Real-Time Rendering of Translucent Meshes

XUEJUN HAO and AMITABH VARSHNEY
University of Maryland at College Park

Subsurface scattering is important for photo-realistic rendering of translucent materials. We make approximations to the BSSRDF model and propose a simple lighting model to simulate the effects on translucent meshes. Our approximations are based on the observation that subsurface scattering is relatively local due to its exponential falloff.

In the preprocessing stage we build subsurface scattering neighborhood information, which includes all the vertices within effective scattering range from each vertex. We then modify the traditional local illumination model into a run-time two-stage process. The first stage involves computation of reflection and transmission of light on surface vertices. The second stage bleeds in scattering effects from a vertex's neighborhood to generate the final result. We then merge the run-time two-stage process into a run-time single-stage process using precomputed integrals, and reduce the complexity of our run-time algorithm to $O(N)$, where N is the number of vertices. The selection of the optimum set size for precomputed integrals is guided by a standard imagespace error-metric. Furthermore, we show how to compress the precomputed integrals using spherical harmonics. We compensate for the inadequacy of spherical harmonics for storing high frequency components by a reference points scheme to store high frequency components of the precomputed integrals explicitly. With this approach, we greatly reduce memory usage without loss of visual quality under a high-frequency lighting environment and achieve interactive frame rates for medium-sized scenes. Our model is able to capture the most important features of subsurface scattering: reflection and transmission due to multiple scattering.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation—*viewing algorithms*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*graphics data structures and data types*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*color, shading, shadowing, and texture*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: BSSRDF, local illumination, reflection models, subsurface scattering

1. INTRODUCTION

Illumination models are important for photo-realistic image synthesis. Correctly modeling the physical interaction of light with objects is an exciting, but difficult task. Over the years, many illumination models have been developed for image synthesis. They can be classified as either empirically-based or physically-based. For example, the Phong illumination model [Phong 1975] is an empirically-based model. Physically based models are derived from principles of light-object interaction, using either geometrical optics or wave optics. Most of them model the bidirectional reflectance distribution function (BRDF).

One example of physically-based models using geometrical optics is the Cook-Torrance [1981] model, which can compute directional distribution of light and color shift with incident angles and materials.

This work was supported in part by National Science Foundation (NSF) grants IIS-00-81847 and ACR-98-12572/02-96148.
Authors' address: Department of Computer Science and UMIACS, 1103 A. V. Williams Building, University of Maryland, College Park, MD 20742; email: {hao,varshney}@cs.umd.edu.

Other geometrical-optics-based models include microfacet-based approaches [Ashikhmin et al. 2000; Blinn 1977]. Inverse rendering methods can produce high-quality illumination models from images [Basri and Jacobs 2001; Cabral et al. 1987; Debevec et al. 2000; Ramamoorthi and Hanrahan 2001; Sato et al. 1997; Yu et al. 1999]. Significant efforts have been devoted to determining the BRDF of an object. Researchers have also developed methods to directly measure the BRDF [Greenberg et al. 1997; Marschner et al. 1999; Ward 1992].

Compared with geometrical-optics-based models, wave-optics-based models are more complicated, but have the advantage of being able to model phenomena which cannot be directly modeled using geometrical optics, such as interference and diffraction patterns. Kajiya [1985] has used scalar-form Kirchhoff approximation to compute the BRDF of surfaces with anisotropy. He et al. [1991] have presented a general local reflection model based on vector-formed Kirchhoff wave diffraction theory and have given an analytical formula to compute the BRDF for surfaces with roughness, including polarization and directional Fresnel effects. Bahar and Chakrabarti [1987] have computed the differential scattering cross-section of a wave from rough metallic surfaces using electromagnetic theory. Stam [1999] and Sun et al. [2000] have extended the He–Torrance model [He et al. 1991] to handle anisotropic reflections and demonstrated diffraction effects on a compact disk.

A good BRDF model, either derived or measured, can give highly realistic visual effects. The basic assumption of BRDF models is that light enters and exits an object on the same surface point. In most cases this assumption is valid and the resulting BRDF models provide appearances convincing visual for simulating many visual effects. But for some cases, the assumption is not valid. For example, BRDF models alone are inadequate to simulate the appearance with subsurface scattering, where light enters an object at one point and exits at another. This effect is very important for simulating the appearance of translucent materials, such as marble, skin, and milk. To simulate these materials, we have to go back to the more general bidirectional surface scattering reflectance distribution function (BSSRDF) models, since BRDF models are just approximations of BSSRDF models.

Many researchers have successfully simulated subsurface scattering effects. Hanrahan and Krueger [1993] have modeled subsurface scattering in layered surfaces in terms of one-dimensional linear transport theory, and derived analytical expressions for single scattering events. They have incorporated their results into a BRDF model. The model is fast but it has the shortcoming of the BRDF assumption. More recently, Dorsey et al. [1999] have simulated subsurface transfer by solving the radiative transfer equation using photon maps. Koenderink and van Doorn [2001] model light scattering in translucent objects as a diffusion process. Stam [2001] used a discrete-ordinate solution of the radiative transfer equation to model multiple anisotropic scattering for a human skin layer bounded by two rough surfaces. Another contribution of Stam [2001] is derivation of a bidirectional transmittance distribution function (BTDF) to complement BRDF models. Pharr and Hanrahan [2000] have taken a different approach. Instead of simulating energy transport, they have focused on scattering behavior and solve a non-linear integral scattering equation using Monte Carlo evaluation. Jensen et al. [1999] have used path tracing to simulate subsurface scattering in wet materials.

The approaches above are able to simulate all the effects of subsurface scattering and generate impressive images, but they are slow. Jensen et al. [2001] have suggested a more efficient approach to simulate scattering media by using a dipole diffusion approximation for multiple scattering events, with an exact solution for single scattering events. With this simple approximation, they achieve more than two orders of magnitude speedup compared with the approach of using full Monte Carlo simulation. As an example, for one scene they have reduced the rendering time from 1250 minutes to 5 minutes with nearly indistinguishable visual difference. Jensen and Buhler [2002] have taken this one step further. They decouple the computation of the incident illumination from the evaluation of the BSSRDF with a two-pass approach. The first pass samples the irradiance at selected points on the

surface. The second pass evaluates the diffusion approximation using a fast hierarchical scheme. They achieve up to 7 seconds per frame using ray-tracing for a teapot dataset with 150K vertices, using a dual 800 MHz Pentium III PC. Lensch et al. [2002] have used a preprocessing step to compute the impulse response for each surface point under subsurface scattering. They separate the response into a local and a global effect. While the local effect is modeled as a filter kernel and stored in a texture map, the global response is stored as vertex-to-vertex throughput factors. The local and global responses are combined during run-time to form the final image. They achieve 5 frames per second on a dataset with about 9K vertices, using a dual 1.7 GHz Xeon computer. In addition, they can accommodate non-homogenous material properties. All these make practical simulation of subsurface scattering phenomena feasible. The next step is to enable subsurface scattering effects for interactive rendering of larger datasets.

We have built a simpler, approximate model [Hao et al. 2003] based on previous methods and accommodated it into a local illumination model to make the effects more widely accessible for different applications. Our approach is based on the observation that subsurface scattering, although a global effect, is largely a local one due to its exponential falloff, which limits the volume it can affect. Therefore even though the light does not necessarily exit an object at the same point where it enters, as required by a BRDF model, it will for all practical purposes exit within a short distance of its entry point. This enables us to make modifications to existing local illumination models to accommodate subsurface scattering effects. We approximate the BSSRDF for subsurface scattering based on both the underlying physical processes and the visual appearance. Jensen and Buhler [2002] have shown that the visual appearance for translucent materials can be almost entirely simulated by only considering multiple scattering. We have used this fact and developed a macroscopic appearance-driven approach to capture the most important features of subsurface scattering: multiple scattered reflection and transmission. We modify the local illumination process into a run-time two-stage process: a traditional local lighting stage and a scatter-bleeding stage. We then merge the run-time two-stage process into a run-time single-stage process by using precomputed integrals and improve the complexity of our run-time algorithm from $O(N^2)$ to $O(N)$. The local illumination characteristics and the preprocessed scattering neighborhood information make our approach very efficient. We have achieved 7.5 frames per second on a teapot dataset with 150K vertices, using a 2 GHz Pentium 4 PC [Hao et al. 2003].

Recently, Sloan et al. [2003] have incorporated surface scattering effects into their precomputed radiance transfer scheme and have achieved 27 frames per second on a Buddha dataset with 50K vertices, using a 2.2 GHz Pentium 4 machine. They represent precomputed view-independent subsurface scattered radiance using low-order spherical harmonics. In addition to subsurface scattering effects, their scheme has successfully simulated many of the global illumination effects, such as soft shadows, interreflections, and caustics. If their approach is used only for simulating subsurface scattering effects, they can achieve significantly faster frame rates. They have assumed low-frequency lighting environments. We instead, focus on subsurface scattering effects, but for high-frequency lighting environments (e.g., a single directional point light source). Carr et al. [2003] have modeled multiple-scattering subsurface light transport to resemble a single radiosity gathering step. By using their GPU algorithm for radiosity with a hierarchy of precomputed subsurface links, they have achieved about 30 frames per second for a dataset with 70K triangles, using GeForce FX card. Mertens et al. [2003] use a hierarchical boundary element method to solve the integral describing subsurface scattering and achieve more than 5 frames per second on a dataset with 132K triangles, using a dual 2.4 GHz Xeon computer. Their algorithm allows users to change object geometry, subsurface scattering properties, lighting, as well as viewpoint at run-time. Dachsbacher and Stamminger [2003] extend shadow maps to store depth and incident light information, and compute subsurface scattering effects by filtering the shadow map neighborhood using a hierarchical approach. They have implemented their algorithm on graphics hardware and achieved

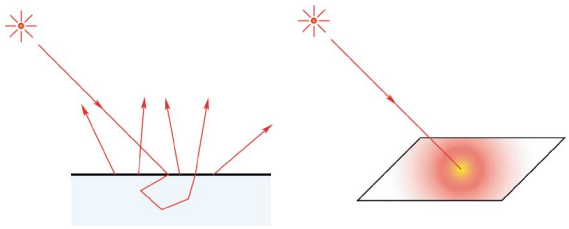


Fig. 1. Scattering of light in BSSRDF models Jensen et al. [2001].

5.7 frames per second on a dataset with 100K vertices, using a 2.4-GHz Pentium 4 machine with ATI Radeon 9700 graphics card.

In this article, we show how to greatly reduce memory storage requirements for our precomputed integrals by using reference points with spherical harmonics. We demonstrate that using only low-order spherical harmonics for representing precomputed integrals produces somewhat unsatisfactory image quality for high-frequency lighting (e.g., single directional light source). To address this, we have designed a reference points scheme. In our scheme, we select a subset of the input mesh vertices and store the precomputed integrals at these reference points. We use spherical harmonics for efficiently representing low frequency integral differences between the reference points and the remainder mesh vertices. This results in little extra storage for precomputed integrals (less than 28 bytes per vertex instead of about 200 bytes per vertex as reported in Hao et al. [2003]) without loss of image quality and a further improvement in the efficiency of our algorithm. In addition, we discuss the techniques presented in Hao et al. [2003] in more detail, compare the image error using different numbers of light sources in the preprocessing stage, and show that about 200 light source directions is a good tradeoff between image quality and storage and preprocessing requirements.

2. SUBSURFACE SCATTERING MODEL AND OUR SIMPLIFICATIONS

To describe subsurface scattering effects for translucent (i.e., highly-scattering) materials, we need general BSSRDF models instead of BRDF models. A BSSRDF model relates the illumination of one surface point with light distribution at other surface points by the following formula [Jensen et al. 2001]:

$$dL_o(x_o, \vec{\omega_o}) = S(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}) d\,\Phi_i(x_i, \vec{\omega_i})$$

where $L_o(x_o, \vec{\omega_o})$ is the outgoing radiance at point x_o in direction $\vec{\omega_o}$, $\Phi_i(x_i, \vec{\omega_i})$ is the incident flux at point x_i in direction $\vec{\omega_i}$, and $S(x_i, \vec{\omega_i}; x_o, \vec{\omega_o})$ is the BSSRDF. Thus the total outgoing radiance is computed by an integral over incoming directions and area A [Jensen et al. 2001]:

$$L_o(x_o, \vec{\omega_o}) = \int_A \int_{2\pi} S(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}) L_i(x_i, \vec{\omega_i})(\vec{n_i} \cdot \vec{\omega_i}) d\,\omega_i\, d\,A(x_i)$$

where $\vec{n_i}$ is the surface normal at x_i. As shown in Figure 1, the effect of BSSRDF results in a scatter-bleeding of a surface point from its neighborhood.

In the following sections, we will assume static geometry with homogeneous translucency and given scattering properties where multiple scattering dominates, and each vertex of the mesh represents a small area on the surface.

2.1 Locality of Subsurface Scattering Effects

To improve efficiency and achieve interactive frame rates for simulating translucent material properties, we intend to incorporate subsurface scattering effects with local illumination. The main rationale

behind a possible combination of a local illumination model with subsurface scattering effects is based on the key observation that the effects are well localized. First, scattering within one object will have very little effect on the appearance of another object; the influence between different objects can be well described by the reflectance values on their surfaces only. So unlike the situation addressed by radiosity methods where every patch has an effect on every other patch in the same scene, subsurface scattering has a prominent effect only within an object. Second, even within the same object, the subsurface scattering due to light entering from one surface point will have little effect on another surface point on the same object if the distance between the two points is large. This property is a result of the exponential falloff of light intensity due to absorption and scattering within the material. Therefore, subsurface scattering, although a global illumination property in the sense that the illumination on one surface point is affected by the illumination on other surface points, is still largely a local effect. Although, the local-effect property of subsurface scattering is useful for efficiency reasons at the preprocessing stage, it is not required for our method to work and will not affect the run-time efficiency of our algorithm. We discuss this further in Section 5.

We therefore conclude that to model the appearance of a surface point with subsurface scattering to a first approximation, we only need to know its scattering neighborhood and associated material properties.

2.2 Multiple Scattering Approximation

As mentioned earlier, a BSSRDF model is needed to describe subsurface scattering effects. The complete BSSRDF model S for subsurface scattering is a sum of a single scattering term $S^{(1)}$ and a multiple scattering term S_d [Jensen et al. 2001]:

$$S(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}) = S^{(1)}(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}) + S_d(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}).$$

Jensen and Buhler [2002] have shown that multiple scattering alone can sufficiently simulate the visual appearance of highly-scattering translucent materials. We follow their results and focus here on modeling multiple scattering effects only. Jensen et al. [2001] have also shown that the dipole diffusion method is a good approximation for volumetric effects due to subsurface multiple scattering. The dipole approximation of the diffusion equations is expressed by the following formula:

$$S_d(x_i, \vec{\omega_i}; x_o, \vec{\omega_o}) = \frac{1}{\pi} F_t(\eta, \vec{\omega_i}) R_d(\|x_i - x_o\|) F_t(\eta, \vec{\omega_o})$$

where F_t is the Fresnel transmission term and R_d is the single dipole approximation for multiple scattering [Jensen and Buhler 2002]:

$$\begin{aligned} R_d(r) &= -D \frac{(\vec{n} \cdot \vec{\nabla} \phi(x_s))}{d\Phi_i} \\ &= \frac{\alpha'}{4\pi} \left[z_r \left(\sigma_{tr} + \frac{1}{d_r} \right) \frac{\exp(-\sigma_{tr} d_r)}{d_r^2} + z_v \left(\sigma_{tr} + \frac{1}{d_v} \right) \frac{\exp(-\sigma_{tr} d_v)}{d_v^2} \right], \end{aligned}$$

where D is the diffusion constant, ϕ is the radiant fluence, Φ_i is the incident flux, α' is the reduced albedo, σ_{tr} is the effective transport coefficient, z_r and z_v are the distance from the dipole lights to the surface, d_r is the distance from x to the real source, and d_v is the distance from x to the virtual source. The configuration is shown in Figure 2. From this equation, we can see that if the scattering property of a material is homogeneous, that is, if the scattering cross-sections are constant, then the formula relates reflectance at one surface point to incident flux at other surface points. Since subsurface scattering has a limited effective range, we can obtain the reflectance of a surface point due to multiple scattering by integrating flux incident at points within a certain distance.

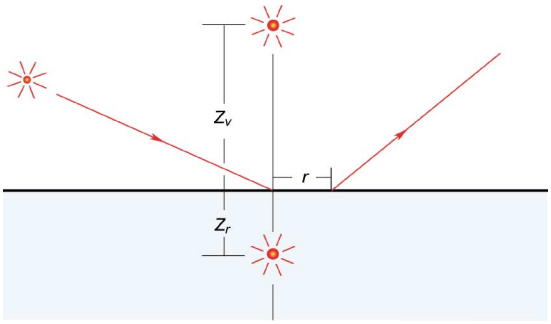


Fig. 2. Dipole approximation of multiple scattering (based on Jensen et al. [2001]).

The multiple scattering term, $S_d(x_i, \vec{\omega_i}; x_o, \vec{\omega_o})$, depends on the transmission terms at the entering and exiting surface points, and the dipole factor $R_d(r)$. We note that the dipole factor, $R_d(r)$, only depends on the distance between two points and decays exponentially with the distance. We define the *scattering neighborhood* $N(x_o)$ of a vertex x_o, to include all vertices x_i of that object that lie within the effective scattering range from x_o. We then compute multiple scattering contribution from the scattering neighborhood of each vertex during the preprocessing stage. Every such neighboring vertex x_i is assumed to represent a small surface area whose size can be approximately defined. We assign the integral of $R_d(\|x_i - x_o\|)$ over this small surface area as the contribution to the multiple scattering at x_o due to x_i and append this information to x_o's list of multiple scattering contributors. Then at rendering time, once we have $F_t(\eta, \vec{\omega_i})$ and $F_t(\eta, \vec{\omega_o})$ from the local illumination computation, the contribution of point x_i to x_o due to subsurface scattering is just the multiplication of $F_t(\eta, \vec{\omega_i})$ with $F_t(\eta, \vec{\omega_o})$ and the precomputed $R_d(\|x_i - x_o\|)$ factor of x_i from x_o's neighborhood list. The values of Fresnel terms and their associated relative indices of refraction that we used in our work can be found in Jensen et al. [2001]. The precomputation and storing of the dipole factors is similar to the approach taken by Lensch et al. [2002]. In their algorithm, instead of storing vertex-to-vertex dipole factors for every vertex in the scattering neighborhood, they distinguish between local responses and global responses. They store the global responses as vertex-to-vertex throughput factors, and the local ones as texture atlas. We have not made that distinction here, and store all of them as vertex-to-vertex factors.

2.3 Run-Time Two-Pass Local Illumination Model

We incorporate subsurface scattering effects into a local illumination model by extending the model into a run-time two-pass one. The traditional local illumination model computes the outgoing radiance from a surface point according to lighting direction, surface normal, and viewing direction in a single pass, using the particular light and material properties.

In our run-time two-pass approach, the first pass generates reflection and transmission radiance at each surface point as if there is no subsurface scattering, using the Fresnel terms for reflection or transmission. After we compute the illumination at all surface points, we come to the second pass, that is, the bleeding pass. During this pass, we combine on-surface reflection with subsurface scattering to get the total radiance at the exterior surface points according to the multiple scattering factors given in Section 2.2, using each point's weighted contributions from its neighbors. This bleeding pass adds subsurface reflection and transmission effects on the surface.

3. IMPROVING EFFICIENCY

Our run-time two-pass process is somewhat similar, but still quite different from the approach proposed by Jensen and Buhler [2002]. The main difference is when to compute the scattering neighborhood factors. We precompute the factors at the preprocessing stage instead of traversing a hierarchical N-body data structure for each frame as in Jensen and Buhler [2002], so bleeding the neighboring effects due to scattering in the second pass is quite efficient.

The run-time complexity of this version of our algorithm is $O(N^2)$, where N is the number of surface points, assuming the size of the object and the scattering properties remain constant. This is due to the fact that the number of vertices at which we have to perform the bleeding step is N, and the scattering neighborhood size is proportional to surface point density, which in turn is proportional to the number of surface points N. While Jensen and Buhler [2002] build a hierarchical $O(N \log N)$ data structure to solve the inherent $O(N^2)$ complexity problem, we propose a quantized light source scheme to merge the two stages of our run-time lighting process into a single-stage process to further improve the efficiency of our algorithm. We can thus reduce the complexity of our run-time algorithm to $O(N)$ with quite small constant factors. It enables us to achieve interactive frame rates for simulating subsurface scattering effects on larger datasets. However, the preprocessing also means that any change of the material subsurface scattering properties will require a new precomputation, which is a limitation not incurred by Jensen and Buhler [2002]. As we stated earlier, our model is an appearance-driven one using local illumination, so its accuracy sometimes can not exactly match the one proposed in Jensen and Buhler [2002].

3.1 Quantized Light Sources for Precomputed Neighborhood Factor

We make further simplifications to reduce the complexity of our algorithm based on the fact that each surface point in the neighborhood of another surface point represents a small area on the surface, and that real surfaces are usually rough. The subsurface scattering contribution to the appearance of a surface point from a directional light source with fixed direction ω_i can be preprocessed as follows:

$$\begin{aligned}
L_o(x_o, \overrightarrow{\omega_o}) &= \int_A S(x_i, \overrightarrow{\omega_i}; x_o, \overrightarrow{\omega_o}) L_i(x_i, \overrightarrow{\omega_i})(\overrightarrow{n_i} \cdot \overrightarrow{\omega_i}) \mu_i \, dA \\
&\approx \int_A S_d(x_i, \overrightarrow{\omega_i}; x_o, \overrightarrow{\omega_o}) L_i(x_i, \overrightarrow{\omega_i})(\overrightarrow{n_i} \cdot \overrightarrow{\omega_i}) \mu_i \, dA \\
&= \int_A F_t(\eta, \overrightarrow{\omega_i}) \left[\frac{1}{\pi} R_d(\|x_i - x_o\|) \right] F_t(\eta, \overrightarrow{\omega_o}) \cdot L_i(x_i, \overrightarrow{\omega_i})(\overrightarrow{n_i} \cdot \overrightarrow{\omega_i}) \mu_i \, dA \\
&= \left\{ \int_A F_t(\eta, \overrightarrow{\omega_i}) \left(\frac{1}{\pi} R_d \right) L_i(x_i, \overrightarrow{\omega_i}) \overrightarrow{n_i} \, \mu_i \, dA \right\} \cdot \overrightarrow{\omega_i} \cdot F_t(\eta, \overrightarrow{\omega_o}) \\
&\equiv \overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i}) \cdot \overrightarrow{\omega_i} \cdot F_t(\eta, \overrightarrow{\omega_o}),
\end{aligned}$$

where μ_i is defined as:

$$\mu_i = \begin{cases} 1 & (\overrightarrow{n_i} \cdot \overrightarrow{\omega_i}) \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

This means we can precompute the vector integral $\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i})$ for the scattering factor during the preprocessing stage, and at run-time perform the dot-product and multiplication operations. Due to the discrete nature of input mesh geometry, the vector integral above will be expressed as a vector

summation in a real implementation:

$$\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i}) = \sum_{x_i \in N(x_o)} F_t(\eta, \overrightarrow{\omega_i}) \left(\frac{1}{\pi} R_d\right) L_i(x_i, \overrightarrow{\omega_i}) \overrightarrow{n_i} \mu_i \Delta A(x_i),$$

where the summation is over all the vertices in the scattering neighborhood $N(x_o)$ of x_o. $\Delta A(x_i)$ is the area represented by vertex x_i, which is a constant if vertices are distributed uniformly as in Jensen and Buhler [2002]. For nonuniformly distributed vertices, we can either resample the geometry, or use one third of the total area of the triangles sharing the vertex as an approximation to ΔA at the vertex. So we actually precompute the summation $\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i})$ for each vertex. Note, if a vertex at x_i in the scattering neighborhood of x_o, is in shadow, then it will not contribute to $\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i})$, because x_i receives no direct irradiance from the light source. The summation will not be affected by the presence of shadow on x_o, though. We use a technique similar to shadow maps to determine if a vertex is in shadow. We first generate a depth image of the scene as seen by the light source. Then, for each vertex, we transform it into light space and compare its depth value against the value on the depth image. If the depth value of the vertex is bigger, the vertex is in shadow.

It will be impossible to compute the vector integral $\overrightarrow{Q}$ for each possible light source direction, of which the number is infinite. Instead, we quantize the directional space and precompute $\overrightarrow{Q}$ for a set of uniformly distributed light source directions. For each light source j within the set, we compute the scattering neighborhood integral $\overrightarrow{Q_j}$ at each vertex during the preprocessing stage. An alternative to precomputing and storing a vector integral $\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i})$ is to precompute a scalar dot-product value $q(\eta, x_o, \overrightarrow{\omega_i})$ instead:

$$\begin{aligned} L_o(x_o, \overrightarrow{\omega_o}) &= \left\{ \sum_{x_i \in N(x_o)} F_t(\eta, \overrightarrow{\omega_i}) \left(\frac{1}{\pi} R_d\right) L_i(x_i, \overrightarrow{\omega_i}) (\overrightarrow{n_i} \cdot \overrightarrow{\omega_i}) \mu_i \Delta A(x_i) \right\} \cdot F_t(\eta, \overrightarrow{\omega_o}) \\ &\equiv q(\eta, x_o, \overrightarrow{\omega_i}) \cdot F_t(\eta, \overrightarrow{\omega_o}). \end{aligned}$$

The advantage of using q instead of $\overrightarrow{Q}$ is the reduction of memory usage. The pseudo-code for precomputing $q(\eta, x_o, \overrightarrow{\omega_i})$ for vertex x_o is shown below (assume the area $\Delta A(x_i)$ and incoming flux $L_i(x_i, \overrightarrow{\omega_i})$ associated with each vertex x_i has been computed, and the effective scattering range is represented by $RANGE$):

```
Find–Scalar–Integral (η, x_o, ω_i)
    q = 0
    for i from 1 to N
        if (x_i == x_o OR x_i in shadow)
            skip
        else
            r = ‖x_i − x_o‖
            if (r > RANGE)
                skip
            else
                q += F_t(η, ω_i)(1/π R_d(r))L_i(x_i, ω_i)(n_i · ω_i)μ_i ΔA(x_i)
    return q.
```

It is clear that the preprocessing stage shown above has complexity of $O(N^2)$. If we use an octree-based data structure as in Jensen and Buhler [2002], then the complexity will go down to $O(N \log N)$.

This sampling of the lighting directional space to precompute a set of subsurface scattered radiance is similar, in spirit, to Debevec et al. [2000]. In their paper, they measure the reflectance field of a

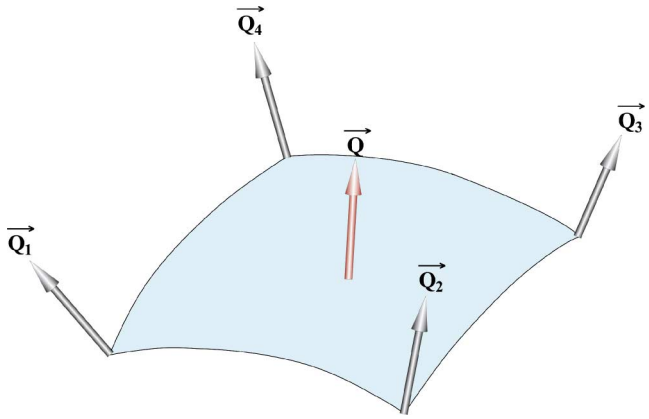


Fig. 3. Interpolation of the vector integral for a new light source direction from its four nearest neighbors in the precomputed set.

human face, by sampling densely the incident illumination directions (2000 lighting directions) and sparsely the viewpoints. The 8-D (eight-dimensional) reflectance field is equivalent to BSSRDF if the field surface is coincident with a physical surface [Debevec et al. 2000]. For non-local reflectance fields where the incident illumination field originates far from the surface, the reflectance field becomes 6-D (i.e., 2-D incoming light direction, 2-D exiting radiance surface point location, and 2-D viewing direction). Their reflectance field includes the on-surface reflected, as well as subsurface scattered radiance. With a clever technique of separating diffuse and specular parts, they are able to render interactively for any new lighting direction with fixed viewer, or several minutes per frame for moving viewer. What we have shown here is that, the subsurface scattered reflectance field is only 4-D once we take out the viewer-dependent Fresnel term, which can be added back easily at run-time. Furthermore, we show that the subsurface scattered reflectance field alone can be sampled at a much sparser rate (200 instead of 2,000 directions). All these enable us to interactively move both the viewer and the light source direction.

3.2 Rendering From Quantized Light Sources

After we precompute either the vector integral $\overrightarrow{Q}$ or scalar integral q for a set of directional light sources, we use interpolation at run-time to find the scattering integral $\overrightarrow{Q}$ or q for a specific light source direction. We use quaternion-based vector interpolation [Pletinckx 1989] to compute $\overrightarrow{Q}$ from its four closest $\overrightarrow{Q_j}$'s in the set (as in Figure 3). Then we compute dot-product of the interpolated scattering integral $\overrightarrow{Q}$ with real light source direction. This kind of interpolation is similar to the normal interpolation scheme used in Phong shading, though quaternion interpolation gives a more accurate result and avoids a vector renormalization step. To compute q, we simply use a linear scalar interpolation scheme, which is similar to the interpolation used in the Gouraud shading algorithm.

During the rendering of the scene, we combine scattering effects with direct on-surface reflected light (including shadow) to give the final appearance of each vertex. As an example, for a light source in direction $\overrightarrow{\omega_i}$, the scattering amount for vertex x_o along viewing direction $\overrightarrow{\omega_o}$ will be $F_t(\eta, \overrightarrow{\omega_o})$ multiplied with the precomputed factor $q(\eta, x_o, \overrightarrow{\omega_i})$, and scaled by this light source's actual intensity. We compute direct on-surface reflected light by a local illumination model.

The pseudo-code for computing the outgoing radiance $L(x_o, \overrightarrow{\omega_o})$ for vertex x_o in direction $\overrightarrow{\omega_o}$ appears below. Here we assume the use of scalar integral q.

(a) Scattering using $\overrightarrow{Q}$ (b) Scattering using q (c) Difference of (a) and (b)

Fig. 4. Comparison of subsurface scattering using precomputed vector integral and scalar integral on the horse model (14,521 vertices).

Find–Outgoing–Radiance $(\eta, \overrightarrow{\omega_i}, x_o, \overrightarrow{\omega_o})$
  $L(x_o, \overrightarrow{\omega_o}) = 0$
  Find 4 nearest matches q_j from the pre-computed set of $\{q\}$ for x_o
  Interpolate the 4 matched values based on $\overrightarrow{\omega_i}$ to get $q(\overrightarrow{\omega_i})$
  $L_{scattered} = q(\overrightarrow{\omega_i}) \cdot F_t(\eta, \overrightarrow{\omega_o})$
  Compute reflected radiance $L_{reflected}(x_o, \overrightarrow{\omega_o})$ using a local illumination model
  $L(x_o, \overrightarrow{\omega_o}) = L_{scattered} + L_{reflected}(x_o, \overrightarrow{\omega_o})$
  return $L(x_o, \overrightarrow{\omega_o})$.

The visual difference between using $\overrightarrow{Q}$ and q for the models we have tested is insignificant. This can be attributed to the diffuse nature of subsurface scattering. Hence we are currently using the precomputed scalar dot-products. Figure 4(a) shows a image generated using $\overrightarrow{Q}$ on a horse model, and Figure 4(b) shows the image generated using q on the same model. The difference image is shown in Figure 4(c). The image space root-mean-square error between Figure 4(a) and 4(b) is 5.26×10^{-3}.

What we have shown is that the light flux at a vertex on the surface due to direct reflection and subsurface scattering can now be computed at the same time under a local illumination model. Thus, with precomputed integral, the run-time two-pass algorithm we suggested before now becomes a run-time single-pass algorithm. Furthermore, this precomputed integral scheme also indicates that the run-time computation of the scattering effect on a vertex is just an interpolation of the four nearest neighbors in the set of the precomputed integrals which have the same size as the light source set we have selected. So the complexity of computing the scattering component at run-time is constant, and not related to surface point density. The total complexity of our run-time algorithm becomes $O(N)$, instead of $O(N^2)$, where N is the number of vertices. Subsurface scattering increases if the translucency of the material increases or the physical size of the object decreases. This increases the scattering neighborhood size that needs to be considered. However, since the scattering neighborhood size only affects the precomputation of integrals $\overrightarrow{Q}$ or q, the rendering-time complexity of our display algorithm stays $O(N)$.

3.3 Determining the Size of the Light Source Set

The above sections show that if we use a quantized light source scheme for precomputation of scattering integrals and do interpolation at run-time, then we will have a linear complexity single-pass run-time algorithm for rendering translucent materials. We have not yet mentioned how to pick the size of the light source set.

As we know, the scattering integral $\overrightarrow{Q}(\eta, x_o, \overrightarrow{\omega_i})$ or $q(\eta, x_o, \overrightarrow{\omega_i})$ is a continuous function of the directional space variable $\overrightarrow{\omega_i}$. Quantization of light source directions is a sampling process and interpolation is a reconstruction process. Similar to other sampling processes, there is a tradeoff between the sampling

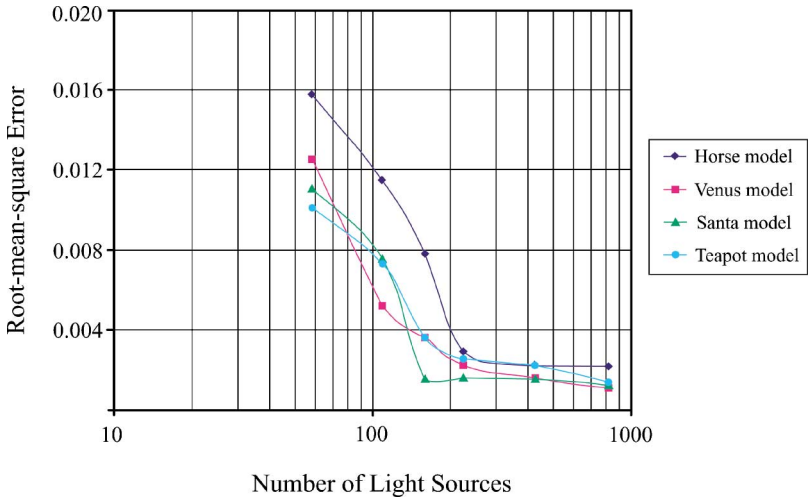


Fig. 5. Root-mean-square error as a function of the number of light sources.

rate and the time and storage needed to process them. Using lower sampling rate is time and memory efficient, but gives us less accurate results. Even worse, a low sampling rate may introduce aliasing problems when the sampling frequency is lower than the Nyquist rate. A general frequency-space analysis for the scattering integrals is difficult because the scattering integrals depend on geometry, and scattering properties of the object and different vertices will have different frequency distributions.

We instead experiment with different sizes of the light source set. We measure the image space root-mean-square error for our test datasets and pick the smallest size which introduces little visual artifacts:

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2 \right]^{1/2}.$$

Here $f(x, y)$ represents the image generated without the specified approximation, $\hat{f}(x, y)$ denotes an estimate of the image, either interpolated or approximated version of the image, $M \times N$ is the image size, and the range for $f(x, y)$ is [0, 1].

The results of the experiment are summarized in Figure 5. The root-mean-square error is measured by comparing the results obtained by interpolation using precomputed scalar integrals q with the exact results. We compute this error for about 100 randomly-generated view directions and take the maximum RMS error as the representative. We can see from Figure 5 that with a set of about 200 light sources, the root-mean-square error is 3×10^{-3} for all the four datasets. So we use 200 light source directions to precompute the scalar integrals q.

This directional quantization scheme can also be extended to include point light sources. We can add one more dimension to the interpolation, that is, we quantize the distance from the light source to the object along with quantization of its direction. Then we can trilinearly interpolate 8 nearest neighbors to get an $O(N)$ complexity algorithm for directional and point light sources.

Here we limit ourselves to local illumination, so we ignore on-surface inter-reflections between vertices during computation of the precomputed integral $q(\eta, x_o, \overrightarrow{\omega_i})$. If we use ray-tracing or Monte Carlo simulation in the preprocessing stage, we can incorporate it in our algorithm and get more accurate $q(\eta, x_o, \overrightarrow{\omega_i})$.

4. CONTROLLING THE MEMORY USAGE

For a set of 200 lights, we need to store 200 integrals per vertex. Instead of storing a floating-point value per integral, we store a normalized unsigned byte value to serve as an index to a lookup table. Thus, we need 200 bytes of extra storage per vertex. We have used the Lloyd quantizer algorithm [Gersho and Gray 1992] to design the lookup table. As an example of the quantized result, the signal-to-noise ratio, SNR [Gersho and Gray 1992] is 50.99 dB for precomputed integrals of the teapot dataset by using this quantization, with a resulting image space root-mean-square error of 8.83×10^{-4}. Normally, at each vertex we need to store three numbers each for position and normal direction, and other numbers such as texture coordinates and color, if any. If we assume floating-point numbers to store these values, we will need 24 bytes to store the position and normal direction alone. Even with this uncompressed number, the extra storage needed for precomputed integrals will increase it by a factor of 8, which is quite a disadvantage of using our algorithm. The number can be reduced though. In the following sections, we show how to dramatically reduce this number so that the extra storage is comparable with the original storage required for the vertex data.

4.1 Decomposition By Spherical Harmonic Basis Functions

Due to the diffuse-like nature of subsurface scattering effects, we apply spherical harmonic functions as used by Basri and Jacobs [2001], Cabral et al. [1987], Kajiya and Von Herzen [1984], Ramamoorthi and Hanrahan [2001, 2002], Sillion et al. [1991], Sloan et al. [2002], Sloan et al. [2003], and Westin et al. [1992]. to compress the directional integrals. Spherical harmonic functions form an efficient basis to represent functions defined over the directional space, such as incident radiance, BRDFs [Basri and Jacobs 2001; Cabral et al. 1987; Ramamoorthi and Hanrahan 2001, 2002; Sillion et al. 1991; Sloan et al. 2002, 2003; Westin et al. 1992], and phase function of particles in clouds [Kajiya and Von Herzen 1984]. Using the spherical harmonic functions for expansion and storing the coefficients up to a given order is similar to a filtering process of the angularly distributed signal [Ramamoorthi and Hanrahan 2001].

The real-valued spherical harmonic basis functions [Sloan et al. 2002] are:

$$y_l^m = \begin{cases} \sqrt{2} K_l^m \cos(m\varphi) P_l^m(\cos\theta), & m > 0 \\ \sqrt{2} K_l^m \sin(-m\varphi) P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0, \end{cases}$$

where $l \in N$, $-l \le m \le l$, P_l^m are the associated Legendre polynomials, and K_l^m are the normalization constants:

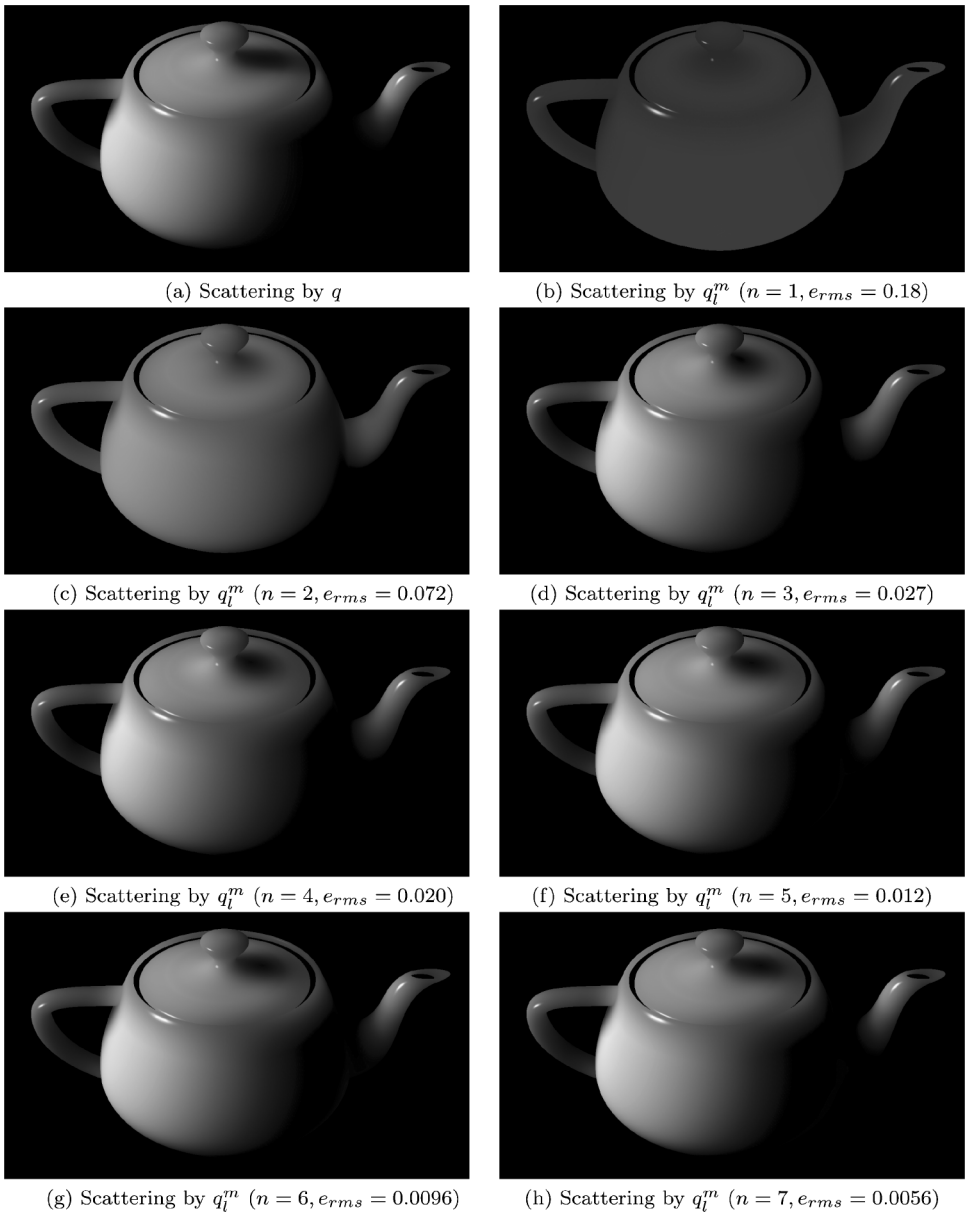
$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}}.$$

The projection of the precomputed scalar integral of $q(\eta, x_o, \overrightarrow{\omega_i})$ onto the spherical harmonic basis is given by:

$$q_l^m(\eta, x_o) = \int q(\eta, x_o, \overrightarrow{\omega_i}) y_l^m(\overrightarrow{\omega_i}) d\overrightarrow{\omega_i}.$$

The reconstructed function up to the nth order is:

$$\tilde{q}(\eta, x_o, \overrightarrow{\omega_i}) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} q_l^m(\eta, x_o) y_l^m(\overrightarrow{\omega_i}),$$



(a) Scattering by q (b) Scattering by q_l^m $(n = 1, e_{rms} = 0.18)$

(c) Scattering by q_l^m $(n = 2, e_{rms} = 0.072)$ (d) Scattering by q_l^m $(n = 3, e_{rms} = 0.027)$

(e) Scattering by q_l^m $(n = 4, e_{rms} = 0.020)$ (f) Scattering by q_l^m $(n = 5, e_{rms} = 0.012)$

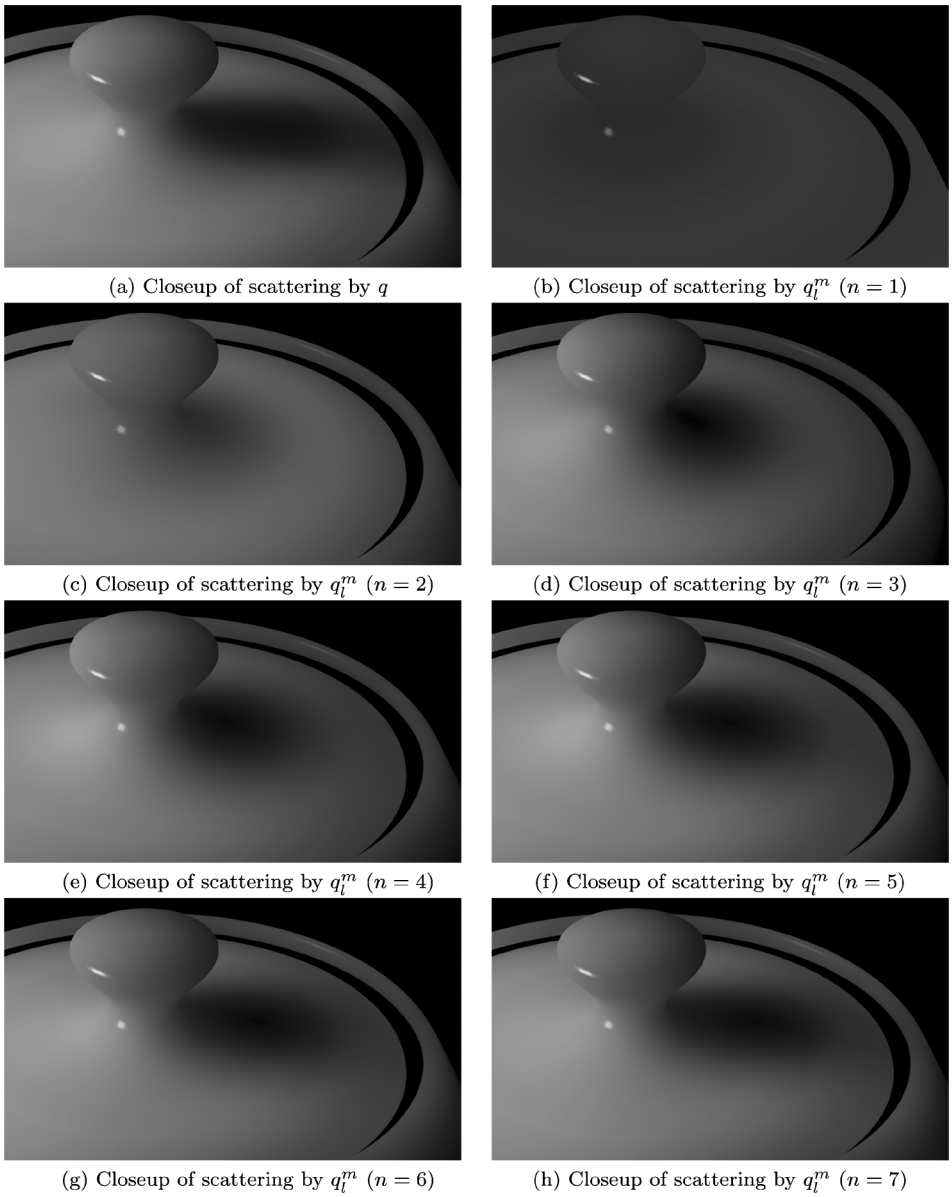
(g) Scattering by q_l^m $(n = 6, e_{rms} = 0.0096)$ (h) Scattering by q_l^m $(n = 7, e_{rms} = 0.0056)$

Fig. 6. Comparison of subsurface scattered teapot using q and q_l^m (150,510 vertices).

where

$$\overrightarrow{\omega_i} = (x, y, z) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta).$$

As an example, we apply the above projection and reconstruction scheme to the subsurface scattered teapot and results are shown in Figure 6, as the order n varies from 1 to 7. Closeup versions are shown in Figure 7. The number of the basis functions is equal to n^2, which results in 1 to 49 basis functions used.



(a) Closeup of scattering by q

(b) Closeup of scattering by q_l^m $(n = 1)$

(c) Closeup of scattering by q_l^m $(n = 2)$

(d) Closeup of scattering by q_l^m $(n = 3)$

(e) Closeup of scattering by q_l^m $(n = 4)$

(f) Closeup of scattering by q_l^m $(n = 5)$

(g) Closeup of scattering by q_l^m $(n = 6)$

(h) Closeup of scattering by q_l^m $(n = 7)$

Fig. 7. Closeup of Figure 6.

For each basis function, we store a normalized short integer value (2 bytes) for $q_l^m(\eta, x_o)$. We therefore need 98 bytes per vertex for $n = 7$. We have not gone to higher n because then the storage required becomes comparable to the method that does not use spherical harmonics. From Figure 6 and Figure 7, we can see that the image quality increases with the number of spherical basis functions. With 49 basis functions, the visual quality is close to the one without compression. However, if one notices carefully, some differences near the shadow boundaries are still visible (Figure 7(a) and (h)). The reason is that the spatial frequency of the precomputed integral q is beyond the spatial frequency that 49 spherical

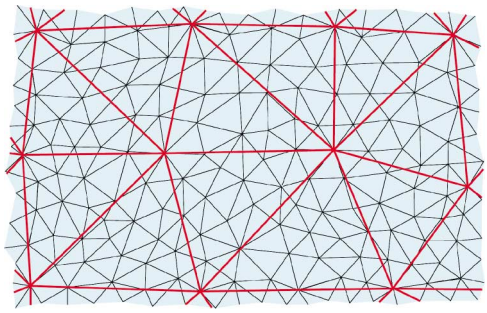


Fig. 8. Construction of reference points.

harmonic basis functions can completely cover. So with spherical harmonics, we can achieve a factor of two compression ratio with small loss of image quality. For low frequency lighting environments, an interesting alternative is to use Clustered Principal Component Analysis (CPCA)-based compression of spherical harmonic coefficients to achieve faster rendering [Sloan et al. 2003]. For general lighting environments, we have to find some way to suppress the spatial frequency of q.

4.2 Reference Points With Spherical Harmonic Basis Functions

We observe that the scattering from light entering one particular vertex and exiting at two other points will not differ much if those two points are close to each other. This is due to the diffuse nature of multiple scattering. The fact that each point receives contributions from all the vertices in its scattering neighborhood will smooth out the difference even further. This means that the difference of the scattering integrals Δq between nearby points will have much lower spatial frequency. So we can pick some reference vertices across the surface and store their scattering integrals q explicitly. For other vertices, we compute differences of the integrals by subtracting their original values from a weighted average of the values from its closest neighboring reference vertices. We can then expect that the spherical harmonic functions can be applied readily to those frequency-suppressed Δq. Ramamoorthi and Hanrahan [2002] have treated similar problem with a different perspective. Instead of trying to compress the frequency before applying the spherical decomposition, they determine the necessary number of basis functions for a faithful representation of the original signal using a signal-processing framework.

We can build reference points using a mesh simplification algorithm similar to Cohen et al. [1996], Garland and Heckbert [1997], and Hoppe [1996] or the retiling scheme of Turk [1992]. We prefer to generate the reference points as a subset of the original vertices to reduce the storage overhead (as shown in Figure 8). After we find the reference points, we generate the differences of precomputed integrals for each vertex with respect to its reference points as discussed next.

We first determine the three reference points for each vertex. Retiling schemes such as the one by Turk [1992] keep track of which triangle each vertex has been flattened to. For other mesh simplification algorithms we know one reference point for the vertex, which is its parent in the simplification hierarchy. The vertex will lie in one of the simplified triangles sharing this reference point. To find out the triangle the vertex lies in, we project the vertex onto planes defined by those triangles, then do a simple orientation test of the projected vertex relative to the three edges of each triangle. Once we find the triangle the vertex V lies in, we compute the barycentric coordinates of the projection V' of V onto the triangle.

Assume the reference points are V_1, V_2, and V_3 with barycentric coordinates w_1, w_2, and w_3. Let $\{q_{1j}\}, \{q_{2j}\}$, and $\{q_{3j}\}$ (j is the quantized light source index) be the precomputed integrals for V_1, V_2,

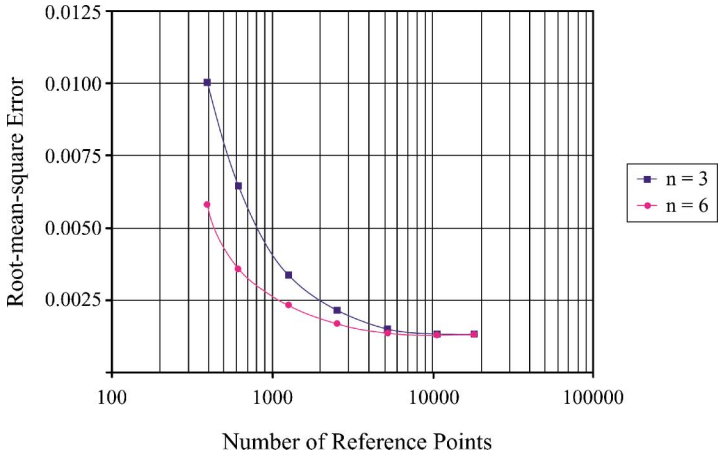


Fig. 9. Root-mean-square error as a function of the number of reference points for teapot dataset with 9($n = 3$) and 36($n = 6$) spherical harmonic basis functions.

and V_3, respectively. The $\{\Delta q_j\}$ set for vertex V can then be computed as:

$$\Delta q_j = q_j - \sum_{k=1}^{3} \omega_k q_{kj}.$$

Finally, we decompose the integral differences $\{\Delta q_j\}$ by spherical harmonic basis functions as before.

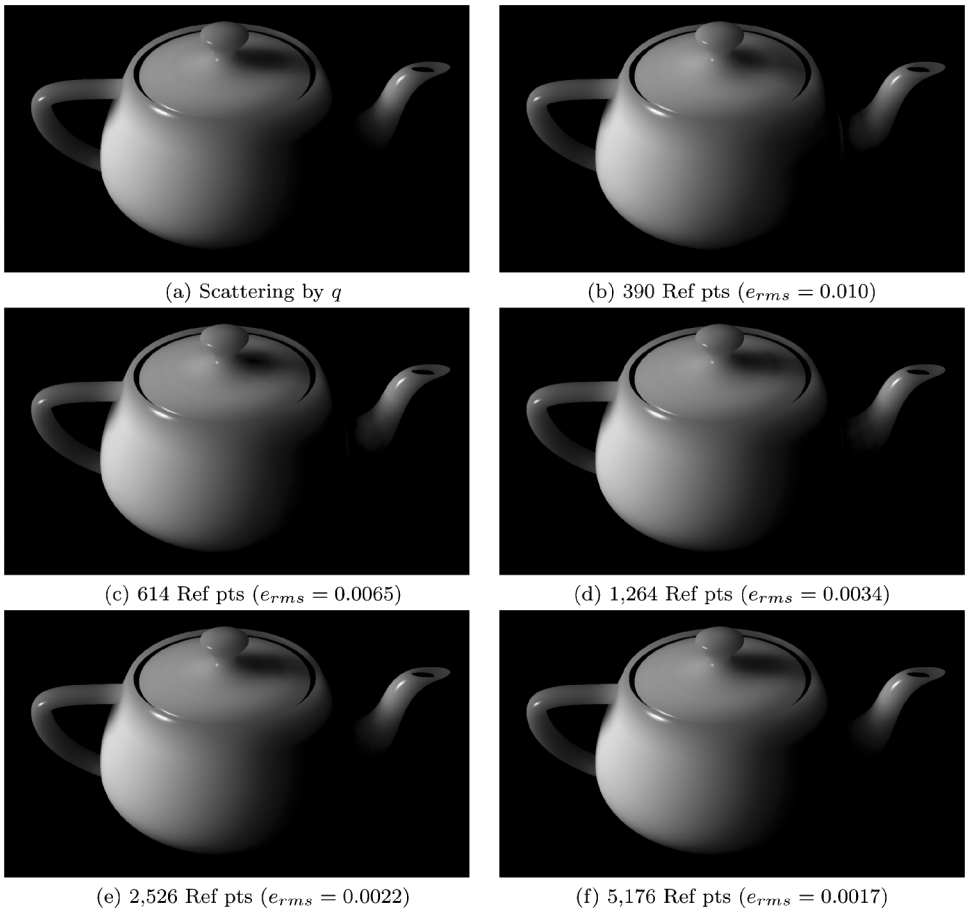
Figure 9 shows the root-mean-square error of using different numbers of reference points and with 9($n = 3$) and 36($n = 6$) spherical harmonic basis functions on the teapot dataset.

Figure 10 shows the scattered teapot images generated using different numbers of reference points with 9($n = 3$) spherical harmonic basis functions. Closeup versions are shown in Figure 11. From Figure 11, we can see that even with about 1,200 reference vertices and 9($n = 3$) basis functions (Figure 11(d)), the result is better (with smaller root-mean-square error) than the one using 49($n = 7$) basis functions alone (Figure 7(h)). For 5K reference vertices (about 3% of the total) and 9($n = 3$) basis functions, the image is almost indistinguishable from the original one (Figure 11(a)), even for the closeup version.

Now let us consider the storage requirements for the above case. We need 200 bytes for each vertex in the 5K reference set to store their original precomputed integrals, 4 bytes for each vertex to store its weight to its three nearest neighbors (the first two values stored as normalized short integers, while the third value can be computed at run-time by one minus the first two values), and 9 bytes for each vertex to store the spherical harmonic basis functions' coefficients (each stored as a normalized byte because the range for the coefficients has also been reduced a lot). Overall, on average we need the following number of bytes per vertex to store the precomputed integrals:

$$\frac{200 \times 5176 + (4 + 9) \times 150510}{150510} \approx 20.$$

So we only need 20 bytes per vertex. We know that each vertex needs a position vector and a normal vector. If we assume floating-point numbers to store them, we will need $(3 + 3) \times 4 = 24$ bytes for each vertex. Then, the storage required by the precomputed integrals is less than the storage required by the position and normal alone. Of course, one can compress positions and normals for vertices, too.



(a) Scattering by q (b) 390 Ref pts ($e_{rms} = 0.010$)

(c) 614 Ref pts ($e_{rms} = 0.0065$) (d) 1,264 Ref pts ($e_{rms} = 0.0034$)

(e) 2,526 Ref pts ($e_{rms} = 0.0022$) (f) 5,176 Ref pts ($e_{rms} = 0.0017$)

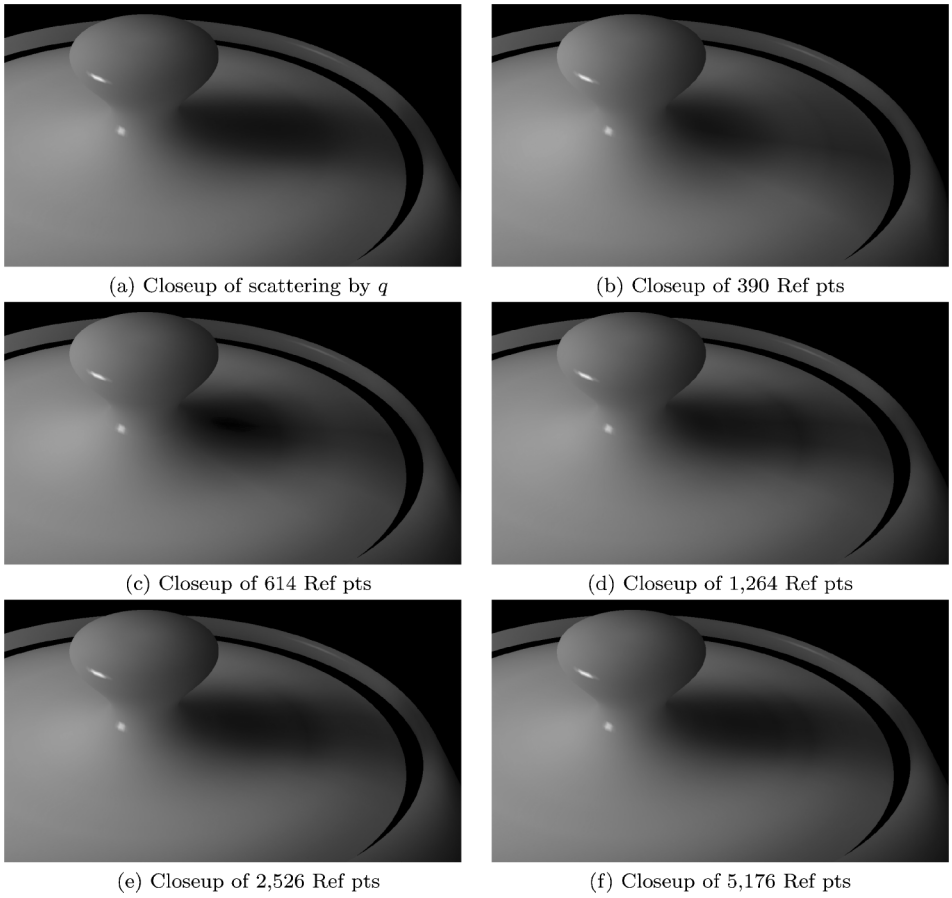
Fig. 10. Comparison of subsurface scattered teapot using q and different numbers of reference points with 9($n = 3$) spherical harmonic basis functions (150, 510 vertices).

Nevertheless, this storage overhead seems quite reasonable for the interactive simulation of translucent materials. Even better, the rendering speed increases from 7.5 frames per second to 8.6 frames per second, which is about a 15% speedup. We achieve similar results on other datasets we have tested. The extra storage will be no more than 27 bytes per vertex, and it decreases as the complexity of the object increases (Table I).

5. RESULTS AND DISCUSSIONS

In this section, we show the results obtained by our algorithm on polygonal datasets. We have used a 2 GHz Pentium 4 PC, running Windows 2000 with a nVIDIA GeForce3 graphics card. The results are summarized in Table I and in Figures 6, 7, and 9–14. The images usually have about 1024 pixels in each dimension, though their sizes have nearly no effect on the total rendering time because we use the graphics hardware mainly to do rasterization.

As one can see from Table I, our scattering model can generate subsurface scattered images within a few tenths of a second for a model with over one million triangles, and achieve interactive frame rates for objects with less than 300K triangles. We compute the BSSRDF for all the sample vertices as in



(a) Closeup of scattering by q (b) Closeup of 390 Ref pts

(c) Closeup of 614 Ref pts (d) Closeup of 1,264 Ref pts

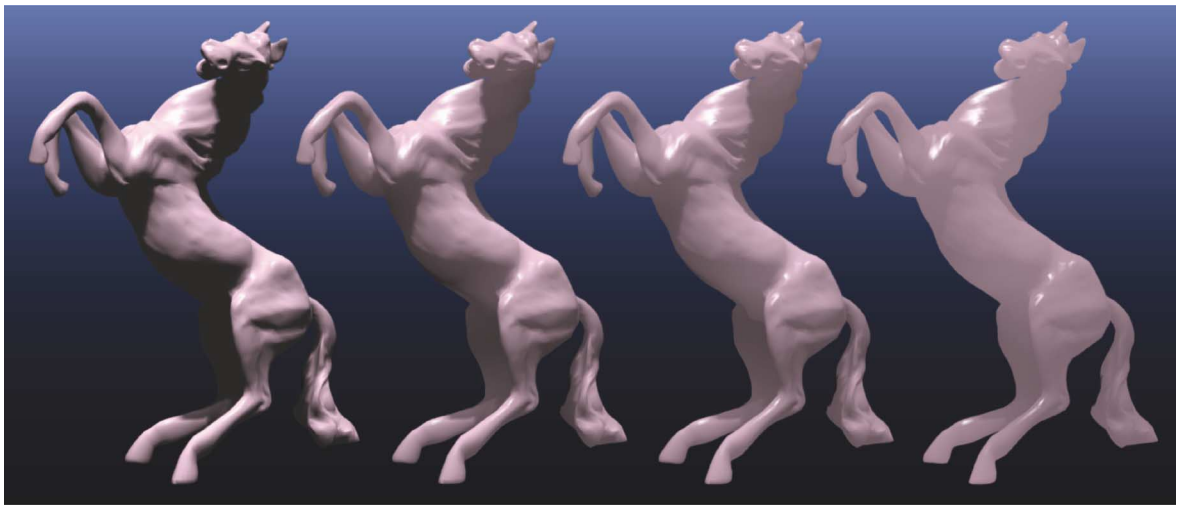
(e) Closeup of 2,526 Ref pts (f) Closeup of 5,176 Ref pts

Fig. 11. Closeup of Figure 10.

Table I. Total Rendering Times for Our Approach

Model Name	No. of Vertices	No. of Triangles	No. of ref pts	Extra storage (Bytes/vert)	Compression ratio by using ref pts	Frame rate(fps)	
						with scattering	without scattering
Horse	14,521	29,054	1,034	27	7.4	79.1	181
Venus	42,656	90,044	2,827	26	7.7	27.3	62.5
Santa	75,781	151,558	3,458	22	9.1	14.6	31.7
Teapot	150,510	292,168	5,176	20	10.0	8.6	19.5
Dragon	437,645	871,414	10,285	18	11.1	2.7	6.3
Buddha	543,652	1,087,716	12,330	18	11.1	2.4	5.2

Table I. The subsurface scattered color of vertices is computed on the CPU, while the non-scattered color is computed on the GPU. All the vertex data (position, color or normal) are uploaded to the GPU at each frame. The GPU memory has not been exploited. Our algorithm has an effective $O(N)$ complexity (N is the number of vertices), with small constant factors. The extra storage for precomputed integrals is less than 28 bytes per vertex using the scheme specified in Section 4.2. This small overhead should give most applications the opportunity to include the subsurface scattering effects for more photo-realistic rendering without sacrificing interactive frame rates.



(a) Without Scattering (181 fps) (b)(c)(d) With scattering (79.1 fps)

Fig. 12. Rendering the horse model with subsurface scattering increasing from left to right (14, 521 vertices with 10% vertices in $N(x_o)$ at (b), 20% vertices in $N(x_o)$ at (c), and 30% vertices in $N(x_o)$ at (d)).

Fig. 13. Rendering the subsurface scattered teapot model with varying light source direction (150, 510 vertices, 8.6 fps).

Figure 12 shows increasing subsurface scattering effects on a horse model from left to right. Figure 13 shows the effects of varying light source direction with fixed viewer position on subsurface scattered teapot. We have used the Perlin noise function [Ebert et al. 2003; Perlin 1985] to generate the marble texture on the Venus model. Here we have made the assumption that the marble texture is on-surface, and will affect both x_i and x_o. Figures 12 and 14 show how the object will appear if either its size shrinks or its material property changes to allow greater subsurface scattering.

Our algorithm can also use a full Monte Carlo simulation in the preprocessing stage. This will allow us to not only have an accurate subsurface scattering term, but we can then also include the single scattering term, treat inhomogeneity, and relieve the algorithm from the limitation of the dipole diffusion approximations for multiple scattering. Subsurface scattering is also characterized by color-shift effects. The correct way to simulate color shifts is to do a full spectral rendering. However, the three channel

(a) Without Scattering (62.5 fps)

(b)(c)(d) With scattering (27.3 fps)

Fig. 14. Rendering the Venus model with subsurface scattering increasing from left to right (42, 656 vertices with 10% vertices in $N(x_o)$ at (b), 20% vertices in $N(x_o)$ at (c), and 30% vertices in $N(x_o)$ at (d)).

RGB approximation can also give visually appealing results. If one would like to use the three-channel approximation of the subsurface scattered color shifts in our algorithm, we can compute three different sets of integrals, one for each channel. The storage requirements will then be a little less than three times, because we only need to store the barycentric coordinates once, instead of three times. That means we will need about 46 bytes extra per vertex for large datasets.

6. CONCLUSIONS AND FUTURE WORK

In this article, we integrate subsurface scattering effects into a run-time single-pass local illumination model with an efficient $O(N)$ run-time complexity using precomputed scattering integrals for a set of quantized light source directions. We also show that a reference points scheme, together with spherical harmonics can be applied to greatly reduce the storage requirements of precomputed integrals and improve the run-time efficiency of our algorithm even further. The results capture the most important effects of subsurface scattering, such as neighborhood bleeding and smooth illumination transitions between regions separated by sharp edges. Our method provides an approximation of subsurface scattering for applications that need to maintain the interactivity with a small memory overhead while preserving the realistic appearance for translucent materials. Our approach,

with a little modification, can also be incorporated into shadow algorithms to generate soft-shadow effects.

One direction for future work is to optimize the algorithm for determining the reference points, given the desired number of points to be referenced and the precomputed sets of integrals. Currently, we use simplification or retiling schemes, which only take geometry into account. Factors related to illumination, such as shadows, have not yet been fully considered during the search for reference points. Determination of reference points based on both geometry and illumination will likely result in a better tradeoff between the image quality and the compression ratio. It will also be interesting to determine the linear combination weights that minimize error instead of just using the spatial barycentric coordinates by solving a constrained optimization problem.

Another interesting direction is to take advantage of current advances in graphics hardware technology to implement the rendering stage on the graphics card itself using pixel and vertex shaders. This will not only allow much faster rendering speeds, but might also result in insights to inspire further algorithmic improvements in subsurface scattering.

ACKNOWLEDGMENTS

We would like to thank Henrik Wann Jensen, Pat Hanrahan, and Juan Buhler for providing the lighting and material parameters for generating the images in this article. We would like to acknowledge Cyberware for providing us the Santa model. We would also like to acknowledge the anonymous referees for their detailed and constructive comments which have led to a much better presentation of our results.

REFERENCES

Ashikhmin, M., Premoze, S., and Shirley, P. 2000. A microfacet-based BRDF generator. In *Proceedings of SIGGRAPH 2000*. 65–74.

Bahar, E. and Chakrabarti, S. 1987. Full-wave theory applied to computer-aided graphics for 3D objects. *IEEE Comput. Graph. Appl. 7*, 7, 46–60.

Basri, R. and Jacobs, D. 2001. Lambertian reflectance and linear subspaces. In *Proceedings of the International Conference on Computer Vision*. 383–390.

Blinn, J. F. 1977. Models of light reflection for computer graphics. *Comput. Graph. 11*, 2, 192–198.

Cabral, B., Max, N., and Springmeyer, R. 1987. Bidirectional reflection functions from surface bump maps. *Computer Graphics* (*Proceedings of SIGGRAPH '87*). ACM, New York, *21*, 4, 273–281.

Carr, N. A., Hall, J. D., and Hart, J. C. 2003. GPU algorithms for radiosity and subsurface scattering. In *Proceedings of Graphics Hardware 2003*. 51–59.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, Jr., F. P., and Wright, W. 1996. Simplification envelopes. In *Proceedings of SIGGRAPH '96*. ACM, New York, 119–128.

Cook, R. L. and Torrance, K. E. 1981. A reflectance model for computer graphics. *Computer Graphics* (*Proceedings of SIGGRAPH '81*). ACM, New York, *15*, 3, 307–316.

Dachsbacher, C. and Stamminger, M. 2003. Translucent shadow maps. In *Proceedings of the 14th Eurographics Symposium on Rendering*. 197–201.

Debevec, P., Hawkins, T., Tchou, C., Duiker, H., Sarokin, W., and Sagar, M. 2000. Acquiring the reflectance field of a human face. In *Proceedings of SIGGRAPH 2000*. ACM, New York, 145–156.

Dorsey, J., Edelman, A., Legakis, J., Jensen, H. W., and Pedersen, H. K. 1999. Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH '99*. ACM, New York, 225–234.

Ebert, D., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. 2003. *Texturing and Modeling*, 3rd ed. Morgan Kaufmann, San Francisco, CA.

Garland, M. and Heckbert, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH '97*. ACM, New York, 209–216.

Gersho, A. and Gray, R. M. 1992. *Vector Quantization and Signal Compression*. Kluwer Academic, Boston, MA.

GREENBERG, D. P., TORRANCE, K. E., SHIRLEY, P., ARVO, J., FERWERDA, J. A., PATTANAIK, S., LAFORTUNE, E. P. F., WALTER, B., FOO, S.-C., AND TRUMBORE, B. 1997. A framework for realistic image synthesis. In *Proceedings of SIGGRAPH '97*. ACM, New York, 477–494.

HANRAHAN, P. AND KRUEGER, W. 1993. Reflection from layered surfaces due to subsurface scattering. *Computer Graphics* (*Proceedings of SIGGRAPH '93*). ACM, New York, *27*, 3, 165–174.

HAO, X., BABY, T., AND VARSHNEY, A. 2003. Interactive subsurface scattering for translucent meshes. In *Proceedings of 2003 ACM Symposium on Interactive 3D Graphics*. ACM, New York, 75–82.

HE, X. D., TORRANCE, K. E., SILLION, F. X., AND GREENBERG, D. P. 1991. A comprehensive physical model for light reflection. *Computer Graphics* (*Proceedings of SIGGRAPH '91*). ACM, New York *25*, 4, 175–186.

HOPPE, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH '96*. ACM, New York, 99–108.

JENSEN, H. W. AND BUHLER, J. 2002. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics* (*Proceedings of SIGGRAPH 2002*). ACM, New York, *21*, 3, 576–581

JENSEN, H. W., LEGAKIS, J., AND DORSEY, J. 1999. Rendering of wet materials. In *Rendering Techniques '99*. Springer Verlag, New York, 273–282.

JENSEN, H. W., MARSCHNER, S., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*. ACM, New York, 511–518.

KAJIYA, J. T. 1985. Anisotropic reflection models. *Computer Graphics* (*Proceedings of SIGGRAPH '85*). ACM, New York, *19*, 3, 15–21.

KAJIYA, J. T. AND VON HERZEN, B. P. 1984. Ray tracing volume densities. *Computer Graphics* (*Proceedings of SIGGRAPH '84*). *18*, 3, 165–174.

KOENDERINK, J. J. AND VAN DOORN, A. J. 2001. Shading in the case of translucent objects. In *Proceedings of SPIE*. Vol. 4299. 312–320.

LENSCH, H., GOSELE, M., BEKAERT, P., KAUTZ, J., MAGNOR, M., LANG, J., AND SEIDEL, H.-P. 2002. Interactive rendering of translucent objects. In *Proceedings of Pacific Graphics 2002*. 214–224.

MARSCHNER, S. R., WESTIN, S. H., LAFORTUNE, E. P. F., TORRANCE, K. E., AND GREENBERG, D. P. 1999. Image-based BRDF measurement including human skin. In *Rendering Techniques '99*. Springer Verlag, 139–152.

MERTENS, T., KAUTZ, J., BEKAERT, P., SEIDEL, H.-P., AND VAN REETH, F. 2003. Interactive rendering of translucent deformable objects. In *Proceedings of the 14th Eurographics Symposium on Rendering*. 130–140.

PERLIN, K. 1985. An image synthesizer. *Computer Graphics* (*Proceedings of SIGGRAPH '85*). ACM, New York, *19*, 3, 287–296.

PHARR, M. AND HANRAHAN, P. 2000. Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proceedings of SIGGRAPH 2000*. ACM, New York, 75–84.

PHONG, B.-T. 1975. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June), 311–317.

PLETINCKX, D. 1989. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer 5*, 1/2, 2–13.

RAMAMOORTHI, R. AND HANRAHAN, P. 2001. A signal-processing framework for inverse rendering. In *Proceedings of SIGGRAPH 2001*. ACM, New York, 117–128.

RAMAMOORTHI, R. AND HANRAHAN, P. 2002. Frequency space environment map rendering. *ACM Transactions on Graphics* (*Proceedings of SIGGRAPH 2002*). ACM, New York, *21*, 3, 517–526.

SATO, Y., WHEELER, M. D., AND IKEUCHI, K. 1997. Object shape and reflectance modeling from observation. In *Proceedings of SIGGRAPH '97*. ACM, New York, 379–388.

SILLION, F. X., ARVO, J. R., WESTIN, S. H., AND GREENBERG, D. P. 1991. A global illumination solution for general reflectance distributions. *Computer Graphics* (*Proceedings of SIGGRAPH '91*). ACM, New York, *25*, 4, 187–196.

SLOAN, P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics* (*Proceedings of SIGGRAPH 2003*). *22*, 3, 382–391.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics* (*Proceedings of SIGGRAPH 2002*). ACM, New York, *21*, 3, 527–536.

STAM, J. 1999. Diffraction shaders. In *Proceedings of SIGGRAPH '99*. ACM, New York, 101–110.

STAM, J. 2001. An illumination model for a skin layer bounded by rough surfaces. In *Rendering Techniques '01*. Springer Verlag, ACM, New York, 39–52.

SUN, Y., FRACCHIA, F. D., DREW, M. S., AND CALVERT, T. W. 2000. Rendering iridescent colors of optical disks. In *Rendering Techniques '00*. Springer Verlag, 341–352.

TURK, G. 1992. Re-tiling polygonal surfaces. *Computer Graphics* (*Proceedings of SIGGRAPH '92*). ACM, New York, *26*, 2, 55–64.

WARD, G. J. 1992. Measuring and modeling anisotropic reflection. *Computer Graphics* (*Proceedings of SIGGRAPH '92*). ACM, New York, *26*, 2, 265–272.

WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. 1992. Predicting reflectance functions from complex surfaces. *Computer Graphics* (*Proceedings of SIGGRAPH '92*). ACM, New York, *26*, 2, 255–264.

YU, Y., DEBEVEC, P., MALIK, J., AND HAWKINS, T. 1999. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proceedings of SIGGRAPH '99*. ACM, New York, 215–224.

Received April 2003; revised October 2003; accepted December 2003