All-Frequency Environment Light Rendering using Rotatable Square Light and Multiple Product Approximation

Kun Xu, Yun-Tao Jia, Shi-Min Hu, Chiew-Lan Tai

Technical Report (TR) 0605 May 2006

Computer Graphics Group, Tsinghua University

Abstract

This paper presents a novel approximation technique that handles object rotation and translation in all-frequency rendering of dynamic scenes. The method can also handle arbitrary isotropic/anisotropic BRDFs, allowing on-the-fly BRDF swapping or editing. The main challenge is in the efficient approximation of the costly rendering integral between light, BRDF and visibility. We decompose the environment map, which is the integral domain, into several subregions (20-40 is sufficient for producing compelling results) so that the rendering integral over the environment map equals the sum of integrals over individual subregions. We introduce a multi-product approximation that approximates the rendering integral in each subregion by the product of separate integrals of light, BRDF and visibility. To fast compute the BRDF integral and visibility integral, we propose a new representation, rotatable square light (RSL), to represent each subregion. Based on the RSL representation, the BRDF and visibility integrals can be approximated quickly using Summed-Area Table (SAT) and Visibility Distance Table (VDT), respectively. Our algorithm can be implemented on current GPU and interactive frame rates can be achieved.

1 Introduction

Fast all-frequency rendering from environment maps is a challenging problem. There has been extensive research addressing this problem in recent years. All-frequency methods [7, 9, 10, 13, 14], based on nonlinear wavelet approximation, have been successfully used to render scenes with realistic shading and shadow effects in all-frequency bands. However, since wavelet basis cannot be easily rotated, [7, 9, 10, 13] have to reproject the environment to wavelet basis in every frame whenever the environment map is rotated, which may cause flickers between frames. The non-rotatability of wavelet also limits the method of Zhou et al. [14] to handle only non-rotating objects of dynamic scenes in all-frequency bands.

To address the limitations caused by non-rotatability of wavelet in all-frequency rendering, we propose a novel approximation method of all-frequency environment lighting. For rendering with environment lighting, the shaded "response" at each vertex of the object is computed by a rendering integral, which integrates three factors, light, BRDF and visibility. Our idea is inspired by this observation: a multi-product integral in a small region can be well approximated with bounded error by the following equation:

$$\int_S f(x)g(x)h(x) \approx \quad |S| \quad \cdot \quad \frac{\int_S f(x)}{|S|} \quad \cdot \quad \frac{\int_S g(x)}{|S|} \quad \cdot \quad \frac{\int_S h(x)}{|S|} \tag{1}$$

Thus, we first decompose the environment map into several subregions, which are guaranteed to be of

small sizes or have small energy. The rendering integral in each subregion can then be approximated by the product of separate integrals of light, BRDF and visibility (Equation 1). The integral of light in the subregion is precomputed during the decomposition of environment map. To fast approximate the integrals of the other two factors, we approximate their common integral domain—a subregion in the environment map—with a rotatable square light (RSL). With each RSL, we utilize Summed-Area Table (SAT) and Visibility Distance Table (VDT) to fast approximate the BRDF integral and the visibility integral, respectively. Instead of approximating the subregions with RSLs in the local frame of object vertex in every frame, we decompose the environment map into subregions and approximate the subregions with RSLs once globally before rendering.

By approximating the rendering integrals in individual subregions with the product of separate integrals, which can be fast approximated, our method can handle all-frequency rendering of static scenes. By adopting the shadow field framework in [14], our method can easily handle dynamic scene rendering and local light sources.

Compared with previous methods, our method make possible several new effects. First, since the integral domain RSL can be easily rotated, we can easily handle all-frequency rendering under environment map rotation and objects rotation in dynamic scenes. Furthermore, our BRDF data is stored separately from objects. This object-separated BRDF representation enables us to easily swap an object's BRDF during rendering or perform on-the-fly BRDF editing. In contrast, in previous all-frequency PRT methods, the BRDF is combined with the object, making it difficult to be edited for scene design.

Our rendering algorithm can be implemented on current graphics hardware, achieving interactive frame rates.

In summary, our main contributions consist of:

- A novel method to approximate multi-product of rendering equation, which enables all-frequency rendering of environment lighting in interactive frame rates through a hardware program.
- Fast all-frequency rendering of dynamic scenes with both object rotation and translation.
- On-the-fly BRDF swapping and editing made possible by our object-separated BRDF representation.

2 Related Work

Environment lighting of static scenes: Sloan et al. [12] introduced a precomputed radiance transfer (PRT) framework for environment lighting. Their method is successful for rendering from low frequency environment maps. They represent the global environment map and the transport function at each vertex in spherical harmonic (SH) basis, which reduces the rendering computation at each vertex to a simple dot product. Ng et al. [9] extended their work to all-frequency bands . Instead of SH, they use nonlinear wavelet basis to represent the environment light and the transport function at each vertex. However, their method is limited to only diffuse objects and glossy objects of fixed views.

Ng et al. [10] subsequently overcome the limitation in their previous work by treating the rendering equation as a triple product of light, view and BRDF factors. They analyze the computation complexity of the triple product, and use the Haar wavelet basis to render realistic images in a few seconds. Based on [9], Liu et al. [7] and Wang et al. [13] factorize BRDF into separate view and light dependent parts, thus they can handle glossy objects in PRT framework with changing views. Liu et al. [7] also applied CPCA to accelerate the rendering and achieve interactive performance. All these methods are wavelet-based, which share a common drawback in that, unlike SH, wavelets representation cannot be easily rotated. In every frame when the environment map rotates, the environment map has to be reprojected to the wavelet representation, which may cause flickers between frames. Another drawback is that, as BRDF is stored in the global frame, it limits the free changing or editing of materials for scene design.

Environment lighting of dynamic scenes: Dynamic scene rendering with PRT is a hard problem because, when objects move, the precomputed light transport function is no longer valid. Mei et al. [8] efficiently rendered shadows of dynamic scenes using precomputed visibility information for each object with respect to uniformly sampled directions. However, their method cannot handle local light sources. Zhou et al. [14] introduced a shadow field framework for rendering dynamic scenes with local light sources. They precompute the shadowing effects, in SH/wavelet basis, at the sample points of each occluder's surrounding space. Each local light's irradiance map is precomputed in a similar way. However, their method cannot handle rotating objects in all frequency bands because the wavelet representation cannot be easily rotated.

Summed-Area Table (SAT): Crow [3] introduced the summed-area table for rapid box filtering (averaging) during texture mapping. Its usage was later extended to volume rendering, image, video processing and so on [5]. As illustrated in Figure 1, by pre-integrating the top-left area of each sample point, the integral in a rectangle area can be rapidly computed with four look-ups:

$$\int_R f(x,y)dxdy = SAT(r,b) - SAT(l,b) - SAT(r,t) + SAT(l,t) \tag{2}$$

Precision problem occurs when SAT is implemented in graphics hardware; the problem can be alle-

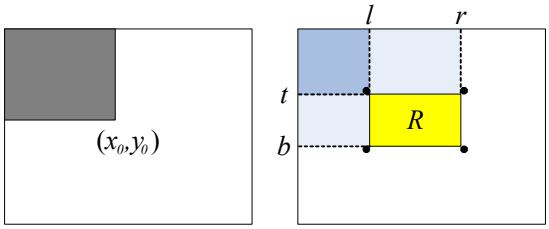

Figure 1: SAT definition and usage. The integral of the rectangle R defined by (l, r, t, b) can be fast computed using four lookups.

viated with techniques in [5].

The remainder of this paper is organized as follows. We outline our rendering framework in Section 3 and describe the algorithm details in Section 4. After presenting the implementation details in Section 5, we analyze the rendering error of our method in Section 6. Results are given in Section 7, and conclusion and future work in Section 8.

3 Rendering Framework

The rendering equation for environment map illumination of dynamic scenes is:

$$B(x, \omega_0) = \int_S L(x,\omega)\rho(x, \omega, \omega_0)V(x, \omega) \prod_j V_{O_j}(x, \omega)d\omega \tag{3}$$

where x is a vertex of a target object, ω and ω_0 are the light and view directions respectively, L is the illumination environment map, V is the self-visibility function of vertex x, $V_{O_j}(x, \omega)$ is the visibility function of an opaque object j at location x, and ρ is the $4D$ reflection function at location x incorporated with the cosine term $\omega \cdot n$.

By decomposing the integral domain into a set of subregions S_i, the rendering equation becomes

$$B(x, \omega_0) = \sum_i \int_{S_i} L(x,\omega)\rho(x, \omega, \omega_0)V(x, \omega) \prod_j V_{O_j}(x, \omega)d\omega \tag{4}$$

With the assumption that each subregion is of small size or has small energy, we approximate the

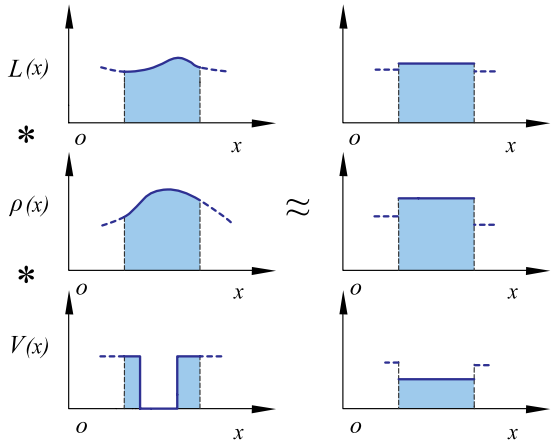

Figure 2: Multi-product approximation. We approximate the integral of the product of light, BRDF and visibility in a subregion by the product of light integral, BRDF integral and visibility integral, as in Equation 5.

render integral in each subregion S_i by the product of separate integrals (see Figure 2)

$$\int_{S_i} L(x,\omega)\rho(x,\omega,\omega_0)V(x,\omega)\prod_j V_{O_j}(x,\omega)d\omega \approx \int_{S_i} L(x,\omega)d\omega \cdot \frac{\int_{S_i}\rho(x,\omega,\omega_0)d\omega}{|S_i|} \cdot \frac{\int_{S_i}V(x,\omega)d\omega}{|S_i|} \cdot \prod_j \frac{\int_{S_i}V_{O_j}(x,\omega)d\omega}{|S_i|} \quad (5)$$

We give an overview of our rendering framework here. The details are explained in the next section. We first globally decompose the environment map into several subregions of arbitrary shapes and approximate each subregion by a *rotatable area light* of square shape. Specifically, we approximate a subregion S_i by an RSL $\langle \vec{c_i}, r_i, I_i \rangle$, where $\vec{c_i}$ is the 3D direction vector indexing the center pixel of the square, r_i is half of the diagonal, and I_i is the average light intensity (Figure 4, more details in next section). Given a desired number of RSLs (which is application dependent), we decompose the environment map by minimizing the sum of the fitting errors between the RSLs and the subregions.

After the light decomposition, the rendering computation is done per-vertex. At each vertex, the rendering integral of the environment map is the sum of the rendering integrals of all the subregions. To approximate the rendering integral in each subregion S_i using Equation 5, we need to calculate the light integral, the BRDF integral and the visibility integrals.

The light integral $\int_{S_i} L(x,\omega)d\omega$ is directly determined as $2r_i^2 I_i$ from the RSL representation. The BRDF integral and the visibility integrals are calculated at runtime in the vertices' local spaces, utilizing a local frame parametrization method that maps a hemisphere to a 2D plane. Figure 3

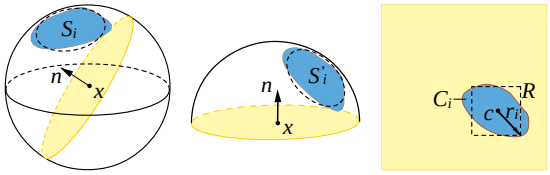

Figure 3: Subregion rotation and projection. A subregion S_i is first rotated from the global frame (left) to the local frame of a vertex x to obtain S_i' (middle), then projected to the 2D parametrization plane (right).

illustrates this process. Suppose we want to integrate over a subregion S_i represented by an RSL $\langle \vec{c_i}, r_i, I_i \rangle$. We first rotate S_i to the local frame of a vertex x to obtain a subregion S_i' with RSL representation $\langle \vec{c_i'}, r_i, I_i \rangle$, where $\vec{c_i'}$ is the rotated $\vec{c_i}$. Next, we project S_i' to the 2D parametrization plane to get a subdomain $C_i$and approximate it by a square R_i with center p_i (corresponding to $\vec{c_i'}$) and edge length $\sqrt{2}r_i$. Then the integral over S_i in the global frame is equal to the integral over C_i in the parametrization domain. An integral over C_i can be approximated by an integral over R_i. Taking BRDF as an example (with ω_0' denoting the view direction ω_0 rotated to the local frame), we have:

$$\frac{1}{|S_i|}\int_{S_i}\rho(x,\omega,\omega_0)d\omega = \frac{1}{|C_i|}\int_{C_i}\rho(p,\omega_0')dp \approx \frac{1}{|R|}\int_{R}\rho(p,\omega_0')dp \tag{6}$$

With this approximation, we fast calculate the BRDF integral using the Summed-Area Table (SAT) method. For the visibility integrals, due to visibility being binary values, instead of using the SAT, we introduce the Visibility Distance Table (VDT) for fast integration.

4 Rendering Algorithm

This section describes the details of our algorithm. Section 4.1 describes how an environment map is globally decomposed into subregions and approximated by RSLs. Sections 4.2 and 4.3 explain how the BRDF integral and the visibility integrals are fast approximated for rendering with each RSL.

4.1 Light Approximation

We approximate environment lighting by decomposing the environment map subregions using a bottom-up merging strategy, with each subregion approximated by a rotatable square light (RSL).

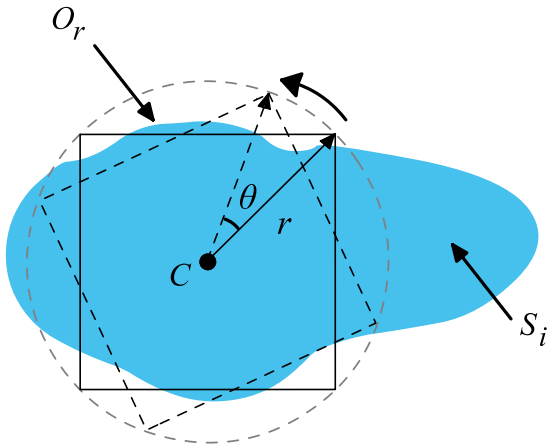

Figure 4: RSL fitting of subregion S_i of arbitrary shape.

To reduce the fitting error, the decomposition process optimizes an energy function defined in terms of the RSL fitting errors.

We first explain how to fit a given subregion S_i with an RSL, and then present how we optimize the RSL fitting error during the environment map decomposition.

4.1.1 Rotatable Square Light (RSL) and RSL Fitting

RSL is a rotatable square light of uniform intensity that best fits a subregion S_i in the environment map. It is defined as $\langle \vec{c}, r, I \rangle$, where $\vec{c}$ is the center direction (3D) of S_i, r is half of the diagonal, and I is the average light intensity (see Figure 4). The fitted RSL can be rotated about its center by any angle θ. The region covered by all possible rotated RSLs forms a circle O_r of radius r.

Fitting a subregion S_i with a RSL is an optimization procedure. We define an extended subregion $S_i' = S_i \cup O_r$. Suppose the light function of S_i is $L(x)$, we define the target light function $T(x)$ and the fitted light function $F(x, \theta)$ in S_i' as

$$T(x) = \begin{cases} L(x) & x \in S_i \\ 0 & x \notin S_i \end{cases} ; \quad F(x,\theta) = \begin{cases} I & x \in O_r(\theta) \\ 0 & x \notin O_r(\theta) \end{cases}$$

where $O_r(\theta)$ is the true region occupied by the RSL rotated by angle θ. The fitting error, $\eta_i^2(\theta)$, which is the optimization function of the RSL fitting, is computed as

$$\eta_i^2(\theta) = \int_{S_i'} (T(x) - F(x,\theta))^2 dx$$

The fitting error can be evaluated at all θ. By minimizing the average of these errors, denoted $E(\eta_i^2(\theta))$, the RSL parameters $\langle \vec{c}, r, I \rangle$ can be determined.

So far, we have only considered light approximation error in RSL fitting. As the rendering error is the ultimate concern, we should also consider the effect of RSL fitting on rendering error. From our mathematical error analysis of the multi-product approximation equation (Equation 5) explained in Section 6, we found that the variance of BRDF and the variance of visibility both influence the error bound of the rendering computation using RSLs. In fact, Agarval et al. [2] prove that, for visibility variance, $E(\sigma_v^2) \propto \sqrt{S}$. Similarly, we experimentally find that, for BRDF variance, $E(\sigma_\rho^2) \propto \sqrt{S}$. Therefore, we add an area penalty term into the objective optimization equation of RSL fitting:

$$\min\{|S_i'|E(\int_{S_i'}(T(x) - F(x,\theta))^2 dx\ )\}. \tag{7}$$

To reduce the search space of RSL fitting, we only optimize $\vec{c}$ and r. The intensity I can be directly derived from $I = \int_{S_i} L(x)dx/2r^2$, which preserves the energy of the subregion. Due to the symmetry of squares, θ only needs to be evaluated in $[0, \pi/2]$. In our experiments, we optimize θ over 0, $\pi/8$, $\pi/4$ and $3\pi/8$.

Using the optimization function in Equation 7, we can solve for the parameters of RSL. The fitting error is used to guide the environment map decomposition, which is described in the next subsection.

4.1.2 Light Decomposition

We decompose the light environment into several subregions by optimizing the sum of the fitting errors by RSLs. This is solved by iteratively merging subregions. At the beginning, we have each pixel in the environment map as a subregion, with a total of $6 \times 32 \times 32$ subregions usually. In each subsequent iteration, we select two subregions and merge them into one subregion. For simplicity, we always select two connected subregions that are in the same face. The criteria of selection is that the merged subregion has the least RSL fitting error among all the candidate subregion pairs in the current iteration. Thus, the number of subregions always decreases by one after each iteration. After thousands of steps, we will obtain a specified count of subregions. This procedure takes about 2 minutes. The pseudo code is shown in Figure 5.

Figure 6 shows two examples of light approximation using RSLs, for high- and low- frequency environment maps. The middle column shows the resulting subregions, and the right column shows the approximated RSLs for the subregions. In our experiments, we found that $20 \sim 40$ RSLs are enough to give compelling rendering results for most kinds of environment maps.

```
Initialize the subregion set S = {s_i | 0 ≤ i < m};
error array error[m];
index array index[m];

While (m>1)
    For i=0 to m
        Try to merge s_i with neighbor subregions,
        and compute RSL fitting errors;
        Record min error in error[i] and neighbor's
        index in index[i];
        If no neighbor, error[i] = ∞, index[i] = -1;
    End For
    Find subregion s_j with minimum fitting error error[j];
    If (index[j] == -1) break;
    Merge s_j and s_index[j] to obtain a new subregion s;
    set s_j = s;
    s_index[j] = s_m-1;
    m--;
End While
```

Figure 5: Pseudo code of light decomposition.

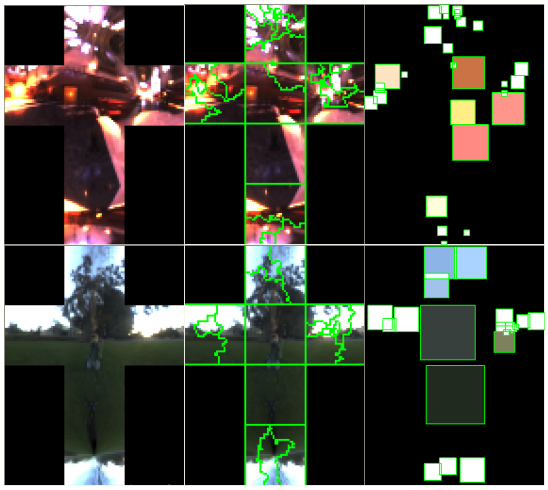

Figure 6: Light approximation using RSLs. Left column shows the original environment maps; middle shows the subregions bounded by green lines; right shows the fitted RSLs. Top and bottom rows illustrate the results for high- and low- frequency light environments respectively.

4.2 BRDF Integral

As described in the framework section, with Equation 6, the BRDF integral in a subregion S_i is approximated by an BRDF integral over a square R in the 2D parametrization plane:

$$\frac{1}{|S_i|}\int_{S_i}\rho(\omega,\omega_0)d\omega \approx \frac{1}{|R|}\int_{R}\rho(p,\omega_0')dp \tag{8}$$

We pre-integrate a BRDF function into an SAT, then a BRDF integral over any rectangle can be fast computed using only four lookups of the SAT. Thus, we perform the BRDF integral computation in two steps: precomputation step and rendering step. In the precomputation step, for each view direction ω_0, we pre-integrate the SAT of the BRDF function of p (correspond to light direction ω) as $I(p,\omega_0)=\int_C\rho(p,\omega_0)dp$. Then, in the rendering step, the integral over the square R is fast calculated by four SAT look-ups; interpolation is done for intermediate view directions.

For an anisotropic BRDF, we have to tabulate the view direction ω_0 over the whole hemisphere which is 2D, therefore the whole BRDF SAT dataset is 4D. For an isotropic BRDF, we only need to tabulate the polar angle of the view direction ω_0, therefore the whole BRDF SAT dataset is 3D.

Because the BRDF integral is independent of the models and vertices, several benefits can be derived. Firstly, we can on-the-fly swap or edit the BRDF of a model. For analytic BRDFs, given the new parameters, the BRDF SAT data (a 3D/4D table) can be regenerated on the fly, achieving interactive editing of BRDF. Secondly, we only need to store a 3D/4D dataset for an isotropic/anisotropic BRDF. In contrast, previous methods like [10] define their BRDF in the global frame, requiring 6D BRDF data storage. Thirdly, local deformable shading effects can be easily handled. Our method can handle deformable shading as the BRDF integral is computed in local frame; visibility integrals can be ignored for local shading.

4.3 Visibility Integral

Similar to the BRDF integral, intuitively, visibility integrals can be fast approximated using SATs. However, because visibility data are binary values, we propose a new method called Visibility Distance Table (VDT), which is more efficient for visibility integral approximation and visibility data compression than the SAT.

As shown in Figure 7, given a 2D visibility map $v(p)$, which is a function of point p in the hemisphere parametrization domain, the VDT is defined as $D(p)=d(p)*sign(p)$, where $d(p)$ is the nearest distance from point p to the binary-change boundary of $v(p)$, and $sign(p)=1$ if $v(p)=1$, otherwise $sign(p)=-1$. We use the method described in [4] to generate a visibility distance table from the

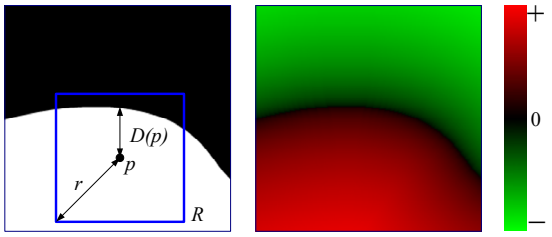

Figure 7: Visibility Distance Table (VDT) definition. Left is a visibility map, right is the corresponding Visibility Distance Table. The distance is normalized.

given visibility map. The integral of $v(p)$ over the square R (center p, half diagonal r) can be simply approximated as

$$\frac{1}{|R|}\int_R v(p)dp \approx min(1, max(0, F(p)) \tag{9}$$

where $F(p) = \frac{\frac{\sqrt{2}}{2}r + D(p)}{\sqrt{2}r}$ approximates the percentage of points with value 1 in the integral square.

VDT has several advantages. First, only one look-up is needed for visibility integral approximation, which is four times faster than looking up SAT; secondly, unlike SAT, VDT has no precision problem in graphics hardware; thirdly, VDT is a continuous signal, which is more suitable for compression while giving fewer artifacts.

Using VDT to look up an integral is only accurate when the visibility boundary is a straight line. Nevertheless, our experiments show that results are acceptable in most cases. There is a tradeoff between performance and accuracy when choosing VDT or SAT.

4.3.1 Self Visibility

As described in Section 3, self-visibility integral in a subregion S_i can be approximated by an integral over the square R in the 2D parametrization plane:

$$\frac{1}{|S_i|}\int_{S_i} V(x,\omega)d\omega \approx \frac{1}{|R|}\int_R V(x,p)dp \tag{10}$$

The integral can be fast approximated using VDT in two steps: pre-computation step and rendering step. In the pre-computation, we ray-trace to compute the visibility map $V(x,p)$ at each vertex x in its local frame, and generate the corresponding Visibility Distance Table $D(x,p)$. We call all these maps at vertex x collectively as the Self-Visibility Distance Field (SVDF). Then, during rendering, for each subregion S_i, the integral is approximated by Equation 10 with a square R. With the

center and the half-diagonal of R, we further approximate the integral using a VDT according to Equation 9.

4.3.2 Occluder Visibility

Similar to self-visibility integral, the occluder visibility integral is also approximated by integrating over a square R in the 2D parametrization plane:

$$\prod_i \frac{1}{|S_i|} \int_{S_i} V_{O_i}(x, \omega) d\omega \approx \prod_i \frac{1}{|R_i|} \int_{R_i} V_{O_i}(x, p) dp \tag{11}$$

Here we adopt the shadow field framework of [14], and again approximate the integral in two steps. For the pre-computation step, we adopt the sampling method of object occlusion field (OOF) from [14], which stores a 2D visibility data for each sample point in the 3D surrounding space of the object. However, our pre-computation step differ from theirs in two aspects. First, at each sampled point q around an object O, we capture the visibility map only on the hemisphere defined by the direction from q to the center of O, rather than the visibility map in the global frame. Second, we store a VDT instead of a visibility map at each sampled point. The VDTs at all sampled points are collectively referred to as the Occluder Visibility Distance Field (OVDF). In the rendering step, as [14], we approximate each occluder integral on the VDTs of the sample points using Equation 9; the VDT at an intermediate point is approximated by a trilinear interpolation of the eight nearest samples.

5 Implementation

We have implemented our rendering algorithm on graphics hardware. This section presents the implementation details, including hemisphere parametrization, data compression and the shader program in GPU.

We use a hemisphere parametrization that maps from a unit hemisphere to a unit disk: define $p(\theta, \phi) : [0, \frac{\pi}{2}] \times [0, 2\pi] \rightarrow [-1, 1] \times [-1, 1]$ as

$$p = \frac{\sin\theta}{\sqrt{1 + \cos\theta}} (\sin\phi, \cos\phi).$$

We have also tested cube map parametrization and the hemisphere parametrization in [6], but we find that our parametrization gives the least distortion. Cube map is not a uniform parametrization,

and the size of a region changes when it is projected from a sphere to a cube map, leading to noises. The hemisphere parametrization in [6] produces artifacts when a region crosses the diagonal of the unit disk.

5.1 Data Compression

Let ω_0 and ω denote the view and light directions respectively, and let θ_0 denote the polar angle of the view direction. For an isotropic BRDF $\rho(\omega, \theta_0)$, ω is sampled at 32*32 directions of the hemisphere parametrization, and θ_0 is sampled at 32 angles. For an anisotropic BRDF $\rho(\omega, \omega_0)$, both ω and ω_0 are sampled at 32*32 directions. We do not do compression for BRDF data.

The 4D Self-Visibility Distance Field (SVDF) is precomputed with a sampling rate of 32*32*N, where 32*32 is the size of the VDT at each vertex, and N is the vertex count. The 5D Occluder Visibility Distance Field (OVDF) is precomputed at 32*32*(6*32*32)*16, where 32*32 is the VDT size, (6*32*32)*16 because, like OOF in [14], we use a cube sampling of 6*32*32 to sample the space around each object on 16 different concentric spheres, with radii ranging from $0.4r$ to $6r$, where r is the radius of the bounding sphere. As a result, the data of the SVDF for a 40k vertex mesh is about 160M (32FP), and the data of the OVDF is about 384M. Both of them need to be compressed before putting into the GPU.

For SVDF compression, we use the clustered PCA (CPCA) method. Take a 40k vertex model for example, with 256 clusters and 8 eigen-vectors for each cluster, a compression ratio of about 1:17(9M) gives a good result.

We compress OVDF as follows. For each concentric sampling sphere (using cube map sampling), we split all the six faces of the cube map into 2*2 segments, and obtain 24 segments on each sphere, like in [7]. We use 16 concentric sampling spheres, resulting in a total of 24*16=384 segments. Each segment is compressed using PCA. With 8 eigen-vectors per segment, a compression ratio of about 1:26(15M) gives a good result.

We pack BRDF, SVDM and OVDF data into textures to load into the GPU. For higher accuracy, we use 16FP textures instead of 8BP. For BRDFs, we pack both isotropic and anisotropic BRDFs into 3D textures. For the compressed SVDF and OVDF, we pack the eigen-values and the cluster indices of the vertices into 2D textures. Eigen-vectors of SVDF and OVDF are packed into 2D and 3D textures, respectively.

We use one more texture to store all the RSLs, which are updated after rotation of the scene in each frame.

```
For each vertex p
    Initialize color Bp = 0;
    For each RSL <c,s,I>
        Project light dir c to p's local frame, get c';
        Project view dir v to p's local frame, get v';
        lookup SVDF with (c', s), get integral Vs;
        lookup BRDF SAT with (c',s,v'), get integral Vq;
        T = I*Vs*Vq/s;
        For each occluder J
            Project c to the local frame of sample point
            p of occluder J(oriented to J's center), get c'';
            Lookup J's OVDF with (c'',s,p), get integral Voj;
            T *= Voj /s;
        End for
        Bp += T;
    End For
End For
```

Figure 8: Pseudo code of pixel shading program.

5.2 Shader program in GPU

Our algorithm is a per-vertex rendering method, however we utilize the render-to-vertex-array technique and perform the rendering in two steps. In the first step, we pack the object vertices into a 2D rectangle, with one vertex on the object corresponding to one pixel in the rectangle, and use a pixel shader program to calculate the color for each pixel. The color is copied from the frame buffer to the vertex array using OpenGL extension *pixel buffer object (PBO)*. In this step, the vertex attributes are needed for color calculation for each vertex, so we pack the position, normal and tangent direction of the vertices into a 2D texture. In the second step, we use the OpenGL extension *vertex buffer object (VBO)* to render the scene with the color calculated in the first step. The pseudo code of pixel shader is shown in Figure 8.

In the shader program, several textures are looked up for each vertex, such as attribute texture, BRDF texture, eigen-value, cluster index and eigen-vector textures of SVDF and OVDF. As a result, texture fetching is the bottleneck. We use occluder culling to accelerate it. To do this, we use multi-pass rendering in the first step, with one pass for each light (RSL). The resulting images of each pass are blended together to generate a final image. Then, before each pass, we determine the occluders that can be neglected in that pass using CPU, and only send the remaining occluders to GPU.

6 Error Analysis

There are two main approximations in our method. One is the multi-product approximation, which approximates the rendering integral with a product of separate integrals. The other is the approxi-

mation of the separate integrals of BRDF and visibility. We will analyze the errors brought about by these two approximations in 6.1 and 6.2, and compare our results with ground truth and with the results of Ng et al. [10] in 6.3.

6.1 Error of Multi-product Approximation

We first give an error bound of the multi-product approximation mathematically. Given a function f defined on a region S, denote the expectation u_f, the variance σ_f^2 and the 3rd central moment m_f^3 of function f as: $u_f = \frac{1}{|S|}\int_S f(\omega)d\omega$, $\sigma_f^2 = \frac{1}{|S|}\int_S (f(\omega) - u_f)^2 d\omega$, and $m_f^3 = \frac{1}{|S|}\int_S (f(\omega) - u_f)^3 d\omega$. Denote $V'(x,\omega) = V(x,\omega)\prod_j V_{O_j}(x,\omega)$ in Equation 5, we can rewrite Equation 5 as

$$\int_{S_i} L(x,\omega)V'(x,\omega)\rho(x,\omega,\omega_o)d\omega \approx |S_i|\, u_L u_{V'} u_\rho \tag{12}$$

The error of Equation 5 is then bounded by

$$\begin{aligned} &\left|\int_{S_i} L(x,\omega)V'(x,\omega)\rho(x,\omega,\omega_o)d\omega - |S_i|\, u_L u_{V'} u_\rho\right| \\ &\leq |S_i|\,(u_L\sigma_{V'}\sigma_\rho + u_{V'}\sigma_\rho\sigma_L + u_\rho\sigma_L\sigma_{V'} + m_L m_{V'} m_\rho) \end{aligned} \tag{13}$$

This can be mathematically proved using *Hölder's Sum Inequality* [1].(To reviewer: if permitted, we can provide the proof.)

From this equation, we could know that the error of the multi-product approximation is bounded by the intensity and variance of light, BRDF and visibility. Light variance is already considered in RSL fitting and is minimized in light decomposition in Section 4. From [2], we know that the average variance of visibility $E(\sigma_v^2) \propto \sqrt{S}$, also we experimentally find that the average variance of BRDF $E(\sigma_\rho^2) \propto \sqrt{S}$. Since we add the region size as a penalty in RSL fitting during environment map decomposition, this strategy guarantees that the sizes of regions with high light intensity will be small, so the average variance of visiblity and BRDF will be small, tending to minimize the overall bounding error.

6.2 Error of Separate Integrals

In order to fast compute BRDF and visibility integrals, we use a square R in parametrization plane to approximate the projected subregion C. The mismatch between R and C leads to error in the separate integrals. The error is determined by two terms: one is the mismatch area percentage between R and C, the other is the variance of BRDF or visibility in the region. The integral error

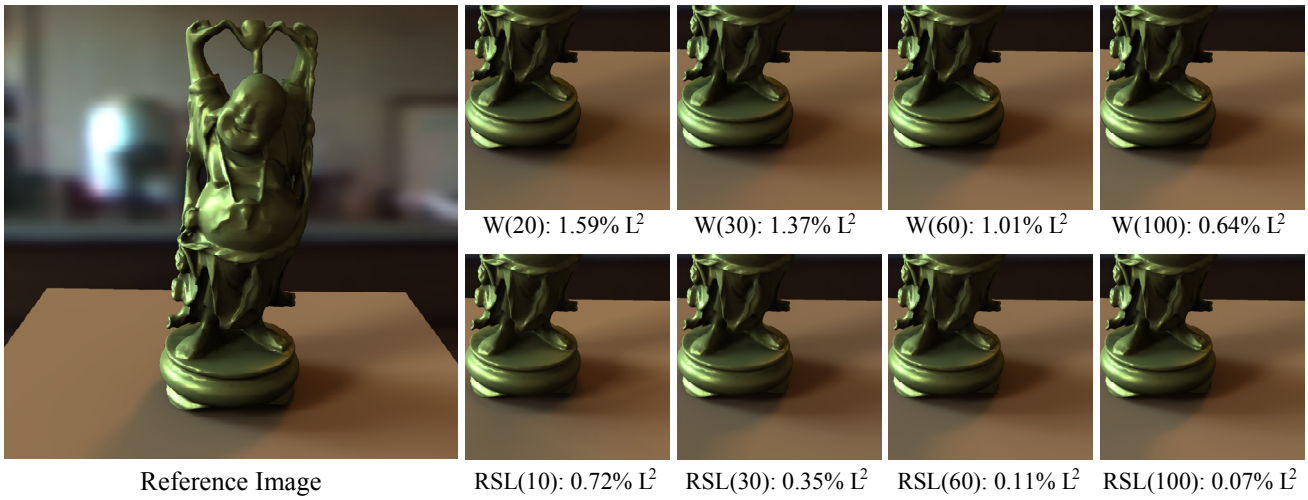

Figure 9: Rendering of a buddha scene in Kitchen. Please notice the difference of the shadow boundaries.

will be small if either term is small. To give a better insight, consider rotating and projecting a subregion that is only a pixel to the parametrization plane. The mismatch percentage may still be as large as 10%-20%, but the error would be nearly zero, because the variance of BRDF and visibility in a small area is nearly zero.

The RSL fitting equation has already taken the mismatch percentage into consideration. In addition, since our light decomposition strategy guarantees that the sizes of regions with high light intensity are small, we can conclude that the average variance of BRDF and visibility is small for a high light intensity region.

6.3 Error in Rendered Images

In Figure 9 and 10, we compare our results with the ground truth (obtained by ray tracing) and the results of the area-weighted wavelet method [9] for both low- and high-frequency environment maps. 10, 30, 60, 100 RSLs are used in our method, and 20, 30, 60, 100 wavelet terms(each channel) are used for the wavelet method [9]. L^2 error are measured for both methods. From the figures, we can see that the rendering error of our method under all-frequency environment lighting is small. Results with 100 RSLs are almost accurate as ground truth; with 30 RSLs, the accuracy is as that of wavelets with 100 terms. Our method is more accurate than wavelet methods because RSLs can better approximate the environment light. RSL is flexible and can freely fit high intensity regions, while wavelet basis has to be aligned to the quad-tree of faces of cube map and cannot freely fit the high intensity regions.

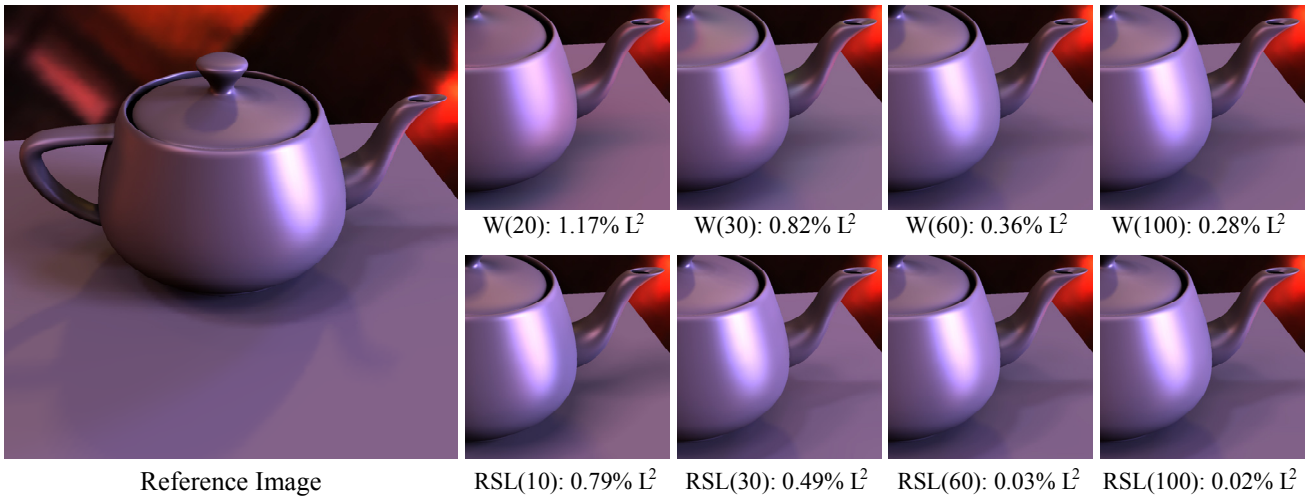

Figure 10: Rendering of a teapot scene in Grace Cathedral. Please notice the difference of the glossy slope and the shadow boundaries.

Scene	Vertices	FPS	Num. of RSLs
Robot	60k	10.8	30
Kitchen	80k	9.9	30
Buddha (static scene)	45k	27	30

Table 1: Result performance.

7 Results

The performance of different scenes is shown in Table 1. We apply occluder culling to dynamic scenes. The performance is reported on a Pentium IV 3.2GHz PC with a Nvidia GeForce 7800GT 256MB graphics card. Interactive frame rates are achieved for large dynamic scenes under all-frequency environment lighting, and realtime frame rates are achieved for static scenes under all-frequency environment lighting, which is much faster than previous methods.

Figure 11 compares the result of SVDF under different levels of compression. Figure 12 compares the result of OVDF under different levels of compression. Figure 13 and 14 show some rendering results of the robot scene and the kitchen scene. Figure 15 shows rendering results of different BRDFs; we use the BRDFs from [11]. Figure 16 shows results of local light illumination and local deformable shading.

8 Conclusion and Future Work

We have proposed a new approximation method for fast all-frequency rendering. Our method can handle both object rotation and translation of dynamic scenes in all-frequency environment lighting, and it enables on-the-fly BRDF editing and swapping. Furthermore, it can be used in applications of local light illumination and local deformable shading.

By first decomposing the whole environment light to a few subregions represented by RSLs, the whole rendering integral can be approximated by the sum of products of separate integrals that integrate light, BRDF and visibility respectively. In individual subregions represented with RSLs, BRDF integral can be fast approximated by SAT and visibility integral can be fast approximated by VDT. $20 \sim 40$ RSLs are generally enough for producing compelling results and interactive performance is achieved for large scenes using a hardware program,

As future work, we would like to enhance to an error-driven method based on rendering error analysis. We will also try to incorporate indirect lighting and interreflection into our method. By computing and storing the SAT of spatial-variant BRDF and BTF. our object-separated BRDF representation can also be extended to these applications.

References

[1] Milton Abramowitz and Irene A. Stegun, *Handbook of mathematical functions, with formulas, graphs, and mathematical tables*, Dover, NewYork, NY, USA, 1972.

[2] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen, *Structured importance sampling of environment maps*, ACM Trans. Graph. **22** (2003), no. 3, 605–612.

[3] Franklin C. Crow, *Summed-area tables for texture mapping*, SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press, 1984, pp. 207–212.

[4] Per-Erik Danielsson, *Euclidean distance mapping.*, Computer Graphics and Image Processing, 1980, pp. 227–248.

[5] Justin Hensley, Thorsten Scheuermann, Greg Coombe, Montek Singh, and Anselmo Lastra, *Fast summed-area table generation and its applications*, Computer Graphics Forum **24** (2005), no. 3, 547–555.

[6] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum, *Synthesis and rendering of bidirectional texture functions on arbitrary surfaces.*, IEEE Trans. Vis. Comput. Graph. **10** (2004), no. 3, 278–289.

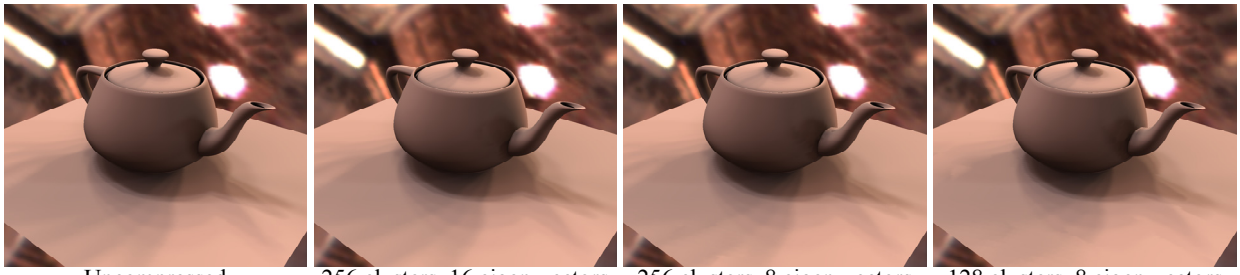

Figure 11: Self visibility distance field(SVDF) compress result. From left to right: uncompressed, using 256 clusters and 16 eigen-vectors, using 256 clusters and 8 eigen-vectors, using 128 clusters and 8 eigen-vectors.

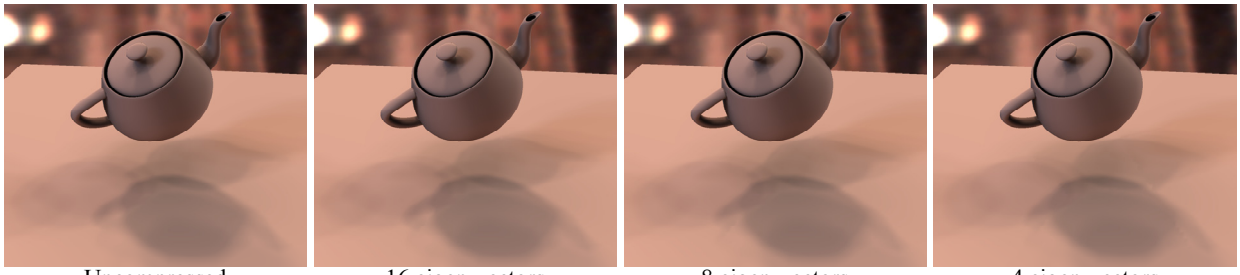

Figure 12: Occluder visibility distance field(OVDF) compress result. From left to right: uncompressed, using 16 eigen-vectors, using 8 eigen-vectors, using 4 eigen-vectors.

[7] Xinguo Liu, Peter-Pike J. Sloan, Heung-Yeung Shum, and John Snyder, *All-frequency precomputed radiance transfer for glossy objects.*, Rendering Techniques, 2004, pp. 337–344.

[8] Chunhui Mei, Jiaoying Shi, and Fuli Wu, *Rendering with spherical radiance transport maps.*, Comput. Graph. Forum **23** (2004), no. 3, 281–290.

[9] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan, *All-frequency shadows using non-linear wavelet lighting approximation*, ACM Trans. Graph. **22** (2003), no. 3, 376–381.

[10] _____, *Triple product wavelet integrals for all-frequency relighting*, ACM Trans. Graph. **23** (2004), no. 3, 477–487.

[11] Addy Ngan, Frédo Durand, and Wojciech Matusik, *Experimental analysis of brdf models.*, Rendering Techniques, 2005, pp. 117–126.

[12] Peter-Pike Sloan, Jan Kautz, and John Snyder, *Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments*, SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press, 2002, pp. 527–536.

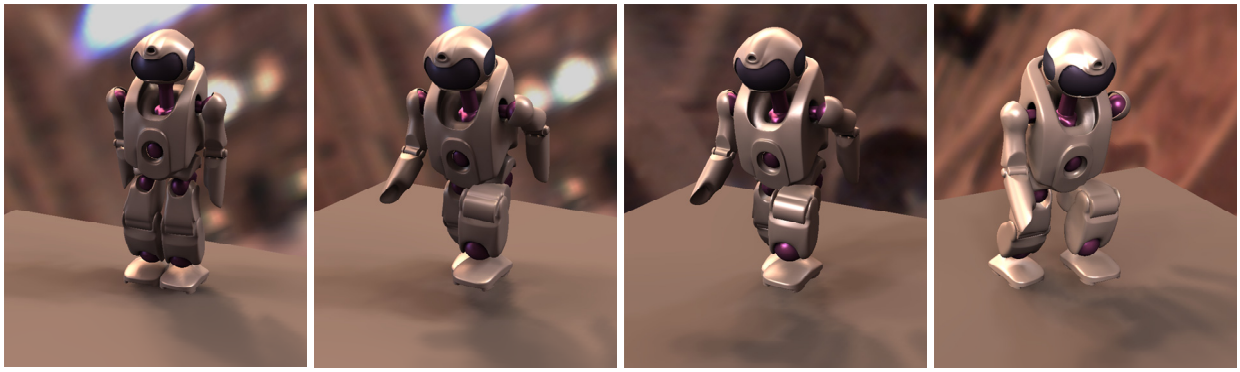

Figure 13: Rendering results of the robot scene under dynamic environment map and changing view. The robot is composed of 12 components.

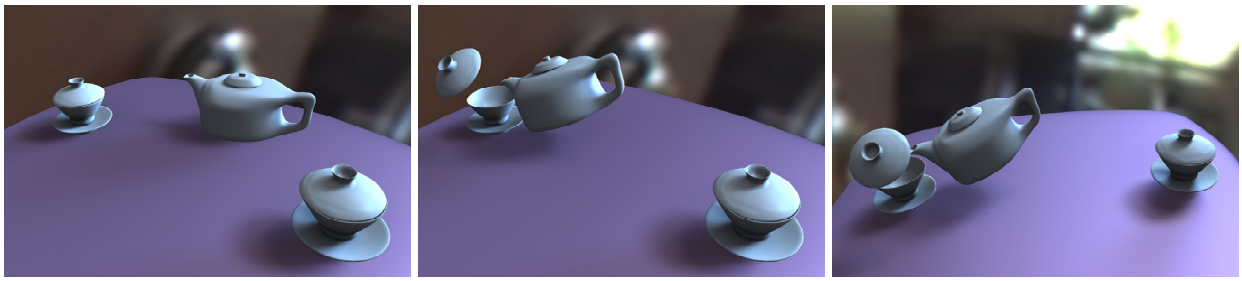

Figure 14: Rendering results of the kitchen scene under dynamic environment map and changing view. The scene is composed of 5 components.

[13] Rui Wang, John Tran, and David P. Luebke, *All-frequency relighting of non-diffuse objects using separable brdf approximation.*, Rendering Techniques, 2004, pp. 345–354.

[14] Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum, *Precomputed shadow fields for dynamic scenes*, ACM Trans. Graph. **24** (2005), no. 3, 1196–1201.

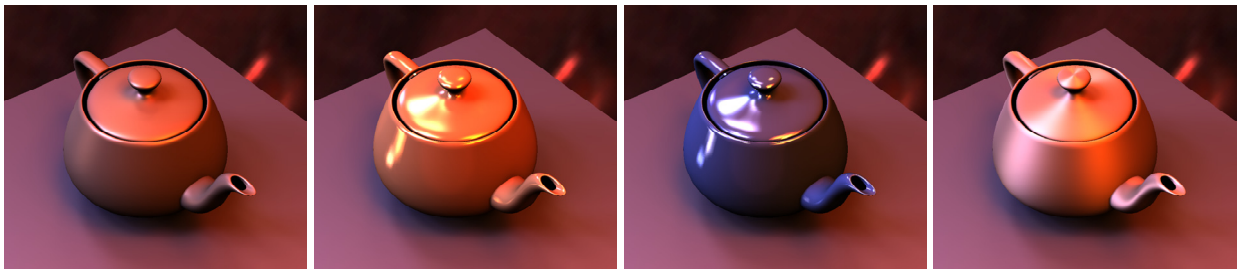

Figure 15: Rendering results of different BRDFs. From left to right, the BRDF models we used are: steel Phong, bronze Cook-Torrance, metal Ward Isotropic, steel Ward Anisotropic.

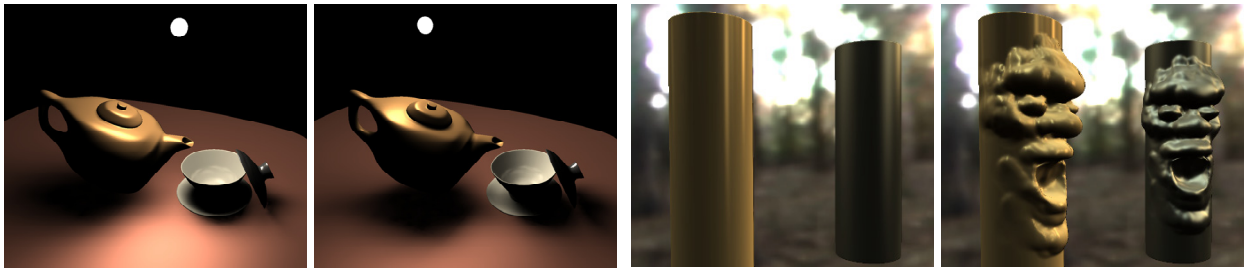

Figure 16: Local light illumination and local deformable shading. The left two figures illustrate the kitchen scene under a local light source. The right two figures show local deformable shading under environment map.