etc. (These coordinates correspond to the horizontal and vertical axes in the square from the Shirley-Chiu disk mapping.) Each branch in turn is comprised of leaves or branch blocks enclosed in curly braces, and every block must contain either 16 subbranches, or 16^N leaves, where N is a positive integer. A leaf is simply a non-negative floating-point value encoded as ASCII text (e.g., "0.3183" or "6.15e-4"). A block containing multiple leaves implies that the BSDF in this subtree has uniform density, so curly braces are unnecessary. Leaves are sorted in the opposite order to

branches, where the Y_e dimension changes fastest, followed by X_e, Y_i, and X_i in that order. Branch blocks and leaf values may be separated by spaces, commas, or both.

For isotropic BSDF data, the base becomes 8 rather than 16, and only one incident coordinate is given (X_i). Furthermore, half the top-level branches are ignored, since symmetry makes this many X_i values redundant. Only X_i coordinates between 0 and ½ are considered valid.

Other XML extensions made in *Radiance* 4.0 are carried over to 4.1, including the <Geometry> tag that provides a detailed CFS description using the MGF format [Ward11], support for multiple basis functions and non-rectangular BSDF matrices (more incoming directions than exiting or vice versa).

2.3. Computing a variable-resolution BSDF with genBSDF

The 4.1 version of **genBSDF** includes new options "-t3" and "-t4" for generating isotropic or anisotropic tensor-tree distributions, respectively. Each option takes a positive integer specifying the maximum number of exiting directions as a power of 4. E.g., "-t3 5" would imply 4^5 or 1024 exiting directions. An isotropic distribution such as this would have at most 2^{N-1} incident directions (16 in this case) but the anisotropic case "-t4 5" would have the same number of incoming and exiting directions (1024). This initial sampling yields 50 times as many BSDF values as the full Klems basis, but **genBSDF** calls the private *Radiance* utility **rttree_reduce** to encode the data as a tensor tree. This reduces the BSDF payload by 90-95% while preserving sharp peaks in the distribution.

One of the issues when creating variable-resolution BSDFs using **genBSDF** is how to set the "-c" sampling option relative to the maximum resolution. If this parameter is set too low, too few samples will be gathered on the exiting side to obtain accurate estimates, which also makes it difficult for **rttree_reduce** to reduce the data properly. If the "-c" parameter is set too high, the program takes a very long time to finish, even when employing the "-n" option for multiprocessing. In the worst case, the "-c" parameter should scale with the number of exiting directions, but we have found much lower settings to work for most complex fenestration systems. Further sensitivity studies are warranted.

2.4. Stratified Monte Carlo Sampling

As we noted earlier, the tensor-tree structure is very efficient for direct query of BSDF values, but we also have a method to generate random samples for a given incident ray. For anisotropic distributions, we start by projecting our incident direction onto a disk and mapping it to a square. We then record all the BSDF values in the slice of our tensor tree that corresponds to this incident vector, building up a cumulative distribution table along a Hilbert traversal of exiting directions. This is the standard process of *Monte Carlo inversion*, but ordered with a Hilbert curve like the one shown in Figure 6.

A Hilbert space-filling curve is recursively defined on a binary grid such that lower curve resolutions cover lower subdivisions of the tensor tree in exact correspondence. This provides consistent ordering of our variable-resolution BSDF in a cumulative table. In practice, we use a very high-resolution (level 16) Hilbert curve and record the entry and exit index for each leaf node in our tensor tree.

Once we have constructed our table with cumulative distribution sums and corresponding Hilbert indices, we use it to generate stratified Monte Carlo sample directions. The caller passes us a (stratified) random variable between 0 and 1, and we identify the cumulative sums above and below this value with corresponding Hilbert indexes from our table using a binary search. The actual Hilbert index for this sample is then generated from the remainder of the random variable within the identified interval, yielding a nicely distributed position within the corresponding 2-D subregion. We then map the square position back to a disk and project it into a hemispherical vector and return this to the caller.

The described method is fast and efficient except for the computation of the cumulative table, which is too expensive to rebuild every time we need a sample. We therefore cache our cumulative tables in memory, reusing them for directions that fall within the same incident slice of the tensor tree. Ultimately, these tables may occupy more memory than the tensor tree itself, but never as much as would be required by a fixed resolution BSDF with

the same level of detail. If memory is constrained, a least-recently-used cache removal scheme could be implemented, but we have thus far not found it necessary.

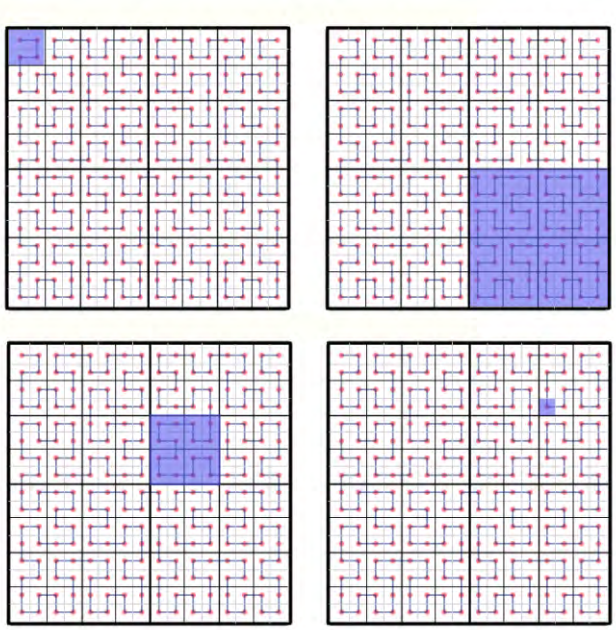

Figure 6. *A Hilbert 2-D space-filling curve drawn to the 4th level. We have shaded four different regions to illustrate a hypothetical tensor tree's subdivision of exiting angles. Larger areas are employed in uniform regions, and smaller areas are used to represent BSDF peaks.*

2.5. Comparison to WINDOW 6 Matrix BSDF Results

Figure 7a shows a corrugated material where one 45° slope is made of a Lambertian (perfectly diffuse) surface and the other 45° slope is a mirrored surface. Figure 7b shows our test room with loosely tiled material samples on the floor, a single overhead light source, and one checkerboard wall. The **genBSDF** utility was used to compute a BRDF for this composite using the full-resolution Klems basis and a rank-4 tensor tree with two different maximum resolutions.

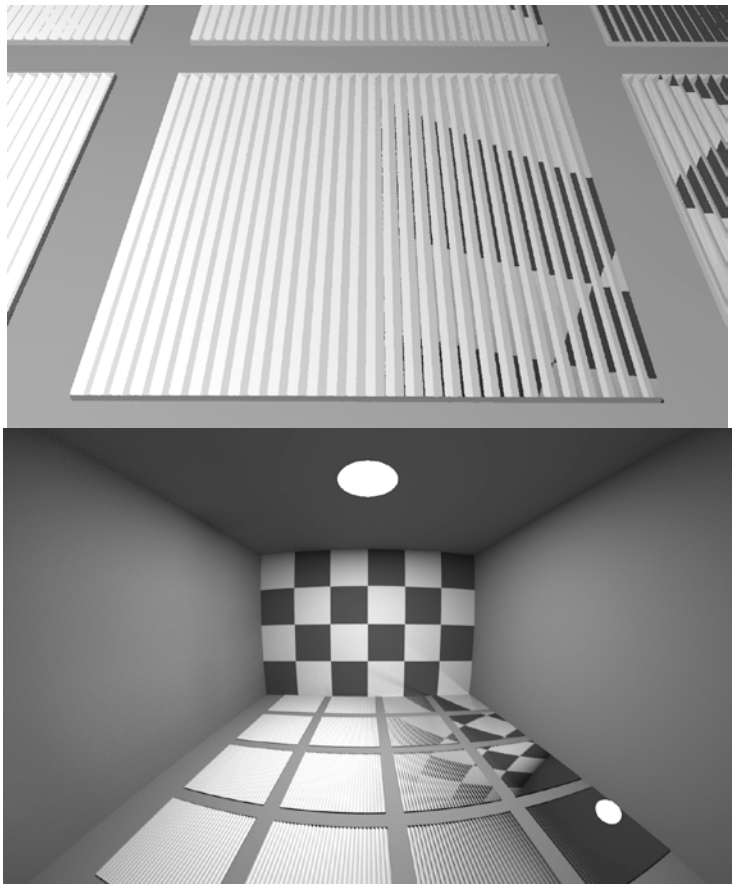

Figure 7. (a) *A close-up of a grooved material tile with alternating diffuse and mirror surfaces.* ***(b)*** *Our test environment.*

Figure 8 shows a rendering of this material modeled as a BRDF surface. The full-resolution Klems angle basis was used in this simulation, showing a rather vague reflection of the adjoining wall.

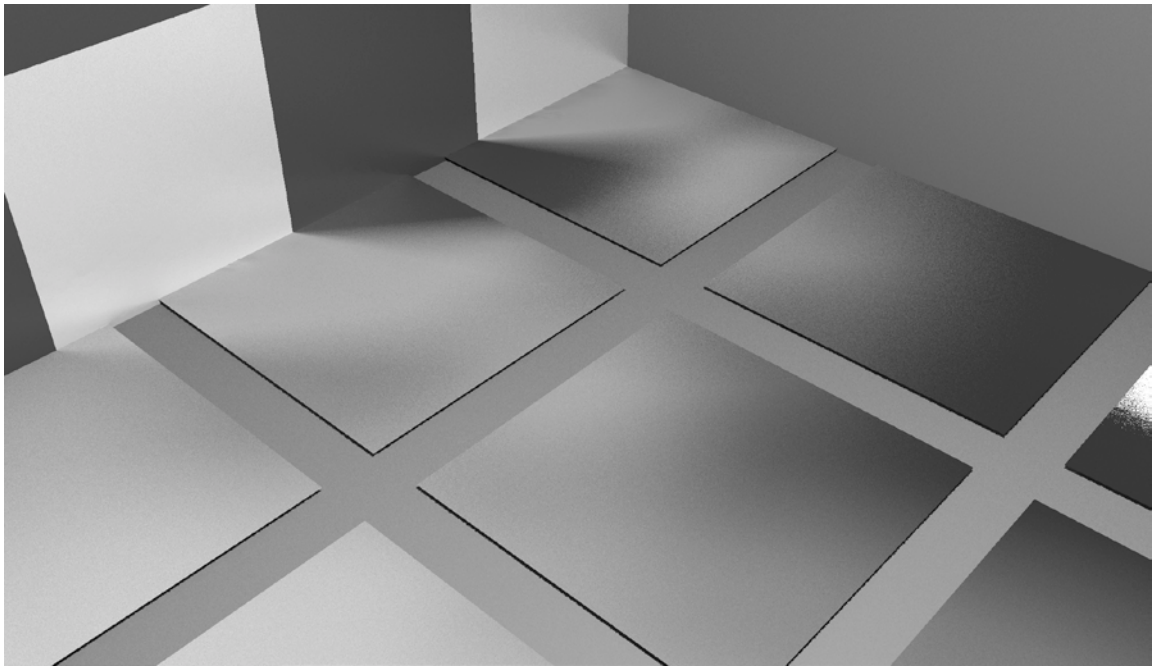

Figure 8. *A rendering of the grooved material modeled using the 145-angle Klems basis.*

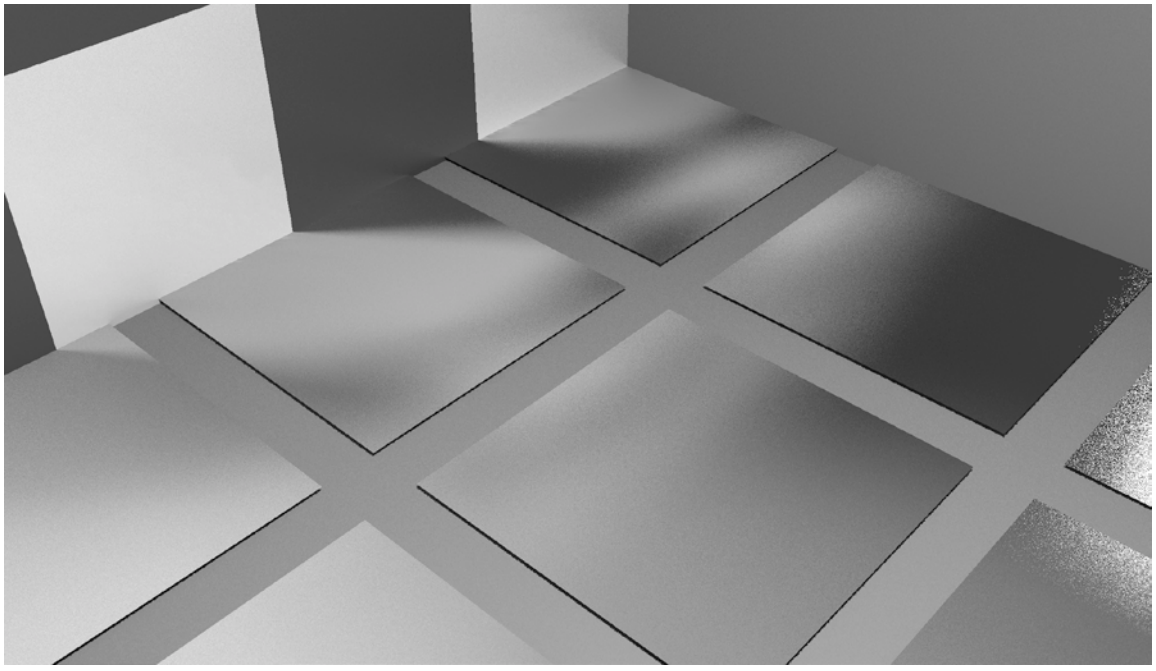

Figure 9. *A rendering using a variable-resolution BSDF reduced from 256 incident and 256 exiting directions.*

Figure 9 shows a rendering using a rank-4 tensor tree representation with a maximum resolution of 256 incident and 256 exiting directions. While the actual BSDF file size was 80% smaller than the Klems basis, we can see that the reflections are better resolved with a tensor tree representation. The resolution improves even further when we go to higher maximum resolutions.

Figure 10 shows a rendering from a tensor tree BSDF with a maximum resolution of 4096 incident by 4096 exiting directions. This is equivalent to a measurement every 1.5° near normal incidence. Unfortunately, even this resolution is not sufficient to achieve sharp reflections, despite the XML file being 30 times larger than the 145x145 Klems basis (17.6 Mbytes versus 538 Kbytes). Since a pixel in a typical perspective image occupies about 3 minutes of arc, sharp reflections would not happen until the maximum BRDF resolution were close to 4 million incident by 4 million exiting directions. While this is possible with a tensor tree representation, the computation and memory costs would likely outweigh the benefits. Specifically, the current implementation of **genBSDF** could not sample at this resolution in less than a geological epoch, and direct measurement with the PAB-Opto is likewise impractical.

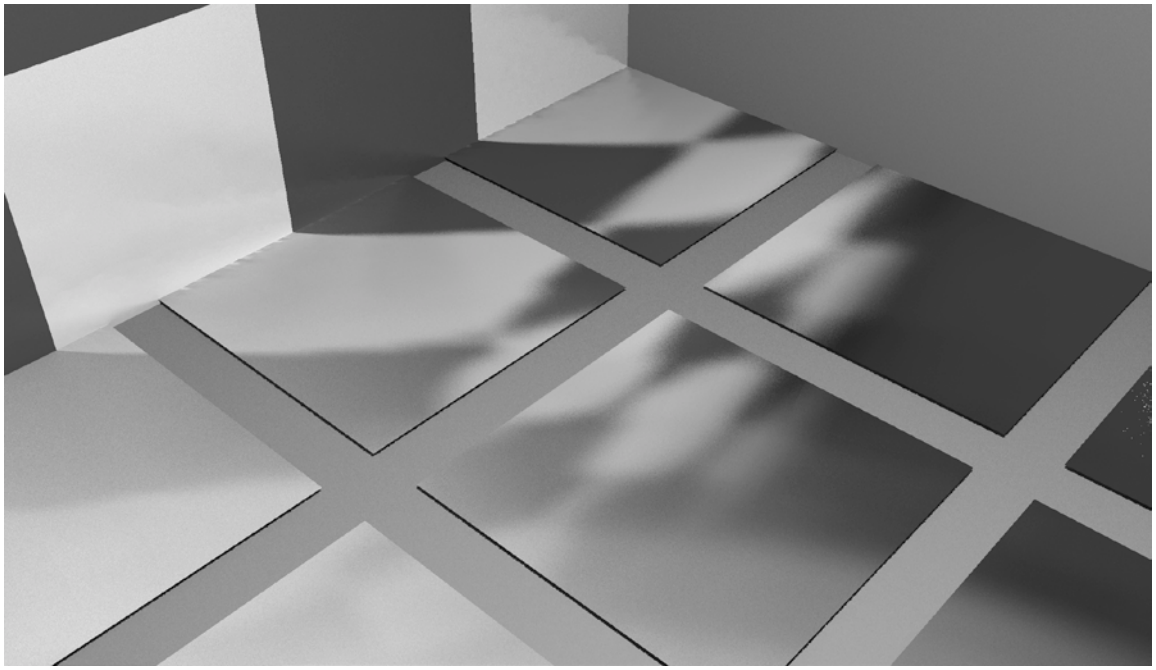

Figure 10. *A rendering using a variable-resolution BSDF reduced from 4096 incident and 4096 exiting directions.*

Rendering times for all of the images shown are comparable. In fact, the Klems representation takes about 10% longer to render than the others, and there is no computational penalty for rendering with the 4096x4096 tensor tree versus the 256x256 peak resolution tree.

3. A New *Radiance* Material Type

Radiance 4.1 provides a new primitive type, named simply *BSDF*. Figures 8 through 10 in the previous section were rendered using this new material, which specifies a WINDOW 6 XML file from which to draw ray samples using the Monte Carlo inversion method described earlier. Arguments orient the distribution relative to the surface and provide proxy behavior similar to the *illum* material for secondary light sources. Quoting the *Radiance* 4.1 User Manual:

> **BSDF**
>
> The BSDF material type loads an XML (eXtensible Markup Language) file describing a bidirectional scattering distribution function. Real arguments to this material may define additional diffuse components that augment the BSDF data. String arguments are used to define thickness for proxied surfaces and the "up" orientation for the material.
>
> ```
> mod BSDF id
> 6+ thick BSDF file ux uy uz funcfile transform
> 0
> 0|3|6|9
>         rfdif gfdif bfdif
>         rbdif gbdif bbdif
>         rtdif gtdif btdif
> ```
>
> The first string argument is a "thickness" parameter that may be used to hide detail geometry being proxied by an aggregate BSDF material. If a view or shadow ray hits a BSDF proxy with non-zero thickness, it will pass directly through as if the surface were not there. Similar to the illum type, this permits direct viewing and shadow testing of complex geometry. The BSDF is used when a scattered (indirect) ray hits the surface, and any transmitted sample rays will be offset by the thickness amount to avoid the hidden geometry and gather samples from the other side. In this manner, BSDF surfaces can improve the results for indirect scattering from complex systems without sacrificing appearance or shadow accuracy. If the BSDF has transmission and backside reflection data, a parallel BSDF surface may be placed slightly less than the given thickness away from the front surface to enclose the complex geometry on both sides. The sign of the thickness is important, as it indicates whether the proxied geometry is behind the BSDF surface (when thickness is positive) or in front (when thickness is negative).
>
> The second string argument is the name of the BSDF file, which is found in the usual auxiliary locations. The following three string parameters name variables for an "up" vector, which together with the surface normal, define the local coordinate system that orients the BSDF. These variables, along with the thickness, are defined in a function file given as the next string argument. An optional transform is used to scale the thickness and reorient the up vector.
>
> If no real arguments are given, the BSDF is used by itself to determine reflection and transmission. If there are at least 3 real arguments, the first triplet is an additional diffuse reflectance for the front side. At least 6 real arguments adds diffuse reflectance to the rear side of the surface. If there are 9 real arguments, the final triplet will be taken as an additional diffuse transmittance. All diffuse components as well as the non-diffuse transmission are modified by patterns applied to this material. The non-diffuse reflection from either side is unaffected. Textures perturb the effective surface normal in the usual way.
>
> The surface normal of this type is not altered to face the incoming ray, so the front and back BSDF reflections may differ. (Transmission is identical front-to-back by physical law.) If back visibility is turned off during rendering and there is no transmission or back-side reflection, only then the surface will be invisible from behind. Unlike other data-driven material types, the BSDF type is fully supported and all parts of the distribution are properly sampled.

The following subsections describe how the *BSDF* proxy mode operates and how this new material relates to other *Radiance* tools and techniques.

3.1. Maintaining Accuracy and Appearance via Proxied Geometry

Especially when used as a stand-in for complex fenestration, the appearance and behavior of a pure BSDF material might not be ideal. Specifically, the interior surface will look flat and undifferentiated relative to the original CFS, and when direct beam radiation is passed, it will lack the complex shadowing typical of real systems. The solution is to use the *BSDF* surface as a proxy for the actual CFS geometry, which we include in our model behind (and/or in front of) the proxy.

An example of such an arrangement is shown in Figure 11. Two *BSDF* surfaces, one inside and one outside, serve as proxies for the Venetian blind system sandwiched in between. The thickness for the front material is slightly larger than the actual system thickness so that indirect rays conveyed by this amount will completely avoid the CFS geometry. The back surface specifies a negative thickness to indicate the CFS is in front of the proxy rather than behind it. Rays from the eye or sent to light sources pass unhindered through the proxy geometry and interact with the CFS directly, yielding a more accurate appearance and complex shadows. Indirect rays see our proxy surfaces and get the aggregate BSDF behavior, reducing noise in the rendered result. The transmitted portion is offset by the specified thickness in order to bypass the CFS geometry, which is accounted for by the BTDF.

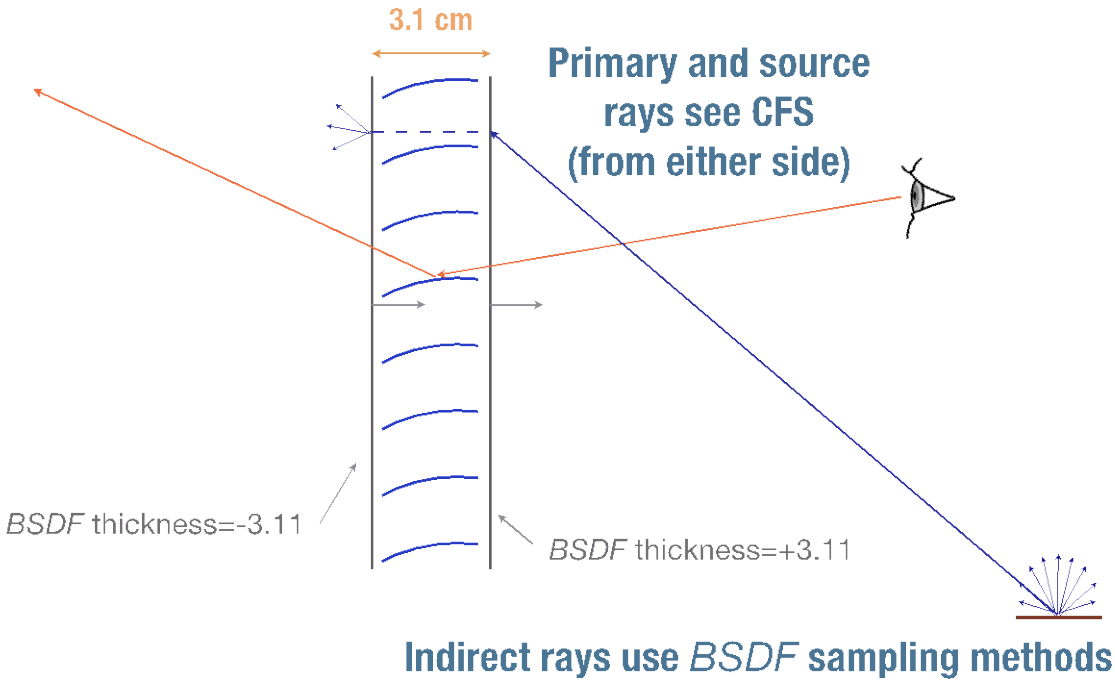

Figure 11. *An example of a CFS (venetian blind) in cross-section with BSDF proxy faces on the inside and outside.*

Figure 12 shows our example Venetian blinds converted to an XML file using **genBSDF** with a peak tensor tree resolution of 256 incident and 256 exiting directions. We have rendered it in Figure 12a without using the BSDF proxy mode or detailed geometry. Figure 12b shows the same data rendered in proxy mode with CFS geometry extracted from the XML file by **pkgBSDF**. The same rendering parameters were used in each case. Besides lacking detail in the shadows and blinds, the direct BSDF rendering suffers higher noise levels due to the undesirable indirection of beam radiation through the window. Average values are comparable, but the details are much improved using the proxy mode.

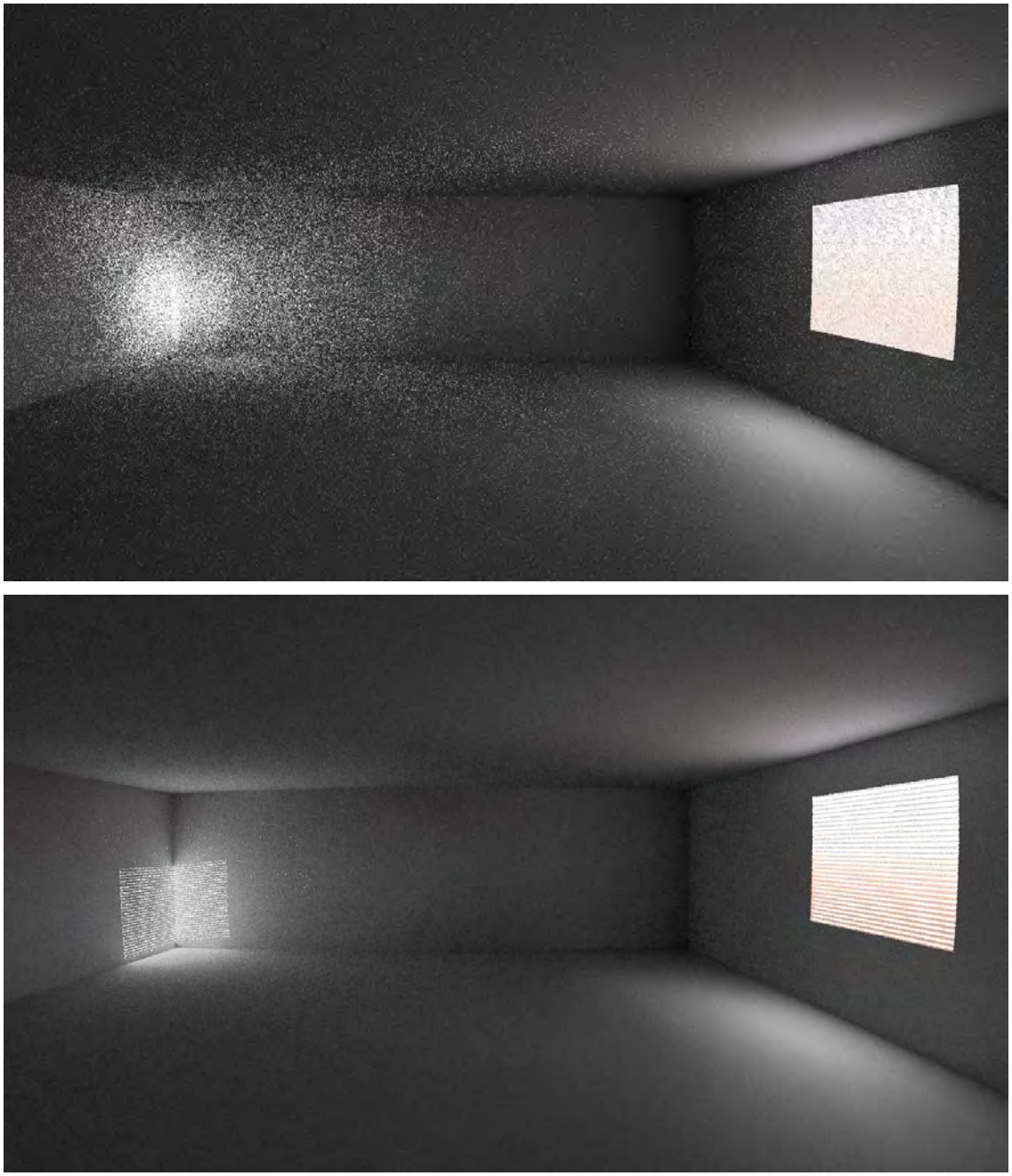

Figure 12. ***(a)*** *Venetian blinds on window rendered in default mode.* ***(b)*** *Venetian blinds rendered using BSDF proxy mode.*

3.2. Relationship to Other Radiance Programs and Methods

Since the new *BSDF* material is fully supported by the rendering core in *Radiance* 4.1, there are no restrictions on how or where it can be used. In the previous 4.0 release, BTDFs (transmission only) could augment calculations in **mkillum**, or be applied in the inner loop of an annual simulation with **dctimestep**. Now, both transmission and reflection are supported at the lowest level, and our variable-resolution format can represent even highly peaked BSDFs.

The following capabilities have thus been added in *Radiance* 4.1:

- Tabulated data for transmission *and* reflection may be applied in any simulation, illuminance calculation, or rendering
- New **–ss** rendering option reduces noise in specular highlights
- Fixed and variable-resolution BSDF files(reflection + transmission) can be generated by **genBSDF**
- Detailed CFS geometry can be extracted with **pkgBSDF** and modeled using the *BSDF* primitive's proxy mode
- Daylight coefficient calculations may be performed directly on complex fenestration systems using BSDFs independent of the three-phase method

There have been some related changes to the way things are done as well:

- **mkillum**'s direct use of XML files is deprecated, since the new *BSDF* material takes care of everything and handles reflection as well
- Since **mkillum** will not be extracting geometry from the XML files and we want to use it more generally, this functionality has been moved to **pkgBSDF**
- Implemented general BSDF handling and sampling library, which should aid in the adoption and standardization of WINDOW 6 XML format
- **dctimestep** uses the new BSDF library to read matrix data, and the old code will be removed as soon as **mkillum** stops calling it

At this point, a few restrictions remain, which are important to keep in mind:

- The *BSDF* proxy mode only works with "thin" systems
- BSDF data is completely uncolored
- Variable-resolution BSDFs are incompatible with the matrix-based implementation of our three-phase annual simulation method

The limitation to gray distributions will be lifted when an efficient means to measure and record color data is chosen. The BSDF support library can already handle spectral data and convert it to the RGB representation needed in *Radiance*.

Resampling the variable-resolution BSDF format to the Klems matrix needed for the three-phase DC method is possible but has not been implemented. Such a utility would also be useful for transferring BSDFs to WINDOW 6, which also requires matrix data.

4. BSDF Support Library

Our BSDF implementation is based on a C-language library that is designed with minimal dependencies on the rest of the *Radiance* distribution. We plan to distribute this code separately so that third parties can support BSDFs with minimum effort and maximum compatibility. Below, we describe the main and support functions provided by the library, followed by a discussion of extensibility.

4.1. Main Functions

The following functionality is provided by the BSDF library:

- Load/cache BSDF data from an XML input file
- Separate diffuse portion(s) and extract MGF detail (if any)
- Evaluate BSDF for a given vector pair (incident & exiting)
- Query the resolution (projected solid angle) of BSDF directions
- Compute directional hemispherical reflectance and transmittance
- Generate a properly distributed Monte Carlo sample for an incident vector
-

A BSDF may be treated as a combined entity or as components that can be queried and sampled separately. Components may be accessed logically, as in transmitted non-diffuse or reflected Lambertian, or they may also be

representation-specific. For example, a particular XML file may offer an uncolored component and a colored component, or a matrix component and a tensor tree component.

All color values are provided by the library as luminance and an associated power spectrum. Currently, the only power spectrum returned is equal energy white, meaning all components are uncolored. This is expected to change in a future release of the XML standard.

All operations commute between incident and exiting vectors, since a BSDF is assumed to obey reciprocity. An XML file may contain both front and back BTDF data, but only one distribution will be utilized.
The caller may either load and free BSDF data manually, or employ the caching mechanism provided by the library. The cache keeps a reference count and optionally frees the BSDF data when all callers have released their data pointers.

Function return values indicate 0 for success, or an error index on failure. A list of error strings is provided in ASCII-encoded English, and additional error details are usually available in an adjunct string.

4.2. Support Functions

In addition to the main functions listed above, a few useful support routines are provided. The conversion to and from local BSDF coordinates is facilitated with one call that computes the transform matrix and a second that inverts it. A third function maps and normalizes direction vectors using a forward or reverse transform matrix.
A call is also provided to free the cumulative distribution cache for individual components if memory is constrained. Finally, there are routines to access the underlying data representations, for example when a BSDF matrix is needed.

4.3. Extensibility

The BSDF library is designed to be extensible inasmuch as new representations may be added without requiring code changes on the application side. Simply downloading and linking the updated library should be sufficient to add support for spectral data or a new v BSDF representation.
Some design decisions limit library extensibility. For example, there is no mechanism for handling invisible BSDF components or polarization. Access to the underlying representations may also require code changes. If the matrix storage method is altered, programs such as **dctimestep** that access this data directly will need to be updated. This is normal and expected for programs that bypass the API to access library internals.

5. Next Steps

The variable-resolution representation and new *BSDF* type in *Radiance* open the way for new types of simulations. However, work remains in the areas of validation to make sure the calculation is accurate, and developing sources of measured BSDF data.

5.1. Validation Exercises

The proposed simplification of **mkillum** should be tested against its previous, validated behavior. Specifically, we need to compare the outcome when **mkillum** loads and applies the BTDF matrix directly to the core *BSDF* material results. In the latter case, **mkillum** will use the new built-in type to handle all interactions at the window. These results should agree with the previous method of applying the BTDF matrix internally excepting missing CFS reflections. By manually removing the reflection distribution from the XML file, we should be able to make a direct comparison to verify that the new method is working properly. At that point, we can remove the redundant code from **mkillum**.

We can validate the *BSDF* material sampling methods by comparing the results to existing *Radiance* material types. Since we know the BSDFs associated with plastic, metal, plastic2, trans, trans2, etc., we can create XML files that match these models and compare the rendered output using the *BSDF* type. We do not expect the results to match exactly, as there are differences in how light sources are sampled and so on, but there should be broad agreement.

The three-phase daylight coefficient method has been validated to some extent [Ward11]. Now that we have a direct method for simulations using BSDF data, we should compare the three-phase method against a standard daylight coefficient technique using the *BSDF* material. Since there are advantages and drawbacks to each method, we need to know they are both trustworthy and have an awareness of any potential differences.
Finally, we should perform some comparisons to scale models or data gathered from the Building 71T testbed.

5.2. Converting PAB-Opto Measurements to Variable-Resolution BSDFs

The PAB-Opto goniophotometer is designed to capture peaks in the BSDF at a higher resolution. Currently, this information is averaged into a Klems-basis matrix representation, losing most of its benefit. We hope to develop techniques for converting these data into a tensor-tree variable-resolution format. Since only exiting directions are sampled at high resolution, we need a reliable means to interpolate incident directions. This is a new research area, and it is difficult to predict our success.

Peter Apian-Bennewitz has been fitting measured data to *Radiance* material models and reported some significant disagreements in his invited talk at the 2011 workshop [Bennewitz11]. We feel it would be worthwhile to use the PAB-Opto measured data directly via the *BSDF* primitive and compare to standard *Radiance* material fits to understand exactly how these inaccuracies manifest in simulations. We will investigate numerical as well as visual differences, comparing also to other BRDF models from the literature.

Acknowledgments

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.

References

[Bennewitz11]. http://www.radiance-online.org/radiance-workshop10/presentations/day1/PAB1_FittingPlasticMetal.pdf

[Klems94a] Klems J.H. A new method for predicting the solar heat gain of complex fenestration systems: I. Overview and derivation of the matrix layer calculation. ASHRAE Transactions 100 (1): 1065-1072.

[Klems94b] Klems J.H. A new method for predicting the solar heat gain of complex fenestration systems: II. Detailed description of the matrix layer calculation. ASHRAE Transactions 100 (1): 1073-1086.

[McNeil11] McNeil, A. 2011. On the sensitivity of daylight simulations to the resolution of the hemispherical basis used to define bidirectional scattering distribution functions. DOE Technical Memo, September 30, 2011.

[Mitchell08] Mitchell R., Kohler C., Klems J., Rubin M., Arasteh D., Huizenga C., Yu T., Curcija D., editors. 2008. *Window 6.2/ Therm 6.2 Research Version User Manual. Lawrence Berkeley National Laboratory*. LBNL-941. Berkeley, CA 94720.

[Nicodemus77] Nicodemus, F.E., J.C. Richmond, J.J. Hsia, *Geometrical Considerations and Nomenclature for Reflectance*, U.S. Dept. of Commerce, Nat'l Bureau of Standards, Oct. 1977.

[Shirley-Chiu98] Shirley, P., K. Chiu, "A Low Distortion Map Between Disk and Square," *Journal of Graphics Tools*, 2(3), 1998, pp. 45–52.

[Ward95] Ward, G., *The Materials and Geometry Format*, web reference radsite.lbl.gov/mgf/mgfdoc.pdf

[Ward11] Ward, G., R. Mistrick, E.S. Lee, A. McNeil, J. Jonsson, "Simulating the Daylight Performance of Complex Fenestration Systems Using Bidirectional Scattering Distribution Functions within Radiance," *Journal of the Illuminating Engineering Soc. of North America*, April 2011.