1 Introduction

In this sketch, we propose a new precomputed radiance transfer method for rendering dynamic scenes taking into account diffuse interreflection between objects, whereas this is not possible using previous methods [Sloan et al. 2002; Zhou et al. 2005]. We demonstrate interactive rendering of dynamic scenes with moving objects under moving local light sources and dynamic distant illumination. Moreover, our method can handle the changes of the diffuse reflectance at runtime.

2 Overview of Our Method

We compute the interreflected light illuminated by an object, by considering the object as a secondary light source. The incident lighting from a light source whose illumination distribution does not change can be represented by a source radiance field (SRF) proposed by Zhou et al. [Zhou et al. 2005]. However, the illumination distribution of the diffuse surface changes in dynamic scenes. To address this problem, we represent the illumination distribution of the object surface using basis functions and precompute each SRF whose illumination distribution is represented by each basis function. We represent the illumination distribution of the diffuse surface using the Fourier series.

Precomputation Each point on the object surface is represented by parameter \((u,v)\) \((0 \leq u,v \leq 1)\), and the illumination distribution of the object surface is also expressed by \(L(u,v)\). We expand \(L(u,v)\) using the Fourier series \(Ψ(u,v)=\sum w_iΨ_i(u,v)\), where \(w_i\) is the Fourier coefficient for the \(i\)-th term \(Ψ_i(u,v)\) of the Fourier series. Here, we expand \(L(u,v)\) for the RGB channels in Fourier series and \(w_i\) includes the RGB channels. Each Fourier coefficient \(w_i\) is calculated from \(w_i = \int_0^1 \int_0^1 L(u,v)Ψ_i(u,v)du dv\). We approximate the illumination distribution of an object by using \(N\) terms of the Fourier series \(Ψ(u,v)\), and precompute the SRF for each \(i\)-th term \(Ψ_i(u,v)\) of the Fourier series (Fig. 1(a)).

Rendering The radiance \(B_A(p)\) at vertex \(p\) illuminated by object \(A\) is expressed by \(B_A(p) = \int_0^1 \int_0^1 \sum \rho_d/\pi \cdot (n_p \cdot v) \cdot V(p,ω) \cdot (ρ_d/\pi) \cdot (n_p \cdot ω) \cdot L(p,ω) \cdot T(p,ω) \cdot dω\) (see Fig. 1(b)). \(T(p,ω)\) is a transfer function at \(p\) and expressed by \(V(p,ω) \cdot (ρ_d/\pi) \cdot (n_p \cdot ω)\), where \(V(p,ω)\) is a binary visibility function, \(ρ_d/\pi\) is a diffuse reflectance, and \(n_p\) is a normal at \(p\). \(V(p,ω)\) incorporates the occlusions of nearer objects than \(A\) from \(p\), and is calculated by using the object occlusion field (OOF) [Zhou et al. 2005]. The incident lighting from object \(A\) at \(p\) is calculated from \(L(p,ω) \approx \sum w_i S_A(p,ω)\), where \(S_A(p,ω)\) is the SRF of object \(A\) for each \(Ψ_i\). \(B_A(p)\) is calculated by using the following equation,

\[
B_A(p) = \sum_{i=0}^{N-1} w_i \left( \int \Omega S_A(p,ω)T(p,ω)dω \right) = \sum_{i=0}^{N-1} w_i B_i^A(p).
\]

We call \(B_i^A(p)\) \(i\)-th basis radiance at \(p\) illuminated by object \(A\) whose illumination distribution is represented by \(i\)-th basis function \(Ψ_i\). We compute the basis radiance for each object and \(Ψ_i\) on the fly by using the precomputed OOFs and SRFs.

For rendering, we first calculate the radiance due to direct illumination and each basis radiance at each vertex. Then each Fourier coefficient \(w_i\) for \(Ψ_i(u,v)\) of the Fourier series is calculated based on the illumination distribution of the object surface. The calculation of the radiance \(B_A(p)\) of interreflected light from object \(A\) is just the dot product of \(w_i\) and each basis radiance \(B_i^A(p)\) (Fig. 1(b)).

GPU implementation Our implementation on the GPU supports low-frequency lighting. Rendering soft shadows and interreflection is performed by calculating the dot and triple products of the coefficient vectors representing the transfer function and SRFs at each vertex. To compute the dot and triple products on the GPU, we prepare a floating point texture for each object and each pixel of the texture corresponds to each vertex of the corresponding object. This makes it possible to calculate the dot and triple products of coefficient vectors at each vertex by using the fragment program of the GPU for the acceleration.

3 Results

Fig. 2 shows the results rendered by using our method taking into account 5 bounce interreflected light. Fig. 2(a) shows a room scene consisting of 10 objects illuminated by a square light source. Fig. 2(b) shows a scene consisting of 12 objects illuminated by low-frequency, distant illumination. The distant illumination in Fig. 2(b) is represented by an environment map shown at the bottom left of Fig. 2(b). We use 4 terms of Fourier series. The computational times on Pentium 4 3.6GHz with GeForce7800 GTX for Figs. 2(a) and (b) are 9.0 and 4.0 fps, respectively. Our method can render dynamic scenes with diffuse interreflection at interactive frame rates.

Figure 1: Our rendering method for dynamic scenes with interreflection.

Figure 2: Examples rendered by using our method.

References
