Bi-Scale Radiance Transfer

Peter-Pike Sloan
Microsoft Corporation

Xinguo Liu
Microsoft Research Asia

Heung-Yeung Shum
Microsoft Research Asia

John Snyder
Microsoft Research

Abstract

Radiance transfer represents how generic source lighting is shadowed and scattered by an object to produce view-dependent appearance. We generalize by rendering transfer at two scales. A macro-scale is coarsely sampled over an object's surface, providing global effects like shadows cast from an arm onto a body. A meso-scale is finely sampled over a small patch to provide local texture. Low-order (25D) spherical harmonics represent low-frequency lighting dependence for both scales. To render, a coefficient vector representing distant source lighting is first transformed at the macro-scale by a matrix at each vertex of a coarse mesh. The resulting vectors represent a spatially-varying hemisphere of lighting incident to the meso-scale. A 4D function, called a radiance transfer texture (RTT), then specifies the surface's meso-scale response to each lighting basis component, as a function of a spatial index and a view direction. Finally, a 25D dot product of the macro-scale result vector with the vector looked up from the RTT performs the correct shading integral. We use an id map to place RTT samples from a small patch over the entire object; only two scalars are specified at high spatial resolution. Results show that bi-scale decomposition makes preprocessing practical and efficiently renders self-shadowing and interreflection effects from dynamic, low-frequency light sources at both scales.

Keywords: Graphics Hardware, Illumination, Monte Carlo Techniques, Rendering, Shadow Algorithms.

1. Introduction

Perhaps the greatest challenge in computer graphics is to model the interaction of light and matter at all relevant scales, from macro-level occlusion and interreflection between large objects down to micro-level quantum and diffraction effects. For efficiency, these scales are traditionally decomposed into geometry, texture, and lighting models [Kajiya 1985]. While this decomposition accelerates both Monte Carlo simulation and real-time hardware rendering, a gap in quality remains between the two.

To bridge this gap, recent work precomputes global transport effects in a way that can be exploited by fast graphics hardware. The idea is to tabulate over an object or small patch how source illumination is shadowed and scattered back to the viewer, in other words, *radiance transfer* as a function of spatial location and view. The result can then be quickly rendered from any viewpoint in a variety of lighting conditions. Current techniques precompute only at a single scale, and so are limited either to coarse [Sloan et al. 2002] or fine [Ashikhmin et al. 2002; Heidrich et al. 2000; Liu et al. 2001; Tong et al. 2002] effects.

Our approach instead models radiance transfer at two scales. A *macro-scale* encompasses coarse, geometry-dependent self-shadowing and interreflection effects such as shadows from a bunny's ears onto its back. A *meso-scale* models finer but still macroscopic structure such as snake scales or stuccowork. The first scale is precomputed over a given geometric object, via *precomputed radiance transfer* (PRT) [Sloan et al. 2002]. PRT stores a transfer matrix at each mesh vertex *p* that converts source illumination into transferred incident illumination, and so accounts for the object's global shadowing and interreflection onto itself. The second scale is precomputed over a small patch to obtain a 4D *radiance transfer texture* (RTT) which is mapped over an object using texture synthesis, as in [Liu et al. 2002]. At run-time, the transfer matrix at each *p* is first applied to a source lighting vector to produce transferred radiance at the macro-scale. The resulting vector is then dotted with a vector looked up from the RTT, indexed by a spatial location and view direction, to produce the final shade at *p*. The result provides view- and light-dependent global illumination effects at both scales.

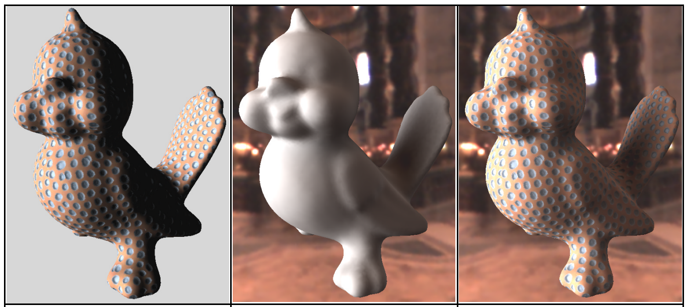

(a) BTF (meso-scale)	(b) PRT (macro-scale)	(c) Bi-Scale

Figure 1: BTFs (a) capture local effects like shading/shadowing over small dents. PRT (b) captures global effects like the bird's tail shadow, but is only practical at a coarse scale. Bi-scale rendering (c) combines coarse- and fine-scale global illumination effects for rendering on graphics hardware.

Bi-scale decomposition has a number of advantages. It samples each scale at the proper rate. In our experiments, the meso-scale is sampled over the surface roughly two orders of magnitude more highly than the macro-scale. Decomposition makes the precomputation practical, since global transport simulation at meso-scale sampling rates requires enormous computation and storage. It also accelerates run-time performance by performing expensive macro-scale computations at coarser sampling rates and retrieving meso-scale results from a small texture. Finally, it allows libraries of meso-textures (e.g., fabric weaves, animal skins, or wall surfaces) to be applied to different geometries.

PRT and RTTs both parameterize appearance by source lighting. Unlike previous fine-scale approaches, they use a low-order spherical harmonic (SH) basis rather than a directional basis. This allows effects like soft-shadows otherwise prohibited by the cost of integrating over many lighting directions. It also avoids aliasing when light sources move. Rendering is accurate if the lighting and BRDF are low-frequency. The technique thus forms a counterpart to traditional methods handling small (point) light sources.

Our main contribution is to simulate and render radiance transfer at both a coarse and fine scale. Our RTTs differ from bidirectional texture functions (BTFs) [Dana et al. 1999;Liu et al. 2002; Tong et al. 2002] by using an SH basis for lights. We also evaluate the RTT using an *id map* to better leverage texture mapping. We generalize PRT [Sloan et al. 2002; Kautz et al. 2002] by composing its result with a general RTT rather than a BRDF. This produces images where fine features shadow and mask each other in a complex, spatially varying way without neglecting large-scale effects on a particular object, as shown in Figure 1.

2. Related Work

Local Effects: We first discuss methods for representing texture effects, but ignoring global transport. The most general of these is the 6D bidirectional texture function (BTF), which encapsulates appearance that varies spatially and with light and view direction. It was first introduced by [Dana et al. 1999] for modeling meso-structure of real-world surfaces.

A densely sampled 6D BTF is too expensive to acquire or use for fast image generation, prompting many to resort to lower dimensional alternatives. Polynomial texture maps (PTMs) [Malzbender et al. 2001] lower dimensionality to 4D by storing coefficients of a bi-quadratic polynomial per texel. This can model surface appearance under varying lighting directions, but then must neglect view-dependent effects such as masking, foreshortening (parallax), and non-diffuse reflectance. Ashikhmin et al. [2002] also ignore view-dependence but use a steerable basis for directional lights that allows smoother lighting rotation. By precomputing visibility inside a height field and repeatedly tiling it onto a base geometry, Heidrich et al. [2000] simulate the local effects of self shadowing and indirect illumination.

Starting with a sparsely-sampled BTF, a denser sampling can be generated by recovering height fields and applying texture synthesis techniques [Liu et al. 2001]. Novel BTF textures can also be generated from synthetic height fields. These ideas are extended in [Tong et al. 2002] to generate BTFs and map them over arbitrary surfaces. BTF indices are synthesized as a per-vertex signal over a fine-scale mesh, whereas we use an id map to completely decouple the textured signal from the geometry.

Other methods have been developed to represent fully 6D BTFs for rendering on graphics hardware. Daubert et al. [2001] approximate a BTF using a product of a spatially-varying function of view times a spatially-varying set of parameters to a simple lighting model. Liu et al. [2002] factor 6D BTFs on arbitrary surfaces into a 4D *point appearance function* multiplied by a 2D *spatial modulation function*. As does ours, this approach parameterizes the surface but to create the spatial modulation function rather than an id map (see Section 3).

Spatially-varying BRDFs used for interactive performance typically represent reflectance change over a smooth surface rather than view- and light-dependent effects from texture with “depth”. McCallister et al. [2002] is an example, but unlike other methods discussed so far, it handles general lighting by prefiltering environment maps to avoid on-the-fly light integration.

Global Effects: PRT [Sloan et al. 2002] captures shadowing and interreflection of an object onto itself. Fast, hardware-accelerated rendering is performed for objects that are diffuse or isotropically glossy (but not highly specular), illuminated by low-frequency lighting that can be changed on-the-fly. Much related work, including previous use of the SH basis in computer graphics and other techniques for representing precomputed appearance, is described in that reference. [Kautz et al. 2002] generalizes the idea to surfaces with arbitrary, including anisotropic, BRDFs, and also demonstrated 1D spatial variation of a single BRDF parameter using a 3D texture map. This paper further generalizes PRT by incorporating a general 4D RTT capturing meso-scale effects mapped over the object.

3. Bi-Scale Transfer Representation

We begin by showing how our bi-scale representation both combines and decouples coarse (PRT) and fine (BTF) transfer methods of previous work. In the following, subscript p denotes that the subscripted variable spatially varies over a surface by sampling at the vertices p of a coarse mesh. u_p is a 2D texture coordinate and v_p the unit-length view vector at each vertex p. Capital letters denote matrices or large vectors; small letters denote scalars or small (2D or 3D) vectors. Function calls represent texturing where coarsely-sampled inputs are linearly interpolated between mesh vertices but the function is evaluated per image sample.

Macro-scale (PRT) transfer models exiting radiance as

$$B(v_p) \cdot (M_p\, L) \tag{1}$$

where L is an n-vector resulting from projecting source lighting into the SH basis, M_p is a $n{\times}n$ transfer matrix, and B is a 2D function producing an n-vector and represents the BRDF. Given a source lighting function $l(s)$ over the sphere $s \in \boldsymbol{S}$ (i.e., an environment map around the object), L is computed from a spherical integral yielding the approximation

$$L_i = \int_{s\in \boldsymbol{S}} l(s)\, y_i(s)\, ds, \quad l(s) \approx \sum\nolimits_{i=1}^{n} L_i\, y_i(s) \tag{2}$$

where $y_i(s)$ is the i-th SH basis function. If the lighting is low-frequency, a good approximation is obtained using a fifth order SH projection having n=25 basis functions [Sloan et al. 2002].

The right vector in (1), $M_p\, L$, is formed by transforming the lighting vector through the transfer matrix and produces incident radiance at p including self-shadowing and interreflection effects. M_p includes a rotation of the lighting from a global coordinate frame into a local frame aligned to the normal and tangent directions at p [Kautz et al. 2002]. The left vector, $B(v_p)$, represents the shading response in the view direction v_p given incident lighting expressed in the SH basis and in the local frame. Dotting these vectors integrates lighting times response over the hemisphere, yielding exiting radiance in the direction v_p [Sloan et al. 2002].

Though PRT captures global effects, recording it at meso-scale sampling rates is impractical. Transfer matrices were computed at a few tens of thousands of samples (vertices) in [Sloan et al. 2002]. At meso-scale sampling rates (~millions of samples over a surface), the Monte Carlo simulation would take hundreds of CPU hours for a single object. Real-time rendering would also be impossible because of the storage (25×25 matrix at millions of vertices) and computation needed (matrix/vector multiply at each vertex).

Meso-scale (BTF) rendering represents transfer from directional light sources. A 6D BTF models exiting radiance as $b(u_p,v_p,d)$ where d is the light direction. For real-time rendering, this is approximated in [Liu et al. 2002] using a singular value decomposition as

$$B(v_p, d) \bullet M(u_p) \tag{3}$$

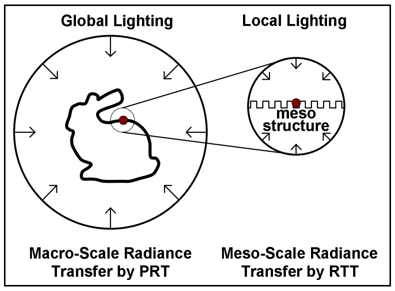
where B and M are vectors having 6-12 component colors (18-36 total channels). More channels increase approximation accuracy.

This representation easily handles lighting from a single lighting direction d. For large lights, B must be repeatedly evaluated and summed over many directions, becoming impractical for real-time rendering. Spatial modulation via M occurs only at one scale. Though the model is based on a small texture patch which was then synthesized onto an object, recording texture synthesis results still requires a very high-resolution texture map M.

Bi-scale rendering combines effects at both scales via

$$B(q(u_p),v_p) \cdot (M_p\, L) \tag{4}$$

B, M_p, and L are defined as for PRT (Formula (1)), but now B is a 4D function (radiance transfer texture) which allows spatial variation as well as view-dependence, as in a BTF. This model first performs coarse shadowing and interreflection via the lighting transformation $M_p\, L$, and uses the resulting vector as lighting incident on the meso-scale, represented by B. As shown in the figure below, this is not just scalar modulation of two separately shaded results from the two scales; the macro-scale produces a spatially-varying *radiance function* over the entire hemisphere illuminating the meso-scale.


Global Lighting
Local Lighting
meso structure
Macro-Scale Radiance Transfer by PRT
Meso-Scale Radiance Transfer by RTT

The RTT B specifies the meso-scale appearance of a small patch. Its i-th output channel, $B_i(u,v)$, encodes the response at location u in the direction v to incident lighting expressed using the i-th lighting basis function. Unlike a directional lighting basis, using the SH basis for lighting effectively pre-integrates the lighting response over large, smooth sources. The other novel aspect of this decomposition is the use of $q(u_p)$, the id map from which B's spatial index is looked up. The id map allows us to place B's samples *within* triangles of a coarse mesh in a fully general way, via precomputed texture synthesis (Section 4.3). Because spatial variation in B is tabulated using relatively few (64×64 or 128×128) samples, each of which is a 25D function of the view vector v_p, it becomes practical for graphics hardware to accelerate rendering using dependent textures. This is much more efficient than rendering a finely-sampled mesh [Tong et al. 2002].

Spatial variation in the bi-scale model occurs in two places: at a coarse scale in M_p, and at a fine scale via B's spatial index $q(u_p)$. The id map, q, contains only two output channels (two coordinates indexing the RTT). This should be compared to the 36 channels of the meso-scale map M from Formula (3), since both q and M are specified at high (meso-scale) spatial resolution. Moreover it is unclear how to extend M to include macro-scale effects, but it is likely even more channels would be required.

4. Precomputing Bi-Scale Radiance Transfer

4.1 Precomputing Macro-Scale Transfer Matrices

We precompute transfer matrices at each mesh vertex, using the Monte Carlo simulation described in [Sloan et al. 2002]. Essentially, this illuminates the geometry with source lighting at infinity represented using each of the SH basis function, and gathers the resulting transferred radiance. The resulting transfer matrices capture self-shadowing and self-interreflection but interreflections are only correct for lighting at infinity.

4.2 Building the RTT

Each spatial sample of the RTT linearly transforms source illumination into view-dependent exiting radiance. The spatial variation of this transformation is tabulated over a small patch. We use a low-order SH basis for lighting, yielding n=25 output channels for each view direction and spatial location. Results are shown in Figure 4, where the weave RTT is mapped to a simple square and rendered at two different viewpoints and 5 different lighting conditions. Note how the patch responds to area lighting and exhibits view-dependent appearance effects.

We generate B by rendering a small patch and recording images over a sampling of views and lighting directions. Theoretically, each channel $B_i(u_p,v_p)$ can be obtained by illuminating the geometry using the SH basis function $y_i(s)$ as a light source (noting that this is a nonphysical emitter of both positive and negative light). We compute B by rendering using a number of directional light sources, $l_k=(\theta_k,\varphi_k)$, k=1,2,…,*nlights*, sampling the hemisphere at 8 polar and 16 azimuthal directions. For a given view direction v_p, each B_i is then obtained by a weighted combination of the resulting *nlights*=8×16 images, $I_{k,vp}(u_p)$. Each weight is the product of an SH basis function, $y_i(l_k)$, times the solid angle, $da(l_k)$, evaluated at the direction sample l_k, yielding

$$B_i(u_p,v_p) = \sum_{k=1}^{nlights} I_{k,v_p}(u_p)\; y_i(l_k)\; da(l_k) \tag{5}$$

To produce the images $I_{k,vp}$, we 2×2 supersample across the 2D spatial parameter u_p and decimate using a box filter.

We assume that the patch's depth variation is much smaller than its tangential extent. Thick meso-structure requires a dense sampling of view directions that is impractical for hardware rendering. Given a thin patch, we sample images along a plane midway between the patch's depth extent, using conventional global illumination algorithms to generate the images for various viewing (v_p) and lighting (l_k) directions. For each view direction v_p, we directly trace rays passing through the sampling points on this sampling plane, in the direction v_p.

Patch geometry can be modeled as a height field [Liu et al. 2001]. Color textures can be applied to produce more vivid effects. The height field images for the "holes" and "bin" samples (Figure 5) were created by an image editing tool; the "stucco" wall sample used a 2D random process. Fully 3D models can also be used: the "weave" example consists of interleaved, deformed cylinders.

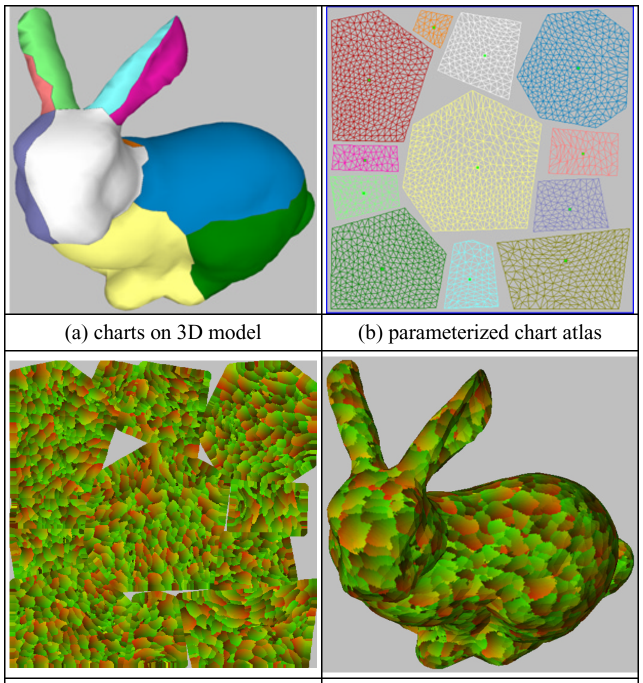
RTTs can also be acquired rather than generated from synthetic geometry. The RTT sample "plaster" (shown in Figure 5) was constructed from an example in the CUReT BTF database [Dana et al. 1999]. Because BTF samples in the CUReT data are sparse, a denser sampling was generated using the technique of [Liu et al. 2001]. Then the interpolated BTF data was converted to an RTT via Equation (5).

4.3 Generating the RTT ID Map

The RTT id map $q(u_p)$ maps meso-structure samples onto the macro geometry/surface. We generate the id map in three steps: (1) synthesize meso-scale texture over the 3D surface, (2) parameterize the surface, and create a 2D map of its geometry, (3) for each point in this 2D geometry map, find its RTT id from (1).

Synthesizing Meso-Scale Texture over the Surface straightforwardly applies the BTF synthesis algorithm in [Tong et al. 2002] to the surface. We first retile the surface into a dense mesh, called a *meso-mesh*. Its number of vertices (960k in our examples) is comparable to the number of samples in typical textures to allow a high frequency mapping. A texture synthesis algorithm [Turk 2001] is then invoked to synthesize a BTF spatial index at each meso-mesh vertex. Recall that the RTT is derived from the BTF by projecting its directional lighting samples to the SH basis via numerical integration (Equation (5)). Since both maps represent appearance of the same patch (only the lighting basis changes), the *spatial* index of the BTF can be used on the RTT.

To use graphics hardware efficiently, we parameterize the surface to convert the finely-sampled, per-vertex signal on this meso-mesh into a 2D texture.



(a) charts on 3D model

(b) parameterized chart atlas

(c) id map in texture space

(d) id map on 3D model

Figure 2: Surface Parameterization and ID Map Generation. The two id map channels are mapped to red and green in (c) and (d).

Parameterizing the Surface is done in three steps. The first step partitions the mesh into charts, the second parameterizes each chart, and the third packs charts together into a single atlas. A very similar technique is used in [Liu et al. 2002] to create the modulation map M from Formula (3); we use it here to texture RTT indices.

To create charts, we manually choose cutting paths using a simple UI (see example in Figure 2a). A user-specified number of 3D points along each chart's boundary are chosen as *corner* points, p_i, which map to 2D vertices of the chart's parameterization polygon, v_i. Initially these points uniformly subdivide the length of the chart's boundary, but they can be modified by the user.

We parameterize each chart into a convex, 2D polygon whose vertices v_i lie on a circle, and where distance between consecutive vertices $\|v_i - v_{i+1}\|$ is proportional to path length of the chart boundary between p_i and p_{i+1}. The interior of each chart is parameterized using the method of [Sander et al. 2001].

Automatic chart packing is a hard problem, so we do it manually. A packing example is shown in Figure 2(b).

Packed patches should not touch; if they do, bilinear reconstruction during rendering blends between charts that aren't neighbors on the mesh. We therefore reserve a one texel space around each texture polygon and dilate the texture to this "gutter" space.

Constructing the ID Map begins by rendering the 3D surface point and normal into each texel of the id map by scan converting the mesh triangles into texture space. That is equivalent to creating a map of the surface geometry including normal.

For each id map texel, we use "normal shooting" [Sander et al. 2001] to find its closest meso-mesh vertex. Given a texel with location P and normal N, we compute this by first finding the k vertices nearest to P in the meso-mesh as candidates. We select one of these by comparing distances of each candidate to the line through P along N. This nearest candidates's RTT location (i.e., the 2D coordinate of its associated RTT sample) is recorded into the id map. The ANN tool [Mount 1998] accelerates the k-nearest-neighbor query. We use k=20, to allow at least a 2-ring of neighboring vertices assuming an average vertex degree of 6.

We use id map resolutions of 2048×2048. A visualization of the id map, in texture space and mapped onto the 3D model, appears in Figure 2c-d. This is the synthesis result for the "weave" RTT appearing in Figure 5.

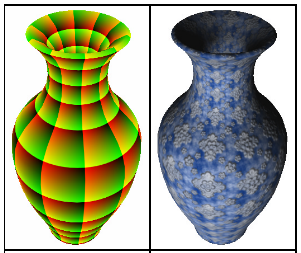
5. Rendering with Bi-Scale Radiance Transfer

Lighting: The source lighting vector L can be computed in various ways [Sloan et al. 2002]. We can dynamically rotate a predefined environment to simulate rigid rotations of the object. Graphics hardware can be used to sample radiance near the object which is then SH-projected via Equation (2). Simple light sources like circles can be analytically projected.

Shading: We compute the per-vertex matrix/vector multiplication $M_p\ L$ on the CPU since the matrix M_p is too large to be manipulated by a vertex shader. The 25D result, representing radiance incident on the meso-scale, is interpolated over triangles by the rasterization hardware. The id map and RTT are accessed using dependent texture mapping, producing another 25D vector $B(q(u_p),v_p)$. The two resulting vectors are dotted in a pixel shader.

Accessing the RTT: B is a 4D texture sampled using 64×64 spatial samples (u_p), and 8×8 view samples (v_p). The view samples are parameterized over a hemisphere using an area preserving map from the unit square to the hemisphere [Shirley et al. 1997].

B's samples are organized as a 2D texture of 8×8 view blocks. Contiguous view samples allows texture mapping hardware to perform smoother (bilinear) interpolation across views. Interpolation over spatial samples is prohibited, but the RTT is spatially smooth enough to produce good results using nearest-neighbor sampling. Image supersampling improves results.



trivial id map

Mapped "flower"

Surfaces defined parametrically already have a mapping (i.e., a u_p at each vertex) which can be used directly rather than as an index to an id map. This allows only continuous replication of the texture square over the surface and typically causes stretching and shrinking on curved objects. The figure on the left shows a parametric vase example with a trivially mapped "flower" RTT.

Diffuse Special Case: Assuming the local surface is diffuse and neglecting local masking effects, we can eliminate B's dependence on v_p. This greatly reduces texture memory and bandwidth, and allows higher spatial sampling. Dynamic shadowing and interreflection effects are retained; but shading is constant regardless of view. Even though the local surface is diffuse, we still need a transfer matrix at each point rather than a transfer vector to allow local shadowing and interreflection effects from the RTT.

6. Results

View-dependent effects of bi-scale rendering can be judged in Figure 4 and Figure 6. The RTT mapped to a simple square in Figure 4 changes appearance as the view changes (top row vs. bottom). We can see structure visible underneath the weave in the oblique view that's invisible in the head-on view. These effects are even more striking when the RTT is mapped onto a 3D object (Figure 6c-d). We modified our viewer to optionally switch off view-dependence by accessing only the "center" view sample of the RTT rather than indexing via the actual view direction. Note how the weave texture changes appearance with view angle around the body of the bird in (d), but appears pasted on in (c).

Figure 4a-d shows how the RTT responds to light sources of various sizes and directions. The (a) and (c) columns use the smallest (highest frequency) light source representable by our n=25 SH basis. The (b) and (d) columns illuminate from the same directions but use a bigger light source (analytic circle subtending a 110° angle). Shadows are softened, as if the RTT were illuminated on a cloudy day. The rightmost column uses a spherical lighting environment (“grove”) acquired from high dynamic range imagery, providing a realistic appearance.

Figure 1 and Figure 6a-b show how macro-scale transport improves image realism. Without macro-scale shadowing, models exhibit a “glowing” appearance revealing their synthetic nature (Figure 1a, Figure **6**a). PRT fixes these problems (Figure 1c, Figure **6**b) making the illusion more complete. Figure 3 shows a similar example using a rock model. This figure compares results with no global transport (a), shadowing transport (b), and interreflection+shadowing transport (c).

Our rendering method allows fast manipulation of the view or lighting with proper shadowing and interreflection effects at both scales. We achieve a frame rate of ~14.5Hz for the bird/bunny/rock models with the weave/bin/stucco RTT, rendering to a 512×512 screen window. The macro-scale of the bunny, bird, and rock models was represented using a 25×25 PRT transfer matrix at each of ~10,800 vertices. View-dependent RTTs (weave, holes, bin) are sampled at 64×64×8×8 while “diffuse” RTTS (plaster, stucco, flower) are sampled at 128×128. Id maps were sampled at 2048×2048. The vase model renders at 35.3Hz with a transfer matrix at each of 4453 vertices. Timings were conducted on a 2.2Ghz Pentium IV with ATI Radeon 9700.

Addressing preprocessing costs, meso-structure ray tracing to build the RTT (Section 4.2) requires 0.7-3.0 seconds per image. It is proportional to the meso-mesh complexity, patch size (64×64), and supersampling rate (2×2). 8×16×8×8 total images are ray traced over light and view direction. Conversion of the directional lighting basis to the SH basis is fast: about 5-10 minutes. Texton extraction (which precomputes distances between BTF sample pairs for synthesis) ranges from 1-3 hours and is proportional to the patch’s spatial size. Texture synthesis requires about 4 hours, longer than that reported in [Tong et al. 2002] since we use a larger searching window and a denser meso-mesh. Total time to create the bunny+weave samples was about 8 hours. PRT simulation (Section 4.1) takes about 8 minutes (at ~10k vertices) for shadowed transport and about 17 minutes for the rock example where we simulated three further light bounces.

Conclusions and Future Work

Bi-scale decomposition of radiance transfer provides global transport effects, ignored by local (BTF) methods, at spatial sampling densities impractical for PRT. Fully general light and view change is supported at interactive rendering rates. Though it limits rendering to distant, low-frequency illumination, our use of the spherical harmonic basis eliminates aliasing artifacts for dynamic lighting and provides soft shadows cheaply.

In future work, we are interested in improved filtering over the RTTs spatial index. Rendering into textures which can then be filtered spatially before mapping onto an object may help. We are also interested in more rigorous sampling of transfer fields that guarantee all “interesting” shadowed areas can be captured without wasteful oversampling.

Acknowledgments

We thank Dan Ling for suggesting this project, David Thiel for editing the video, and Jingdan Zhang for help on BTF synthesis.

References

ASHIKHMIN, M, AND SHIRLEY, P, Steerable Illumination Textures, ACM Transactions on Graphics, 2(3), 2002.

DANA, K, VAN GINNEKEN, B, NAYAR, S, AND KOENDERINK, J, Reflectance and Texture of Real World Surfaces, ACM Transactions on Graphics, 1999, 18(1):1–34.

DAUBERT, K, LENSCH, H, HEIDRICH, W, SEIDEL, H, Efficient Cloth Modeling and Rendering, EG Rendering Workshop 2001.

HEIDRICH, W, DAUBERT, K, KAUTZ, J, AND SEIDEL, H, Illuminating Micro Geometry based on Precomputed Visibility, SIGGRAPH 2000, 455-464.

KAJIYA, J, Anisotropic Reflection Models, SIGGRAPH 1985, 15-21.

KAUTZ, J, SLOAN, P, AND SNYDER J, Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics, Eurographics Workshop on Rendering 2002, 291-296.

LIU, X, YU, Y, AND SHUM, H, Synthesizing Bidirectional Texture Functions for Real-World Surfaces, SIGGRAPH 2001, 97–106.

LIU, X, HU, Y, ZHANG, J, TONG, X, GUO, B, AND SHUM, H, Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces, submitted for publication to IEEE TVCG, Nov, 2002.

MALZBENDER, T, GELB, D, AND WOLTERS, H, Polynomial Texture Maps, SIGGRAPH 2001, 519-528.

MCALLISTER, D, LASTRA, A, AND HEIDRICH, W, Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions, Graphics Hardware 2002.

MOUNT, D, ANN Programming Manual, Dept. Comp. Sci., Univ. of Maryland, College Park, Maryland, 1998. http://www.cs.umd.edu/~mount/ANN/

SANDER, P, SNYDER, J, GORTLER, S, AND HOPPE, H, Texture Mapping Progressive Meshes, SIGGRAPH 2001, 409-416.

SHIRLEY, P, AND CHIU, K, A Low Distortion Map between Disk and Square, Journal of Graphics Tools, vol. 2, no. 3, 1997, 45–52.

SLOAN, P., KAUTZ, J, AND SNYDER J, Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments, SIGGRAPH 2002, 527-536.

TONG, X, ZHANG, J, LIU, L, WANG, X, GUO, B, AND SHUM, H, Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces, SIGGRAPH 2002, 665-672.

TURK, G, Texture Synthesis on Surfaces, SIGGRAPH 2001, 347-354.

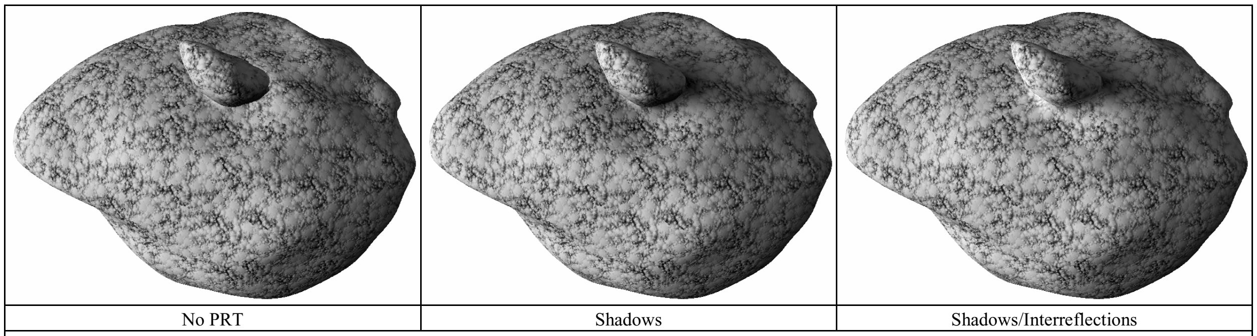
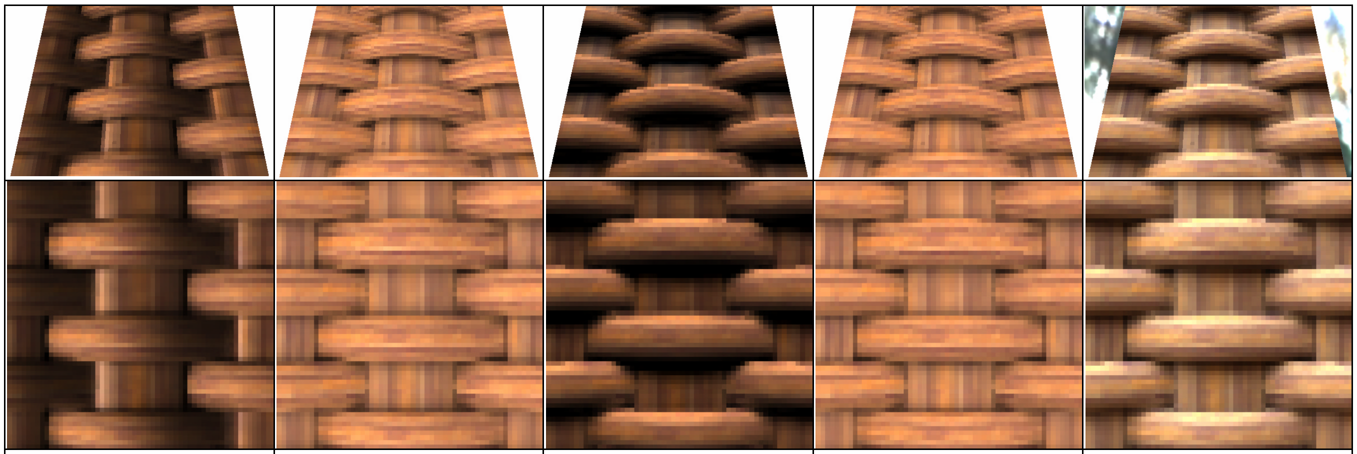


Figure 3: Rock model with three types of PRT transport.



(a) Light 1, 0°	(b) Light 1, 110°	(c) Light 2, 0°	(d) Light 2, 110°	(e) Lighting environment

Figure 4: RTT mapped to a square, two different views (rows) × 5 different lights (columns).

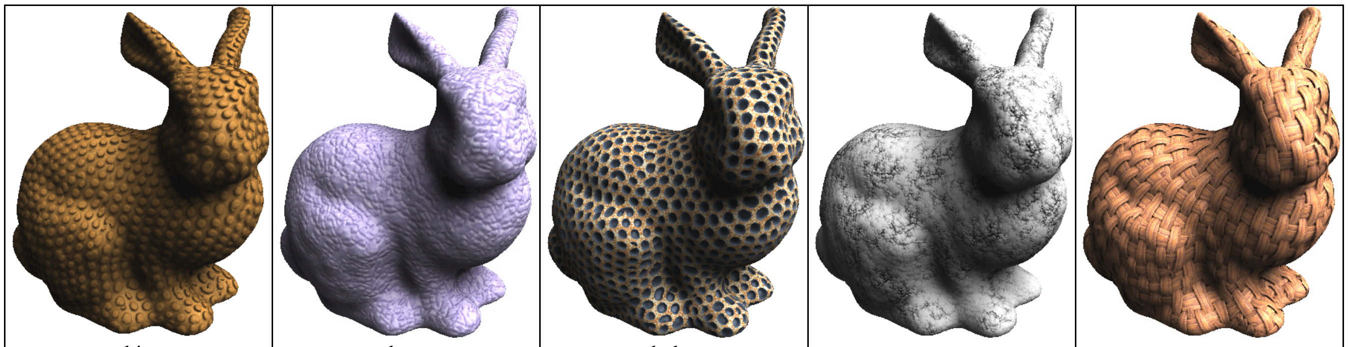
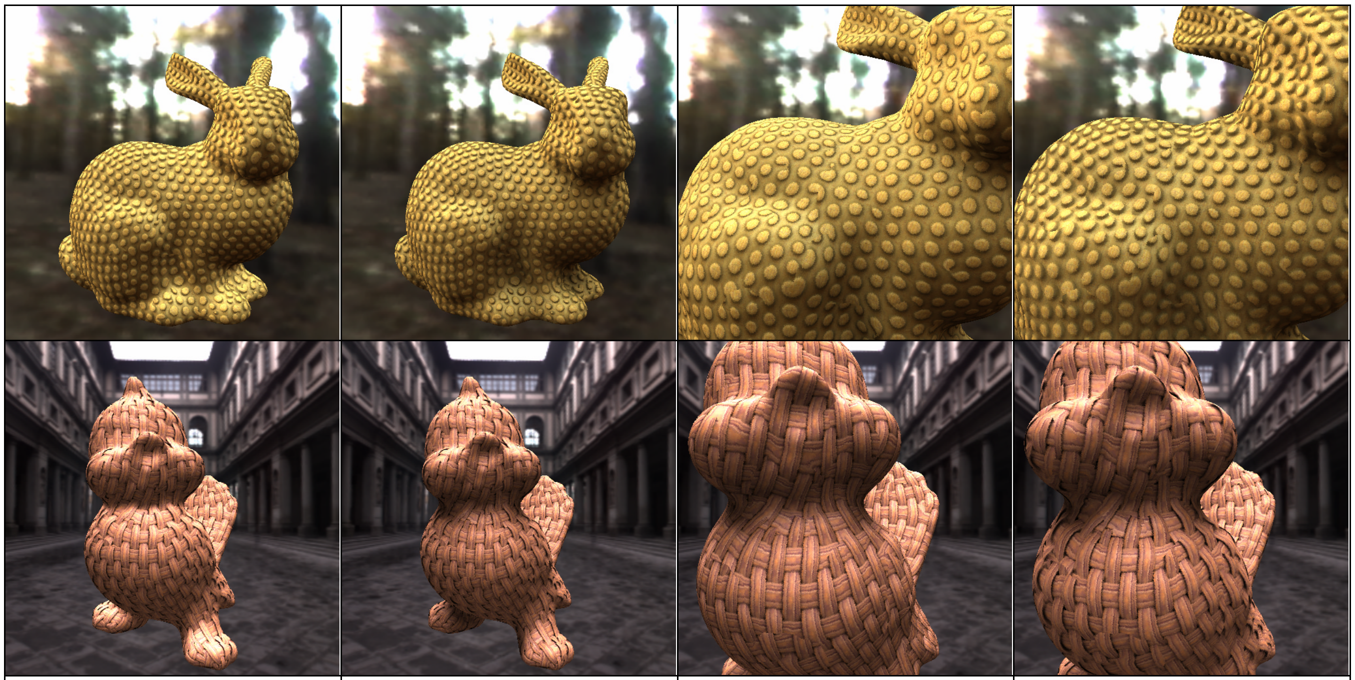


Figure 5: Different RTTs mapped to the same bunny.



(a) No PRT	(b) PRT	(c) Masking off	(d) Masking on

Figure 6: Bi-scale Rendering Effects. The rows are different models in different lighting environments. Columns (a) and (b) compare results with/without macro-scale transport (PRT). Columns (c) and (d) compare results using a zoomed in view with/without view-dependence (masking) in the RTT.