EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Precomputed Radiance Transfer Field for Rendering Interreflections in Dynamic Scenes

Minghao Pan Rui Wang Xinguo Liu[†] Qunsheng Peng Hujun Bao

State Key Lab of CAD&CG, Zhejiang University

Abstract

In this paper, we introduce a new representation – radiance transfer fields (RTF) – for rendering interreflections in dynamic scenes under low frequency illumination. The RTF describes the radiance transferred by an individual object to its surrounding space as a function of the incident radiance. An important property of RTF is its independence of the scene configuration, enabling interreflection computation in dynamic scenes. Secondly, RTFs naturally fit in with the rendering framework of precomputed shadow fields, incurring negligible cost to add interreflection effects. In addition, RTFs can be used to compute interreflections for both diffuse and glossy objects. We also show that RTF data can be highly compressed by clustered principal component analysis (CPCA), which not only reduces the memory cost but also accelerates rendering. Finally, we present some experimental results demonstrating our techniques.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1. Introduction

Interreflections are critical for realistic image generation. Many physically based rendering techniques compute interreflections using Monte Carlo ray tracing or radiosity to simulate light transportation. These methods are usually very slow, even when acceleration techniques are used Dynamic scenes are even more challenging, because the Objects' movement renders most acceleration data structures invalid.

Precomputed Radiance Transfer (PRT) encapsulates a family of recently developed techniques that separate image generation into pre-rendering and rendering phases, storing intermediate light transportation results in radiance transfer functions [SKS02]. The pre-rendering phase is executed only once and may involve complex operations such as casting millions of rays and building complex data structures. Therefore, The pre-rendering phase enables PRT to render realistic images with global illumination effects, such as soft shadows and interreflections. The rendering phase is executed repeatedly and must be of limited complexity if rendering is to be done at interactive or real-time rates. Modern GPUs with powerful, programmable pixel shaders have created the opportunity to rebalance the workload between the pre-rendering and rendering phases.

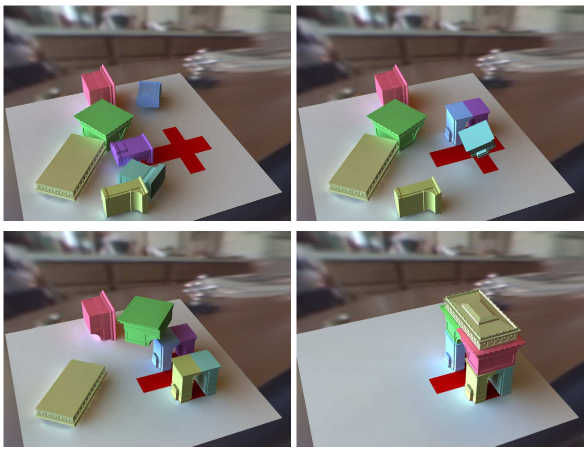
Figure 1: *Some screen shots of a toy bricks game.*

However, most PRT techniques are limited to static scenes

[†] Corresponding author: Xinguo Liu and Hujun Bao. E-Mails: {panminghao, rwang, xgliu, peng, bao}@cad.zju.edu.cn

or precomputed sequences because the offline simulation is for fa ixed scene configuration. Naïvely extending the PRT techniques to dynamic scenes would incur massive amounts of data, making precomputation, storage and relighting intractable. The work of shadow fields proposed a new paradigm for rendering soft shadows cast by moving objects. But it cannot render interreflections – one of the most important global illumination effects. To generate images with interreflections, we need to integrate not only the contribution directly from light sources but also indirect lights from neighboring objects.

In this paper, we introduce precomputed *radiance transfer fields* (RTF) – a new representation for rendering interreflections in dynamic scenes. RTFs describe the transferred radiance of an individual object in its surrounding space as a function of the incident light. RTF is 7-dimensional, two dimensions for input light, two for transferred radiance and three for the transferred location. To make our idea practical, we focus on *low frequency* illumination and transfer, and represent radiance in the spherical harmonics (SH) basis. In contrast to *all frequency* PRT [NRH03], low frequency RTF has more spatial coherence, which enables efficient compression using, for example, the *clustered principal component analysis* (CPCA) [SHHS03] method.

The RTF is fully determined by the object itself and is independent of the scene configuration, which is critical for computing interreflection in dynamic scenes. For any object location and incident radiance, we can look-up the RTF for any receiver point to determine the indirect light reflected by the object.

However, an object's incident radiance may contain indirect lighting from neighboring objects, to compute it will recursively invoke interreflection computations. Moreover the incident radiance may vary across the object due to shadowing, interreflection and local light sources. To capture such variations with reasonable and acceptable cost, we propose a proxy based approximation. This approximation is very critical for enabling fast interreflection rendering as shown in Section 5.

RTF naturally fits in with the rendering framework of precomputed shadow fields [ZHL*05], as shown in **Algorithm** 1. We divide the rendering phase into two stages. The first stage invokes a number of iterations to determine incident radiance of the proxies including the interreflections, while the second stage computes the final shading for all objects. As shown in Section 5, the extra cost for computing interreflections is very small compared with the cost of the shadow field rendering.

2. Related Work

There are many approaches for generating realistic images. In the following, we will focus only on interactive rendering techniques using precomputed light transport. Traditional global illumination methods based on ray tracing and radiosity are beyond the scope of this paper.

Sloan et al. introduced precomputed radiance transfer (PRT) for realtime rendering under dynamic low-frequency environment illumination [SKS02]. They achieved compact representation of radiance transfer functions by using a low order spherical harmonics (SH) light basis. The radiance transfer data can be further compressed by CPCA [SHHS03]. Since this early work, PRT has attracted much attention due to its potential in realtime realistic rendering applications.

Light basis To handle arbitrary environment lighting, Ng et al. introduced wavelet bases for the precomputed radiance transfer functions [NRH03]. Recently, Tsai et al. introduced a new approximation using a spherical radial basis [TS06]. To handle local point light sources in real time lighting design, Kristensen et al. proposed a PRT technique to compute indirect illumination effects [KAMJ05] for static scenes. Green et al. used Gaussian functions to approximate transport data [GKMD06]. Hašan et al. also used a wavelet basis to encode direct-to-indirect transfer [HPB06].

Illumination distribution approximation Sloan et al. [SKS02] suggested using iterated closest point (ICP) to determine a set of sample points and then using the sampled incident light at these points to approximate local lighting variation over the whole object. Annen et al. introduced Spherical Harmonics gradients to give a better approximation with fewer samples [AKDS04]. Iwasaki et al. uses Fourier series to represent illumination distribution of diffuse surfaces [IDT*06].

Surface materials In low frequency environment illumination, it is relatively easy to render diffuse and glossy objects [SKS02, KSS02, LK03], translucent objects [SHHS03, HBV03], and complex mesostructures [SLSS03]. Sunshine-Hill and Faloutsos proposed offset transfer maps to enable fast GPU rendering of diffuse and translucent scenes [SHF06]. However, when moving to all-frequency environment illumination, many practical issues on transfer matrices make rendering the above effects very challenging. To render all frequency glossy effects, Ng et al. designed a triple product wavelet integration technique [NRH04], and Liu et al. and Wang et al. proposed to factorize the transfer matrix by special BRDF factorizations [LSSS04, WTL04]. A similar factorization method is applied to subsurface scattering phase functions for rendering translucent objects [WTL05].

Dynamic scenes The most challenging task for PRT techniques is to handle dynamic scenes and deformable objects, because object's movement and deformation make the precomputed transfer functions invalid. James et al. simply precompute the transfer functions for many key frames [JF03], which is however impractical for general object movements. Kautz et al. proposed to compute the visibility function on-the-fly by a novel hemispherical rasterization tech-

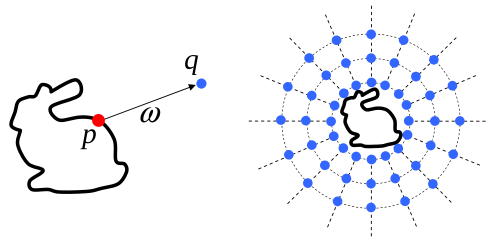

Figure 2: *Interreflection at a neighboring point (Left) and the interreflection transfer field (Right).*

nique [KLA04]. This method doesn't generalize well to scenes with many objects. The neighborhood-transfer presented by Sloan et al. [SKS02] can record reflections around an object. However, it assumes uniform illumination over the object and, therefore, does not support dynamic scenes with local light sources.

Precomputed shadow fields [ZHL*05] can interactively render soft shadows for general dynamic scenes of rigid objects. Precomputed shadow fields were further optimized and implementated on a GPU by Tamura et al. [TJCN06]. For general deformable objects, Ren et al. proposed SH exponentiation technology [RWS*06]. For dynamic character skins, Kirk et al. presented a single-pass hardware accelerated method to reconstruct compressed ambient occlusion values in real-time [KA07]. However, all of these methods can only render shadows from direct illumination, ignoring interreflection effects.

Wyman et al. proposed an interactive caustic rendering technique [WHS04] using precomputed caustic intensity values in the surrounding space of the objects. This technique has some similarity to RTF, but it doesn't include directional information and ignores global shadowing effects between objects. Iwasaki et al. proposed a PRT method for diffuse interreflections [IDT*06] using a transfer function of vertex outgoing irradiance whose distribution is represented by Fourier series. Unlike their method, we use a transfer function of incident radiance which enables support of non-diffuse materials. Also, our proxy scheme can yield efficient interreflection computation because we only compute illumination at proxies.

3. Radiance Transfer Field (RTF)

In this section, we give the definition of the *radiance transfer field* (RTF). We first formulate the problem with respect to distant environment lighting, then introduce a proxy based approximation method to address local lighting.

3.1. Interreflection Transfer Function (ITF)

Let q be a point in the neighboring space of an object O. As shown in Figure 2, the reflected light incident at q in direction ω comes from a point p on the object O. Thus it can be expressed as an integral on an unit sphere of incoming directions:

$$I_q(\omega) = \int L_p(\mathbf{s})\ f_p(\omega,\mathbf{s})\ (\mathbf{s}\cdot\mathbf{n}_p)_+ \mathrm{d}\mathbf{s}, \tag{1}$$

where $\mathbf{n}_p$ is the surface normal at point p, $(\ )_+$ denotes clamping negative values to zero, f_p is the *bidirectional reflectance distribution function* (BRDF) of point p, and L_p denote p's *local* incident light. It is worth pointing out that p is actually the intersection point of ray $q \rightarrow \omega$ with object O, which in turn depends on direction ω.

Suppose object O is illuminated only by a distant environment light L. Then L_p can be represented using a light transport operator $\mathbf{T}_p$ as: $L_p = \mathbf{T}_p[L]$, representing self shadowing and self interreflections. Note that L is the *global* distant environment light independent of the location of point p. See [SKS02, SLSS03] for a detailed description on the transport operator $\mathbf{T}_p$.

By representing the environment light in SH basis functions: $L_p(\mathbf{s}) = \sum_{i=1}^n l_i y_i(\mathbf{s})$, we have $L_p(\mathbf{s}) = \mathbf{T}_p[L](\mathbf{s}) = \sum_{i=1}^n \left(\sum_{j=1}^n \mathbf{T}_p^{ij} l_j\right) y_i(\mathbf{s})$, where $y_i(\mathbf{s})$ denote the SH basis functions, $\mathbf{T}_p^{ij}$ denotes the element of the transfer matrix $\mathbf{T}_p$. Substituting $L_p(\mathbf{s})$ in Equation 1, the reflected light received by q can be reexpressed as: $I_q(\omega) = \sum_{j=1}^n \mathbf{M}_q^j(\omega) l_j$, where

$$\mathbf{M}_q^j(\omega) \ = \ \sum_{i=1}^n \mathbf{T}_p^{ij} \int y_i(\mathbf{s})\ f_p(\omega,\mathbf{s})\ (\mathbf{s}\cdot\mathbf{n}_p)_+ \mathrm{d}\mathbf{s}.$$

We represent $\mathbf{M}_q^j(\omega)$ in the SH basis as well: $\mathbf{M}_q^j(\omega) = \sum_{i=1}^n \mathbf{M}_q^{ij} y_i(\omega)$, to arrive at the following SH expansion for the reflected light:

$$I_q(\omega) = \sum_{i=1}^n \left(\sum_{j=1}^n \mathbf{M}_q^{ij} l_j \right) y_i(\omega). \tag{2}$$

We call $\mathbf{M}_q^j(\omega)$ the *interreflection transfer function* (ITF) of point q, since it expresses the interreflected radiance as a function of the incident light of the object.

However, due to occlusion, interreflection and local lighting, the incident light over an object's surface cannot be represented as a single environment light. In the next subsection, we introduce a proxy based approximation method to address this problem.

3.2. Proxy based approximation

We observed that for low frequency local lighting, though the global incident light of an object varies from part to part, the variation is very smooth. Therefore, we can place several sample points, called *proxies*, near the object to capture such variation in global incident light. Let $\chi_1,\ldots,\chi_m$ be the proxies, and $L_1,\ldots,L_m$ be the global incident lights going towards the proxies. Then we can use them to approximate

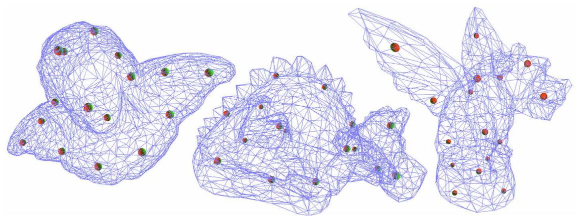

Figure 3: *Proxy locations shown as small balls. Simplified meshes are used for better visualization.*

the global incident light elsewhere. For any point p on the object, we approximate its global incident light by a distance based interpolation method: $\sum_{k=1}^{m} w_{p,k} L_k(\mathbf{s})$, where $w_{p,k}$ are normalized inverse distances, i.e.,

$$w_{p,k} = \frac{\|p-\chi_k\|^{-1}}{\sum_{j=1}^{m}\|p-\chi_j\|^{-1}}. \quad (3)$$

By the linear addition property of light transfer, the transferred local incident light $L_p(\mathbf{s})$ of point p can be expressed as follows:

$$L_p(\mathbf{s}) = \sum_{k=1}^{m} \mathbf{T}_{p,k}[L_k](\mathbf{s}), \quad (4)$$

where $\mathbf{T}_{p,k}$ denotes the operator which transfers the global incident light at proxy χ_k to the local incident radiance of point p. As explained in Section 3.1, $\mathbf{T}_{p,k}$ can be obtained by a Monte Carlo simulator [SKS02, LK03].

Let $l_{k,j}$ be the j-th SH coefficient of the global incident radiance of the k-th proxy χ_k. Substituting Equation 4 into Equation 1 and following the same development in Section 3.1, we have

$$I_q(\omega) = \sum_{i=1}^{n}\left(\sum_{k=1}^{m}\sum_{j=1}^{n} \mathbf{M}_{q,k}^{ij} l_{k,j}\right) y_i(\omega), \quad (5)$$

$$\mathbf{M}_{q,k}^{ij} = \int \mathbf{M}_{q,k}^{j}(\omega) y_i(\omega) \mathrm{d}\omega, \quad (6)$$

$$\mathbf{M}_{q,k}^{j}(\omega) = \sum_{i=1}^{n} \mathbf{T}_{p,k}^{ij} \int y_i(\mathbf{s})\, f_p(\omega,\mathbf{s})\, (\mathbf{s}\cdot\mathbf{n}_p)_+ \mathrm{d}\mathbf{s}. \quad (7)$$

Every proxy is associated with a transfer matrix $\mathbf{M}_{q,k} \equiv (\mathbf{M}_{q,k}^{ij})_{n\times n}$ for each point q. A large number of proxies will incur a huge amount of precomputation, storage and rendering cost. Therefore it is important to choose as few proxies as possible and carefully choose the proxies' locations to reduce the approximation error.

Proxy Selection In selecting the location of the proxies, the goal could be to minimize the total interpolation error all over the surface of the object. However, the global incident light is not known until rendering time and can't be determined in the precomputation phase. Therefore, we adopt the k-means clustering algorithm on the object's vertex set and choose the resultant clusters' centers as the proxies to obtain uniform proxy distribution.

In Figure 3 we show the proxy locations for several objects used in this paper. The figure demonstrates that nearly uniform proxy distribution can be obtained by using k-means clustering.

3.3. Radiance Transfer Field (RTF)

In order to render dynamic scenes with moving objects, we precompute for each object the interreflection transfer functions in its surrounding space, storing them in a *radiance transfer field* (RTF). As shown in Figure 2 (Right), the sampling points of the radiance transfer field are arranged at some concentric spheres, which is the same arrangement as the precomputed shadow fields [ZHL*05].

At each sample point of the precomputed RTF, there are a number of interreflection transfer matrices, $\mathbf{M}_{q,1},\ldots,\mathbf{M}_{q,m}$, each of which corresponds to a proxy of the object. Therefore the size of precomputed data for each sample is $n\times n\times m$. In the next section, we will present how to precompute the RTF.

4. Precomputation

Since the transfer operators, $\mathbf{T}_{p,1},\ldots,\mathbf{T}_{p,m}$, are involved in computing the transfer matrices $\mathbf{M}_{q,1},\ldots,\mathbf{M}_{q,m}$ (see Equation 6, Equation 7), we first run a Monte Carlo simulator to compute them for all vertices p of the object.

Consider the local incident radiance $L_p(\mathbf{s})$. Let γ denote the ray originating from p and going in direction $\mathbf{s}$, $V_p(\mathbf{s})$ be the self visibility function of vertex p valued 1 when visible in direction $\mathbf{s}$ and 0 when invisible. $L_p(\mathbf{s})$ consists of two parts: when $V_p(\mathbf{s})=1$, $L_p(\mathbf{s})$ equals the global incident radiance ; when $V_p(\mathbf{s})=0$, ray γ intersects with the object itself and $L_p(\mathbf{s})$ equals to the outing radiance of the intersection point in direction $-\mathbf{s}$. Therefore $L_p(\mathbf{s})$ can be expressed as:

$$L_p(\mathbf{s}) = V_p(\mathbf{s})\sum_{k=1}^{m} w_k(p) L_{c,k}(\mathbf{s}) + (1-V_p(\mathbf{s}))\, I_{x(\mathbf{s})}(-\mathbf{s}), \quad (8)$$

where $x(\mathbf{s})$ is intersection point and $I_{x(\mathbf{s})}(-\mathbf{s})$ is the outgoing radiance of $x(\mathbf{s})$ in direction $-\mathbf{s}$.

Since $I_{x(\mathbf{s})}(-\mathbf{s}) = \int L_{x(\mathbf{s})}(\mathbf{s}')\, f_{x(\mathbf{s})}(-\mathbf{s},\mathbf{s}')\, (\mathbf{s}'\cdot\mathbf{n}_{x(\mathbf{s})})_+ \mathrm{d}\mathbf{s}'$, computing the local incident light $L_p(\mathbf{s})$ at vertex p involves recursively computing the local incident light at $x(\mathbf{s}')$ to account for the self interreflection contribution. To facilitate the simulation, the weighting coefficients $w_k(p)$ of the vertices can be computed in advance and stored for reusing.

To compute the self interreflection term, we use Monte Carlo simulation. Our simulator is almost the same as that in [SKS02], except that we use weighted global incident radiance of the proxies to compute the direct illumination component. When there is one proxy per object, our simulator is equivalent to that in [SKS02].

Now we compute the interreflection transfer functions in RTF using Equation 6 and Equation 7. For any sampling point q of the RTF, we first determine for each direction ω the intersection point p on the object and evaluate $h_p^i(\omega) \equiv \int y_i(\mathbf{s})\ f_p(\omega,\mathbf{s})\ (\mathbf{s}\cdot\mathbf{n}_p)_+ d\mathbf{s}$ for all $1 \le i \le n$ in Equation 7; then compute $\mathbf{M}_{q,k}^j(\omega) = \sum_{i=1}^n \mathbf{T}_{p,k}^{ij} h_p^i(\omega)$ for all $1 \le j \le n$ and $1 \le k \le m$. At last we project $\mathbf{M}_{q,k}^j(\omega)$ to the SH basis obtaining the final transfer matrix $\mathbf{M}_{q,k} = \left(\mathbf{M}_{q,k}^{ij}\right)_{n\times n}$ for all $1 \le k \le m$. We compute $\mathbf{M}_{q,k}^j(\omega)$ for $6\times 64\times 64$ directions on a unit cubemap.

4.1. Compression of RTF

As shown in Equation 7, we have m transfer matrices $\mathbf{M}_{q,1},\dots,\mathbf{M}_{q,m}$ for each sample q in a RTF, which is much larger than precomputed shadow fields in the same resolution. In our experiments, we use 4th-order SH, $8\sim 24$ proxies, and sample 16 concentric spheres with $6\times 16\times 16$ directions, which results in $192\sim 576$MB of storage per object. This data size will be intractable when there are multiple objects in the scene.

Interreflection is usually very smooth in the surrounding space, particularly when the illumination is of low-frequency. Hence, strong spatial coherence exists between the transfer matrices of nearby positions and the matrices can be highly compressed. We employ CPCA [SHHS03] to compress the RTF data.

We rearrange the transfer matrices of each sample to form a $n\times n\times m$ dimension vector for each sample position. After CPCA, the sample positions are partitioned into a few clusters, say $C_1,\dots,C_{n_c}$. Consider cluster C_i, $\forall$ sample $r \in C_i$, we can reconstruct its transfer matrix by

$$\mathbf{M}_{r,k} = \sum_{j=1}^{n_e} \alpha_j(r)\mathbf{E}_{ij,k}, \tag{9}$$

where $\mathbf{E}_{ij,k}$ is the j-th eigen transfer matrix corresponding to proxy k of cluster C_i, and $\alpha_j(r)$ are the reconstruction coefficients.

In our experiments, we use 32 clusters and keep 16 terms for each cluster. The compression ratio is about $40\sim 50$: 1 and the compression error is about 1% for diffuse object and 3% for specular objects. The error is defined using the Frobenius norm of the difference matrix. Figure 4 shows more error analysis.

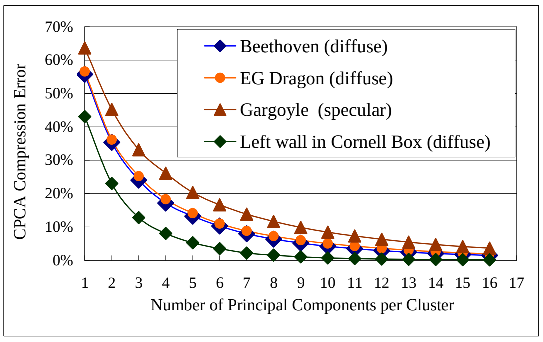

Figure 4: *CPCA compression error v.s. number of principal components per cluster.*

5. Rendering

In this section, we introduce our rendering algorithm, which computes the interreflection effects using the precomputed RTF. We adopt precomputed shadow fields as our rendering framework. The scene is regarded as a set of entities including objects and local light sources. For each scene entity, if it casts shadow, then its *object occlusion field* (OOF) is precomputed; if it generates interreflection, then its RTF is precomputed; if it is a light source, then its *source radiance field* (SRF) is precomputed. Details on OOF and SRF can be found in [ZHL*05]. The distant environment light is also regarded as the furthest scene entity with constant SRF. For all objects, the self-transfer matrices are precomputed and stored for rendering self shadows and self interreflection.

Rendering interreflection effects in our system consists of two steps: during the “bouncing step" we compute the global incident light at proxies; during the “shading step" we compute the final shadings for all object vertices.

Bouncing step By the definition of the interreflection transfer function in Section 3, the interreflected light at a point q can be simply computed by matrix-vector multiplications $\sum_k \mathbf{M}_{q,k}L_k$ given the global incident radiance $L_1,\dots,L_m$ of the proxies. If there are multiple objects, reflected light from all the objects is summed after being combined with objects' occlusion. Therefore, the key is to compute $L_1,\dots,L_m$ for all objects from occlusion, interreflection and light sources.

The procedure to compute the global incident radiance for all proxies of all objects is outlined as follows: (i) we first reset all proxies' incident radiance to $\mathbf{0}$; (ii) then for each object J and each proxy χ_k of J, we call ComputeIncidentRadiance(χ_k, J, $L_k(J)$) in **Algorithm 1** to compute the incident radiance $L_k(J)$ of the next bounce from the current bounce; we repeat step (ii) until the proxies' incident radiance converge or we reach a maximum number of bounces. Finally, the incident radiance from each bounce is summed up to obtain the overall incident radiance at each proxy. Subroutine *ComputeIncidentRadiance* in **Algorithm 1** computes the incident radiance for any point based on the shadow field rendering framework.

Shading step After the global incident radiance at all proxies is precomputed, the shading step can be executed as follows: for each visible vertex v of each object J_v in the scene, we first compute the viewing direction ω_v and obtain $B_v(\omega_v)$, which is the triple product of the self-

Algorithm 1. Compute incident radiance for a given point

```
ComputeIncidentRadiance($q$,$J_q$, $L_q$)
 $L_q$           : local incident radiance at $q$, return value
 $q$             : a point in the scene
 $J_q$           : the scene entity to which $q$ belongs
 $m_i$           : number of proxies for entity $J_i$
 // the following 4 variables are in $J_i$'s local frame
 $L_k(J_i)$      : global incident light of the $k$-th proxy of $J_i$
 $SRF(J_i,q)$    : source radiance field of $J_i$ at $q$
 $OOF(J_i,q)$    : object occlusion field of $J_i$ at $q$
 $RTF_k(J_i,q)$  : radiance transfer field of $J_i$ at $q$
                   corresponding to $J_i$'s $k$-th proxy
begin
    $L_q \leftarrow \mathbf{0}$   // reset local incident radiance
    $V_q \leftarrow \mathbf{1}$   // initialize visibility function to 1
    Compute distance from $q$ to all scene entities
    Sort the entities except $J_q$ from near to far: $J_1,\dots,J_N$
    for $i = 1$ to $N$ do
        if $J_i$ is a light source do
            Look up $SRF(J_i,q)$
            Rotate $SRF(J_i,q)$ to the global frame
            $L_q \leftarrow L_q + TripleProduct(V_q, SRF(J_i,q))$
        end
        if $J_i$ generate interreflection do   // our contribution
            $I_q \leftarrow \mathbf{0}$   // for accumulating interreflection
‡           for $k = 1$ to $m_i$ do
†               Lookup $RTF_k(J_i,q)$
                $I_q \leftarrow I_q + RTF_k(J_i,q)L_k(J_i)$
            end
§           Rotate $I_q$ to the global frame
            $L_q \leftarrow L_q + TripleProduct(V_q, I_q)$
        end
        if $J_i$ casts shadow do
            Look up $OOF(J_i,q)$
            Rotate $OOF(J_i,q)$ to the global frame
            $V_q \leftarrow TripleProduct(V_q, OOF(J_i,q))$
        end
    end
end
```

visibility and the 2D BRDF slice with viewing direction of ω_v, both represented by a SH coefficient vector. Then ComputeIncidentRadiance(v, J_v, L_v) is called to compute the incident radiance L_v going toward vertex v. Finally, the shading for vertex v is evaluated as a dot product: $I_v = B_v(\omega_v)\cdot L_v$.

For the above shading step, we designed two caching techniques to accelerate computing interreflection contributions in ComputeIncidentRadiance(v, J_v, L_v).

Caching RTF samples When we lookup $RTF_k(J_i,q)$ (refer to the line marked with † in **Algorithm** 1), we first find the eight samples that are nearest to q (4 on each of the 2 adjacent spheres), then compute $RTF_k(J_i,q)$ using trilinear interpolation: $RTF_k(J_i,q) = \sum_{l=1}^{8} w_l(q)\ RTF_k(J_i,r_l)$, where $RTF_k(J_i,r_l)$ are the transfer matrices precomputed at sample

	Objects	Vertices	RTF size	FPS
Cornell-box	7	22K	44.1MB	8-12
Cornell-box dynamic	8	40K	50.0MB	10-16
Table	4	40K	32.8MB	8-10
Brick game	10	86K	63.5MB	5-8

Table 1: *RTF rendering performance.*

r_l, $w_l(q)$ are the trilinear weighting coefficients. Therefore, we have

$$I_q \leftarrow I_q + \sum_{l=1}^{8} w_l(q)\ (\sum_{k=1}^{m_i} RTF_k(J_i,r_l)\ L_k(J_i)) \tag{10}$$

(refer the for loop marked with ‡ in **Algorithm** 1).

Since during the shading step within each frame the proxies' incident radiance are fixed, we can cache the results of

$$\xi_{r_l,i} \equiv \sum_{k=1}^{m_i} RTF_k(J_i,r_l)L_k(J_i) \tag{11}$$

for computing Equation 10 and reuse them when we call ComputeIncidentRadiance for another vertex which also interpolates from these samples. This caching technique is very effective, particularly when the object has a dense tessellation.

Caching RTF eigens Recall that our RTF is stored in a compressed form after CPCA compression, as shown in Equation 9. Let C_h be the cluster containing sample r_l, then $RTF_k(J_i,r_l) = \sum_{j=1}^{n_e} \alpha_j(r_l)\mathbf{E}_{hj,k}(J_i)$. Substitute $RTF_k(J_i,r_l)$ in Equation 11, yielding

$$\xi_{r_l,i} \equiv \sum_{k=1}^{m_i}(\sum_{j=1}^{n_e} \alpha_j(r_l)\mathbf{E}_{hj,k}(J_i)L_k(J_i)) \tag{12}$$

Since the number of eigens (principal components) is very small, we can cache the result of $\mathbf{E}_{hj,k}(J_i)L_k(J_i)$ for fast updating of $\xi_{r_l,i}$.

Experimental results shows that the above two caching techniques both have a hit rate over 95%. With each cache hit, we save a lot of computation on matrix-vector multiplication and vector interpolation. In our experiments, these caching techniques reduced the cost of computing interreflections by more than 90%.

6. Results

We have implemented the above techniques and developed a prototype interactive rendering system. The system is fully implemented on CPU and uses SSE and multithreading, which together gives a 6x speed boost. Now we present some results obtained on a desktop PC with Intel Core 2 Duo E6300, 2G RAM and a Geforce 7800 graphics card. The performance for the scenes are listed in Table 1.

We precompute RTF for each object on concentric spheres

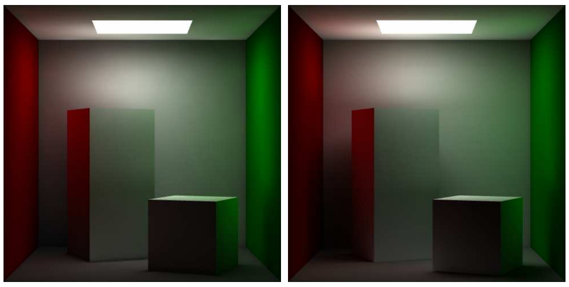

Figure 5: *Comparison between an image rendered by our method (left) and a reference image rendered by radiosity method (right).*

around the object uniformly distributed in the range from $0.1 \times \eta$ to $8 \times \eta$, where η is the radius of the object's bounding sphere. This is a reasonably large range for interreflection effects. Each concentric sphere is sampled with $6 \times n_r \times n_r$ directions by uniformly subdividing a cubemap. We take $n_r = 16$ for diffuse objects and $n_r = 32$ for glossy objects. This sampling density works well in our experiments. However, adaptively sampling the field remains an open problem. The precomputation time ranges from 40 minutes to 2 hours, based on the number of proxies.

In Figure 5 we compare the rendered result of a Cornell-box using our method with the result of a radiosity method. There is a white area light on the top of the scene. Though there are a lot of approximation in RTF and shadow fields, the rendered image of this Cornell-box consisting of pure diffuse objects is visually convincing.

In Figure 6 we show more rendering results of Cornell box by changing the BRDF on the left wall. The floor and the lower part of the left box receives more interreflections from the left wall as its specular component increases, while the ceiling gets less interreflections from the left wall as its diffuse component decreases. These effects make sense considering that a specular wall will reflect more light around the reflection direction and the light source is on the ceiling.

We put some dynamically moving object in the Cornell-box and show some screen shots in Figure 7. As the objects move, we can see very nice interreflections, color bleeding and soft shadow effects between the walls, the angel and the sphere.

In Figure 8 we show a comparison of rendering results with and without interreflections. The scene is illuminated by a distant environment lighting. From side-by-side comparison, it can be easily seen that the image including interreflections looks more realistic. Figure 8 (right) shows the difference image, i.e. the interreflection component. Though the magnitude of interreflection is relative small, it provides important visual cues for realistic images.

In Figure 1 and Figure 9 we show some scene shots of a toy bricks game. The scene is lit with an environment map and a moving local light. It can be seen in Figure 9(b) and (c) that the reflected light on the plane faithfully follows the distribution of the reflectors.

More animation results of the dynamic scenes are presented in the supplementary video.

7. Conclusions

We have presented a novel PRT technique for rendering interreflections in dynamic scenes. Experimental results show that our technique can render very convincing interreflection effects. Our main contributions are (1) the introduction of the *radiance transfer field* (RTF), a field-based formulation for computing interreflections; (2) a proxy based approximation method for precomputed RTF; and (3) an efficient interreflection rendering algorithm using RTF and shadow fields.

There are some limitations in our work. Currently we can only handle low-frequency interreflections and illumination. Though RTF is conceptually applicable to all-frequency effects, it is impractical for now due to the huge amount of data and the lack of an efficient wavelet rotation algorithm. We are also limited to rigid objects. Rendering interreflections with deformable objects is interesting area of future work.

In the future, we would like to extend RTF to render more global illumination effects such as caustics and translucent materials. We are also interested in a complete GPU implementation of our rendering algorithm.

Acknowledgements

We would like to thank Adam W. Bargteil in CMU Graphics Lab for paper proofreading, the anonymous reviewers for their valuable comments, Jianxiong Pan and Zhi He for scene modeling and video production, and Paul Debevec for the light probes.

This work is supported by 973 Program of China (No. 2002CB312102), NSFC (No. 60603078), the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China (No.705027), and Program for New Century Excellent Talents in University.

References

[AKDS04] ANNEN T., KAUTZ J., DURAND F., SEIDEL H.-P.: Spherical harmonic gradients for mid-range illumination. In *Eurographics Symposium on Rendering* (June 2004), pp. 331–336.

[GKMD06] GREEN P., KAUTZ J., MATUSIK W., DURAND F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Symposium in Interactive 3D Graphics and Games* (March 2006).

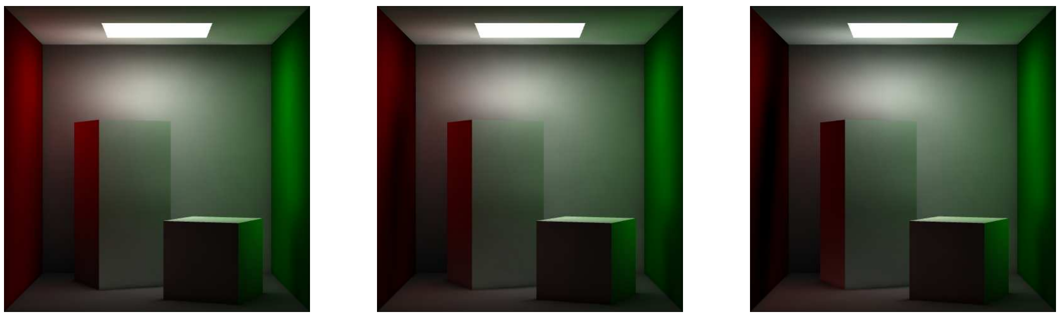

Figure 6: *Interreflection effects generated by changing the BRDF of the left red wall. Left: pure diffuse; Middle: less diffuse + specular; Right: pure specular. The floor receives more interreflections as the specular component increases, and the ceiling receives less interreflections as the diffuse component decreases.*

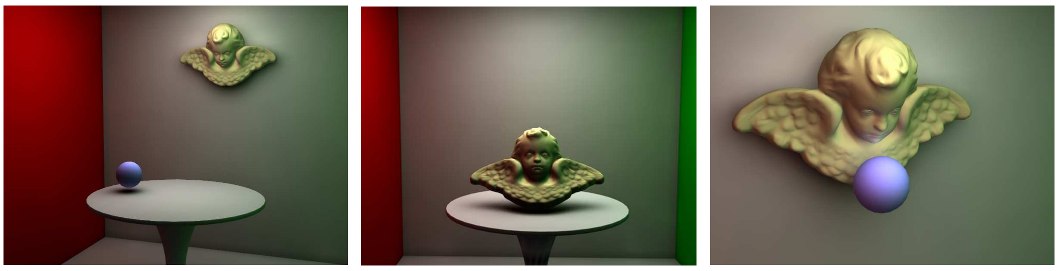

Figure 7: *Screen shots of the Cornell-box with dynamically moving objects. We obtain very nice interreflection effects (color bleeding) as well as soft shadows between the wall and the angel (left), the angel and the table (middle) and the sphere and the angel (right).*

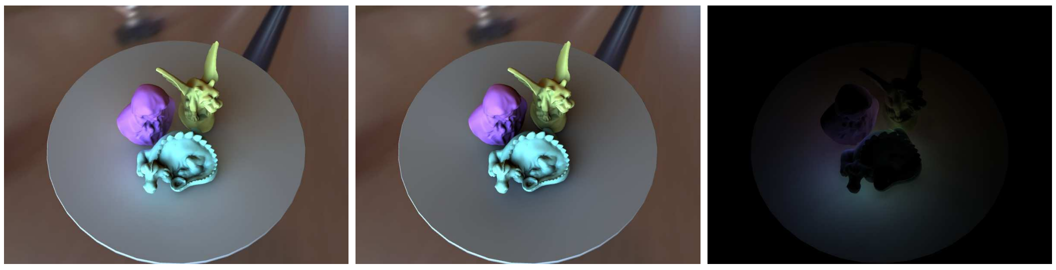

Figure 8: *Comparison between rendering with (left) and without (middle) interreflections. The interreflection component is shown on the right.*

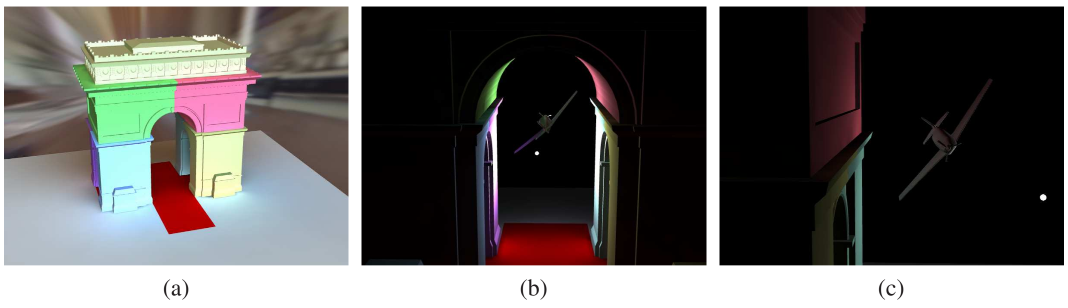

Figure 9: *Some screen shots of a toy bricks game.*

[HBV03] HAO X., BABY T., VARSHNEY A.: Interactive subsurface scattering for translucent meshes. In *Symposium on Interactive 3D graphics* (2003), pp. 75–82.

[HPB06] HAŠAN M., PELLACINI F., BALA K.: Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph. 25*, 3 (2006), 1089–1097.

[IDT*06] IWASAKI K., DOBASHI Y., TAMURA N., JOHAN H., YOSHIMOTO F., NISHITA T.: Precomputed radiance transfer for dynamic scenes with diffuse interreflection. In *Sketches of SIGGRAPH* (2006).

[JF03] JAMES D. L., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. 22*, 3 (2003), 879–887.

[KA07] KIRK A. G., ARIKAN O.: Real-time ambient occlusion for dynamic character skins. In *Symposium on Interactive 3D graphics and games* (2007), pp. 47–52.

[KAMJ05] KRISTENSEN A. W., AKENINE-MÖLLER T., JENSEN H. W.: Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph. 24*, 3 (2005), 1208–1215.

[KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical rasterization for self-shadowing of dynamic objects. In *Eurographics Symposium on Rendering* (2004), pp. 179–184.

[KSS02] KAUTZ J., SLOAN P.-P., SNYDER J.: Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *Eurographics Rendering Workshop* (2002), pp. 291–296.

[LK03] LEHTINEN J., KAUTZ J.: Matrix radiance transfer. In *Symposium on Interactive 3D graphics* (2003), pp. 59–64.

[LSSS04] LIU X., SLOAN P.-P., SHUM H.-Y., SNYDER J.: All-frequency precomputed radiance transfer for glossy objects. In *Eurographics Symposium on Rendering* (2004), pp. 337–344.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 3 (2003), 376–381.

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple product integrals for all-frequency relighting. *ACM Trans. Graph. 23*, 3 (2004), 477–487.

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Trans. Graph. 25*, 3 (2006), 977–986.

[SHF06] SUNSHINE-HILL B., FALOUTSOS P.: Photorealistic lighting with offset radiance transfer mapping. In *Symposium in Interactive 3D Graphics and Games* (March 2006), pp. 15–21.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (2002), 527–536.

[SLSS03] SLOAN P.-P., LIU X., SHUM H.-Y., SNYDER J.: Bi-scale radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 370–375.

[TJCN06] TAMURA N., JOHAN H., CHEN B.-Y., NISHITA T.: A practical and fast rendering algorithm for dynamic scenes using adaptive shadow fields. *The Visual Computer 22*, 9 (2006), 702–712.

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25*, 3 (2006), 967–976.

[WHS04] WYMAN C., HANSEN C. D., SHIRLEY P.: Interactive caustics using local precomputed irradiance. In *Pacific Conference on Computer Graphics and Applications* (2004), pp. 143–151.

[WTL04] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of non-diffuse objects using separable brdf approximation. In *Eurographics Symposium on Rendering* (2004), pp. 345–354.

[WTL05] WANG R., TRAN J., LUEBKE D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. 24*, 3 (2005), 1202–1207.

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph. 24*, 3 (2005), 1196–1201.