EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

Real-Time Translucent Rendering Using GPU-based Texture Space Importance Sampling

Chih-Wen Chang, Wen-Chieh Lin, Tan-Chi Ho, Tsung-Shian Huang and Jung-Hong Chuang

Department of Computer Science, National Chiao Tung University, Taiwan

Abstract

*We present a novel approach for real-time rendering of translucent surfaces. The computation of subsurface scattering is performed by first converting the integration over the 3D model surface into an integration over a 2D texture space and then applying importance sampling based on the irradiance stored in the texture. Such a conversion leads to a feasible GPU implementation and makes real-time frame rate possible. Our implementation shows that plausible images can be rendered in real time for complex translucent models with dynamic light and material properties. For objects with more apparent local effect, our approach generally requires more samples that may downgrade the frame rate. To deal with this case, we decompose the integration into two parts, one for local effect and the other for global effect, which are evaluated by the combination of available methods [DS03, MKB*03a] and our texture space importance sampling, respectively. Such a hybrid scheme is able to steadily render the translucent effect in real time with a fixed amount of samples.*

Keyword: BSSRDF, Translucent Rendering, Texture Space Importance Sampling

1. Introduction

Translucent materials are commonly seen in real world, e.g., snow, leaves, paper, wax, human skin, marble, and jade. When light transmits into a translucent object, it can be absorbed and scattered inside the object. This phenomenon is called subsurface scattering. Subsurface scattering diffuses the scattered light and blurs the appearance of geometry details, which is known as local effect. Therefore, the appearance of translucent materials often looks smooth and soft, which is a distinct feature of translucent materials. Additionally, in some cases, light can pass through a translucent object and light the object from behind, which is known as global effect. These visual phenomena require accurate simulation of subsurface scattering and make rendering translucent objects a very complicate and important problem in computer graphics.

Lambertian diffuse reflection or Bidirectional Reflection Distribution Function (BRDF) considers the case that light striking a surface point gets reflected at the same point. Nicodemus et al. [NRH*22] proposed Bidirectional Scattering Surface Reflection Distribution Function (BSSRDF) that considers the light scattering relation between the incident flux at a point and the outgoing radiance at another point on the surface. They formulated the subsurface scattering equation without providing an analytic solution. Hanrahan and Krueger [HK93] first proposed a method for simulating subsurface scattering by tracing light rays inside an object. Their approach, however, can only be applied to thin objects, such as leaf and paper; Light scattering in thick objects is too complicated to be fully traced.

Recently, Jensen et al. [JMLH01] proposed an analytic and complete BSSRDF model, in which BSSRDF is divided into a single scattering term and a multiple scattering term. Single scattering term is contributed by light that scatters only once inside an object; Light rays that scatter more than once contribute to the multiple scattering term. Jensen et al. approximated the multiple scattering term as a dipole diffusion process. The diffusion approximation works well in practice for highly scattering media where ray tracing is computationally expensive [HK93, Sta95]. This approach is much faster than Monte Carlo ray tracing, because the scattering effect is evaluated without simulating the light scattering inside the model.

Several recent papers exploit the diffusion approximation to render translucent materials [CHH03, DS03, HBV03, HV04, JB02, LGB*02, KLC06,

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

MKB*03a, MKB*03b, WTL05]. Jensen and Buhler [JB02] proposed a two-stage hierarchical rendering technique to efficiently render translucent materials in interactive frame rate with acceptable error bound. Keng et al. [KLC06] further improved this approach using a cache scheme. Lensch et al. [LGB*02] modified Jensen's formulation as a vertex-to-vertex throughput formulation, which is precomputed and used for rendering at run-time. Carr et al. [CHH03] further improved this approach by using parallel computation and graphics hardware. In [HBV03, HV04], the contribution of light from every possible incoming direction is precomputed and compressed using spherical harmonics. With the precomputation, images can be rendered in real-time frame rate. Wang et al. [WTL05] presented a technique for interactive rendering of translucent objects under all-frequency environment maps. The equations of single and multiple scattering terms are reformulated based on the precomputed light transport.

Although precomputed schemes can achieve real-time performance, they require extensive precomputation and extra storage for the results. To avoid the extensive precomputation, Mertens et al. [MKB*03a] and Dachsbacher and Stamminger [DS03] proposed image-based schemes for computing translucent effects in real-time without precomputation. The method by Mertens et al. [MKB*03a] renders the irradiance image from camera view and evaluates the local subsurface scattering by sampling the irradiance in a small area around the outgoing point. Dachsbacher and Stamminger [DS03] extended the concept of shadow map by first rendering the irradiance image from light view and then evaluating translucent effects by applying filtering around the projected outgoing point on irradiance image. This approach can catch both local and global subsurface scattering, but the effects of global subsurface scattering are inaccurate because only an area of single incident point, rather than all incident points, is taken into account.

The most costly operation in rendering translucent objects is sampling over a model surface, which is hard to be implemented on graphics hardware. The texture map has been used to store surface properties and its operations are well supported by graphics hardware. Through a proper mesh parameterization, a bijective mapping between object surface and texture space can be assured, and hence the sampling over the texture space can be considered to be equivalent to the sampling over the model surface.

In this paper, we propose a real-time translucency rendering approach for homogeneous materials. Our framework performs importance sampling over the texture space using graphics hardware. We first convert the integration over a 3D model surface into an integration over the 2D texture space, and then do the importance sampling based on the irradiance to compute translucent rendering. Our experiments have depicted that plausible images can be rendered in real time for many complex translucent models with dynamic light and material properties.

For objects with more apparent local effect, higher number of samples is generally required to generate accurate images for translucent rendering. For such objects, the proposed texture space importance sampling may require more samples than the cases in which global effect is more apparent. To deal with such cases, we propose a hybrid approach that decomposes the integration into two parts, one for local effect and the other for global effect, and evaluate the local and global effects using the translucent shadow map approach [DS03, MKB*03a] and the proposed texture space importance sampling, respectively. Such a hybrid scheme is able to steadily render the translucent effect in real time with a fixed amount of samples.

2. Translucent Rendering Using Texture Space Importance Sampling

2.1. BSSRDF and Dipole Approximation

The scattering of light in translucent materials is described by the Bidirectional Surface Scattering Reflection Distribution Function(BSSRDF) [NRH*22]:

$$S(x_i, \vec{\omega_i}, x_o, \vec{\omega_o}) = \frac{dL(x_o, \vec{\omega_o})}{d\Phi(x_i, \vec{\omega_i})},$$

where L is the outgoing radiance, Φ is the incident flux, and $\vec{\omega_i}$ and $\vec{\omega_o}$ are incoming and outgoing direction of the light at the incoming and outgoing position x_i and x_o, respectively. In order to compute the shading of translucent materials, the following integral needs to be computed :

$$L(x_o, \vec{\omega_o}) = \int_A \int_{\Omega_+} L_i(x_i, \vec{\omega_i}) S(x_i, \vec{\omega_i}, x_o, \vec{\omega_o}) |\vec{N_i} \cdot \vec{\omega_i}| d\vec{\omega_i} dx_i \quad (1)$$

where A is the surface of the object, Ω_+ is the hemisphere positioned at the incoming position x_i, $\vec{N_i}$ is the normal direction at x_i, and $L(x_o, \vec{\omega_o})$ is the outgoing radiance. By solving this equation, subsurface scattering can be simulated.

$S(x_i, \vec{\omega_i}, x_o, \vec{\omega_o})$ is an eight-dimensional function relating incident flux and outgoing radiance. It is not easy to efficiently precompute and store BSSRDF in memory because of the high dimensionality, especially when surface geometry is complicated. Jensen and Buhler [JB02] have shown that when the single scattering term is ignored, BSSRDF with multiple scattering term can be presented as a four-dimensional function $R_d(x_i, x_o)$. Taking into account the Fresnel transmittance $F_t(\eta, \vec{\omega})$ results in the following function

$$S(x_i, \vec{\omega_i}, x_o, \vec{\omega_o}) = \frac{1}{\pi} F_t(\eta, \vec{\omega_o}) R_d(x_i, x_o) F_t(\eta, \vec{\omega_i}).$$

Substituting this into Equation 1 yields

$$L(x_o, \vec{\omega_o}) = \frac{1}{\pi} F_t(\eta, \vec{\omega_o}) B(x_o), \quad (2)$$

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

where

$$B(x_o) = \int_A E(x_i) R_d(x_i, x_o) dx_i \tag{3}$$

is the outgoing radiosity, and

$$E(x_i) = \int_{\Omega_+} L_i(x_i, \vec{\omega_i}) F_t(\eta, \vec{\omega_i}) |\vec{N_i} \cdot \vec{\omega_i}| d\vec{\omega_i} \tag{4}$$

is the irradiance. Note that $R_d(x_i, x_o)$ is the diffusion function approximated by the dipole diffusion equation shown in Table 1 [JMLH01].

$R_d(x_i,x_0) = \frac{\alpha'}{4\pi}[z_r(1+\sigma_{tr}s_r)\frac{e^{-\sigma_{tr}s_r}}{s_r^3} + z_v(1+\sigma_{tr}s_v)\frac{e^{-\sigma_{tr}s_v}}{S_v^3}]$		
r	=	$\|x_o - x_i\|$
z_r	=	$1/\sigma_t'$
z_v	=	$z_r(1+4A/3)$
s_r	=	$\sqrt{z_r^2 + r^2}$
s_v	=	$\sqrt{z_v^2 + r^2}$
A	=	$\frac{1+F_{dr}}{1-F_{dr}}$
Fdr	=	$-\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta$
σ_{tr}	=	$\sqrt{3\sigma_a\sigma_t'}$
σ_t'	=	$\sigma_a + \sigma_s'$
α'	=	σ_s'/σ_t'
σ_s'	:	reduced scattering coefficient
σ_a	:	absorption coefficient
η	:	relative refraction index

Table 1: *The dipole diffusion equation for BSSRDF model.*

These equations describe the properties of local and global subsurface scattering. The outgoing radiosity is contributed by the integration of irradiance and the diffusion function $R_d(x_i, x_o)$. The value of diffusion function falls off rapidly as the distance $\|x_i - x_o\|$ increases. In consequence, the integration for the outgoing radiosity is mostly contributed by the points in the nearby area of the outgoing point that is visible to the light source. This is termed as the local subsurface scattering. When the outgoing point is invisible to the light source, the irradiance in the nearby area is low and the outgoing radiosity is instead gathered from a farther area that is visible to the light source. This feature is termed as the global surface scattering.

2.2. Texture Space Importance Sampling

Computing the integration for outgoing radiance by sampling over the object surface is usually computationally expensive. Therefore, we first construct a bijective mapping from object surface to the texture space using mesh parameterization and then do the sampling on the texture space. Performing importance sampling over the texture space to compute the integration is more efficient and can be greatly accelerated by graphics hardware.

In the following, we explain how to reformulate Equation 3 in order to compute the integration over the texture space.

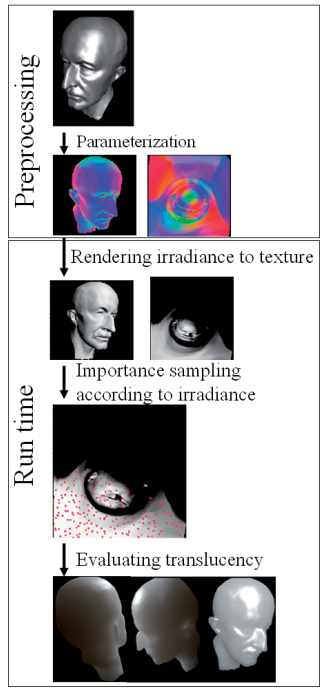


Figure 1: *The framework of proposed approach.*

Since the input model is a triangle mesh, the integration can be represented as summation over triangle faces as follows :

$$B(x_o) = \sum_{j=1}^{N_F} \int_{F_j} E(x_i) R_d(x_i, x_o) dx_i, \tag{5}$$

where N_F is the total number of triangle faces and F_j is the the jth triangle face.

After parameterization, each triangle face will be associated with a unique texture coordinate. In theory, we can directly map a triangle face to texture space. However, triangles may be shrunk or enlarged during the parameterization process. In order to obtain correct integration result, we scale the area of each triangle face in the integration:

$$B(x_o) = \sum_{j=1}^{N_F} \frac{A_S^j}{A_T^j} \int_{T_j} E(x_i) R_d(x_i, x_o) dx_i, \tag{6}$$

where A_S^j and A_T^j are area of the triangle face j on 3D model surface and 2D texture space, respectively, and T_j is the jth triangle face on the texture space. Note that theoretically Equation 6 is only valid when the parameterization is with constant jacobian; however, from our experiments, we found that translucent rendering results do not vary much whether this assumption holds or not.

Since the object surface is usually segmented into patches to reduce the parameterization distortion, the texture map is formed by packing a set of atlases, each of which corresponds to a patch. Among atlases, there is empty space. In

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

the empty space, the area of triangle A_S^j and irradiance $E(x_i)$ are considered as both zero. Therefore, the integration can be evaluated over the whole texture space as follows:

$$\begin{aligned} B(x_o) &= \sum_{j=1}^{N_F} \frac{A_S^j}{A_T^j} \int_{T_j} E(x_i) R_d(x_i, x_o) dx_i \\ &= \int_T E'(x_i) R_d(x_i, x_o) dx_i, \end{aligned} \tag{7}$$

where T is the texture space of the model surface, and $E'(x_i) = A_S^j E(x_i)/A_T^j$ is the product of the irradiance term and the area scaling term.

Next, we describe how to evaluate Equation 7 using importance sampling [DBB02, KW86]. Importance sampling approximates an integral by the weighted mean of the integrand values that are sampled over the integration domain D:

$$\int_D f(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \tag{8}$$

where the samples x_i are generated according to a probability density function $p(x)$ whose value is high when $f(x)$ is large; there are more samples in important regions.

We need to define an appropriate probability density function to render translucency. In our case, the integrand f is the product of irradiance function $E'(x_i)$ and diffusion function $R_d(x_i, x_o)$. If we perform importance sampling based on $E'(x_i)R_d(x_i, x_o)$, precomputation at every outgoing point is required since the distance and neighboring relation between triangles on the 3D model surface is lost in the 2D texture space after parameterization. This is against our goal of rendering translucency with low precomputation overhead. Therefore, we define $p(x_i)$ over texture space to be proportional to the value of the irradiance function $E'(x_i)$:

$$p(x_i) = E'(x_i)/E'_{avg}, \tag{9}$$

where $E'_{avg} = \int_T E'(x)dx$ is the integration of irradiance over texture space. With N samples distributed based on $p(x_i)$, the integration of outgoing radiosity can be rewritten as

$$\begin{aligned} B(x_o) &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{E'(x_i) R_d(x_i, x_o)}{p(x_i)} \\ &= \frac{E'_{avg}}{N} \sum_{i=1}^{N} R_d(x_i, x_o). \end{aligned} \tag{10}$$

Note that samples generated according to irradiance can now be shared for different outgoing points since the irradiance function in Equation 7 only depends on incident positions.

3. Implementation on GPU

In this section, we describe implementation details of our approach. As shown in Figure 1, the proposed approach consists of a precomputation process and a three-stage GPU-based translucent rendering process. In precomputation, parameterization is applied to an input mesh and a normal map is derived. To avoid the undersampling problem, area preserving or signal-specialized parameterization is preferred. In our implementation, an area-preserving parameterization method proposed by Yoshizawa et al. [YBS04] is used.

The three-stage run-time process includes rendering the irradiance to a texture map, performing importance sampling according to the irradiance, and evaluating the the outgoing radiance based on the generated samples. Tone reproduction is applied before the final translucent images are output. We implemented the run-time process on GPU using OpenGL, OpenGL extension, shading language, and shader model 2.0.

Rendering Irradiance to Texture

At the first stage, we render the irradiance $E'(x_i) = A_S^j E(x_i)/A_T^j$ into a texture map. In our implementation, point light sources are used. We evaluate the irradiance on the model surface in pixel shader and render it into the associated texture coordinate. In addition, 3D position x_i is rendered to another texture map.

Importance Sampling

In the second stage, N samples are distributed over the texture space according to Equation 9. We apply the inverse cumulative distribution function (ICDF) to determine the location of each sample. CDF is a function that accumulates the values of the probability density function in sequence, and is defined as $F(x_i) = \int_{z_0}^{x_i} p(z)dz$, where z_0 is the beginning of the sequence. Samples are generated according to $p(x)$ by solving $F(x_i) = y$ for the values of y uniformly sampled over the interval [0,1). To exploit GPU acceleration to speed up the process, we store each sample as a pixel of screen with assigned probability y, use pixel shader to compute ICDF, and finally output the resultant sample positions into a texture map.

Conventionally, row-major order is used for computing CDF on GPU. Figure 2(a) shows a row-major order on a 4x4 texture map, on which the inverse function $F^{-1}(u)$ can usually be derived by a binary search over the data structure. The row-major order, however, does not fully utilize the parallel computational power of GPU. We propose a mipmap-based order as shown in Figure 2(b), in which the cumulation process can be performed using the mipmapping hardware of GPU. For each sample, the inverse function $F^{-1}(u)$ is derived by traversing the mipmap structure from the top level to the lowest level. Since the evaluation of inverse function can be performed independently, samples can be generated in pixel shader.

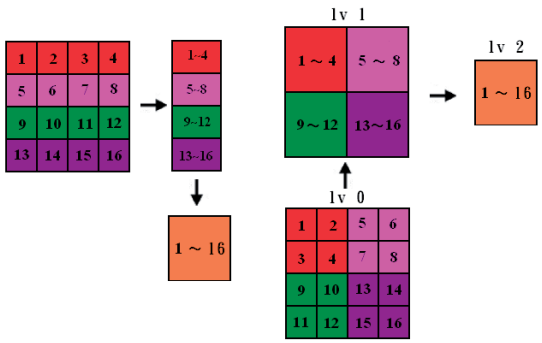
Evaluating Translucency

After generating N samples, we evaluate the outgoing radiance for each point on the model surface. For each rendered pixel, we fetch all samples by a 2D texture lookup and sum up the contribution from these samples. Combining Equations 2 and 10 yields the following equation for computing outgoing radiance

$$L(x_o, \vec{\omega_o}) = \frac{F_t(\eta, \vec{\omega_o}) E'_{avg}}{N\pi} \sum_{i=1}^{N} R_d(x_i, x_o). \tag{11}$$

The diffusion function $R_d(x_i, x_o)$ and the Fresnel transmittance $F_t(\eta, \vec{\omega_o})$ contain computation of trigonometry,

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.



(a) Row-major Order (b) Mipmap-based Order

Figure 2: *Two different data orders for evaluating cumulative distribution function: Row-major order and mipmap-based order.*

square root, and exponential function, which are complicated instructions. To facilitate GPU programming, we store $R_d(x_i,x_o)$ and $F_t(\eta,\vec{\omega_o})$ into texture maps and evaluate the functions by texture lookup. According to the equation shown in Table 1, $R_d(x_i,x_o)$ can be represented as a one-dimensional function,

$$R_d(x_i,x_o) = R_d(r),$$

where $r = \|x_o - x_i\|$ is the distance between the incident and the outgoing point. We normalize the diagonal length of the input object to 1 and store the diffusion function $R_d(r)$ as a one-dimensional texture over interval $[0,1]$. The Fresnel transmittance $F_t(\eta,\vec{\omega_o})$ is also stored as a one-dimensional texture using the dot product of normal and view direction as the indexing parameter.

After computing outgoing radiance, we transform the radiance to color using DirectX's HDR Lighting, which is a tone mapping algorithm commonly used in real-time high dynamic range lighting. We approximate the average radiance $Radiance_{avg}$ of rendered pixels by the average irradiance E'_{avg} computed at the second stage. Therefore, we can integrate the tone mapping into the final stage without additional passes. The equations for computing the tone mapping are as follows:

$$\begin{aligned} Radiance_{scale}(x,y) &= \frac{Radiance(x,y)}{Radiance_{avg}} \\ Color(x,y) &= \frac{1+Radiance_{scale}(x,y)}{Radiance_{scale}(x,y)}. \end{aligned}$$

After the tone mapping, we add the Phong-based specular effect to the final image.

4. Hybrid Method

The diffusion function is the most costly to sample for translucent rendering since it depends on the lighting from a large fraction of the model surface. Jensen and Buhler [JB02] evaluated the integration by a uniform sampling over the model surface. In order to have sufficient number of samples to generate quality results, the mean-free path ℓ_u is used as the maximal distance between neighboring samples and $\ell_u = 1/\sigma'_t$ is the average distance at which the light is scattered. As a result, for a given object, the required number of samples is approximately $A/(\pi\ell_u^2)$, where A is the surface area of the object. The required number of samples will be higher for objects with larger value of σ'_t or larger size. In this case, the local translucency is the dominant rendering effect.

The aforementioned problem also happens in our texture space importance sampling method. In order to deal with this problem, we proposed a hybrid approach that first decomposes the integration equation into two parts, one for local effect and the other for global effect. We then evaluate the local and global effects using the translucent shadow map [DS03] and the proposed texture space importance sampling, respectively.

4.1. Decomposition of Diffusion Function $R_d(r)$

We decompose the diffusion function $R_d(r)$ into a local term and a global term:

$$R_d(r) = R_d(r)W_l(r) + R_d(r)W_g(r),$$

where $W_l(r)$ and $W_g(r)$ are weighting functions defined by

$$\begin{aligned} W_l(r) &= \begin{cases} 1.0 - 0.5e^{-(r-R_p)\cdot K} & r \le R_p \\ 0.5e^{-(r-R_p)\cdot K} & r > R_p \end{cases}, \\ W_g(r) &= 1.0 - W_l(r), \end{aligned}$$

where R_p is the distance at which the contributions of the global and local term are equal and $K = 1.5$ is a constant. The outgoing radiosity $B(x_o)$ in Equation 3 can then be reformulated as follows:

$$\begin{aligned} B(x_o) &= \int_A E(x_i)R_d(x_i,x_o)dx_i \\ &= \int_A E(x_i)R_d(r)(W_l(r)+W_g(r))dx_i \\ &= \int_A E(x_i)R_d(r)W_l(r)dx_i + \int_A E(x_i)R_d(r)W_g(r)dx_i \\ &= B_l(x_o) + B_g(x_o) \end{aligned}.$$

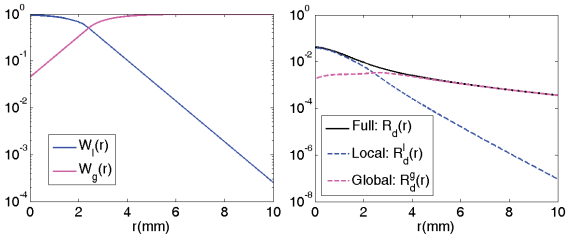
Figure 3 shows an example of weighting function, and the decomposed diffusion functions, $R_d^l(r) = R_d(r)W_l(r)$ and $R_d^g(r) = R_d(r)W_g(r)$. Notice that, as shown in Figure 3(b), the function value of small distance r is retained in the local term while the global term preserves the function value for large distance r.

Choosing an appropriate R_p value is important. The global term will still contain the distinct local effect if R_p is too small, and leave nothing if R_p is too large. In the following, we describe a method to choose R_p. We first define an importance function as the product of the diffusion function and the circumference of a circle of radius r:

$$R_d^{Imp}(r) = R_d(r) * 2\pi r,$$

and find the intersections of $y = R_d^{Imp}(r)$ and $y = c$ for RGB channels, where c is the value of a user-specified importance

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.



(a) $W_l(r)$ and $W_g(r)$ (b) $R_d(r)$, $R^l_d(r)$, and $R^g_d(r)$

Figure 3: *Graphs of weighting function and diffusion function for red channel of Skimmilk material [JMLH01] (σ_a is 0.0024, σ'_s is 0.70, and η is 1.3). R_p is 2.4141.*

lower bound. The largest intersection point is then chosen as R_p. Figure 4 illustrates choosing R_p for skimmilk material.

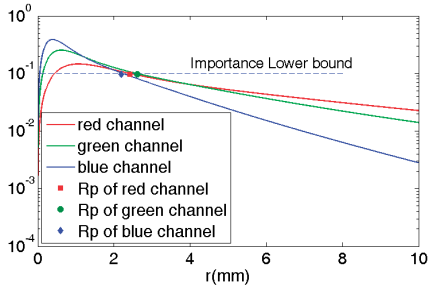


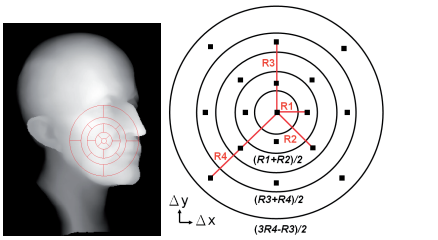
Figure 4: *Determining R_p for skimmilk material.*

4.2. Computing Local Translucency

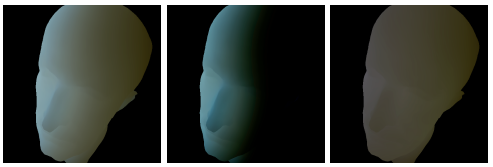
We modified the translucent shadow map approach proposed in [DS03] to compute the local translucency. Translucent shadow map approach renders an irradiance map on the projection plane viewed from the light source and computes translucency by applying filtering on the irradiance map. We replaced the fixed filtering pattern with 21 samples proposed in [DS03] by a circular filtering pattern with $L \cdot C + 1$ samples, where L is the number of concentric circles and C is the number of samples uniformly distributed in each circular ring. The radii of concentric circles are determined by applying the importance sampling on the projection plane as proposed by Mertens et al. [MKB*03a]. The circular filtering pattern makes sampling isotropic and can easily control the sampling density according to the radius. Figure 5(a)(b) shows an circular filtering pattern. An example of decomposition of global and local translucency is shown in Figure 5(c)(d)(e).

5. Experimental Results

In this section, we show the results of the texture space importance sampling method and the hybrid method. We compare the proposed texture space importance sampling approach with an interactive rendering approach [MKB*03a] and a non-real-time approach [JB02]. All experiments are performed on an Intel Core 2 Duo E6600 2.4GHz PC with NVIDIA Geforece 8800 GTX graphics hardware. All images are rendered in 512×512 screen resolution except that those in Figure 11 are rendered in 300×300 screen resolution. The scattering parameters for different materials used in our experiments are adopted from Figure 5 of [JMLH01].



(a) (b)



(c) Full (d) Local (e) Global

Figure 5: *(a) Irradiance map rendered from the light view. (b) Circular filtering pattern with 17 $(= 4 \cdot 4 + 1)$ samples. Images with (c) full R_d function, (d) local R^l_d function, and (e) global R^g_d function. Model is 10mm MaxPlanck and material is skimmilk.*

5.1. Texture Space Importance Sampling

Table 2 and 3 show the performance of the texture space importance sampling method using an irradiance map with 1024×1024 resolution. Table 2 measures the frame rate in the run-time process. If lighting and material are fixed, then only the third stage of the run-time process needs to be recomputed when view changes. In this situation, higher frame rate can be achieved as shown in Table 3. Note that material properties only affect the content of $R_d(r)$, which is stored as a 1D texture. The performance of our algorithm mainly depends on the number of samples and rendered pixels. One can observe from Table 2 and 3 that the computational time is roughly proportional to the number of samples. The resolution of geometry also has influence to the performance; models with fewer triangles are rendered faster, e.g., the Cowhead example in the table. Result images of three cases in Table 3 are shown in Figure 6. Figure 7 shows that the RMSE decreases as number of samples increases in rendering a 30mm Santa model with skin2 material.

We compare the proposed texture space importance sampling approach to an interactive rendering method proposed by Mertens et al. [MKB*03a]. Figure 8 compares images

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

Model	# of Triangles	Number of Samples 49	100	169	256	400	900	1600	2500
Sphere	28560	119.2	99.3	78.5	62.9	47.2	25.3	15.0	9.8
CowHead	2072	156.6	142.7	126.9	110.4	91.5	57.1	37.0	25.2
TexHead	8924	148.6	128.0	110.4	91.8	72.6	42.4	26.5	17.6
MaxPlanck	9951	135.2	120.2	102.5	83.1	64.7	37.3	22.9	15.1
Parasaur	7685	148.4	127.9	110.0	93.7	74.7	43.2	27.0	18.2
Igea	10000	145.7	122.6	103.3	85.5	67.0	38.2	23.4	15.4
Santa	10000	152.0	118.7	106.3	90.7	69.0	39.7	24.8	16.5

Table 2: *Performance of the entire run-time process for different models in frame per second (fps).*

Model	# of Triangles	Number of Samples 49	100	169	256	400	900	1600	2500
Sphere	28560	449.3	242.4	150.2	101.2	66.0	29.9	16.6	10.4
CowHead	2072	1144.9	634.1	408.6	278.1	181.7	82.9	46.4	29.1
TexHead	8924	819.1	449.8	277.7	187.8	122.3	55.6	31.1	19.5
MaxPlanck	9951	698.3	376.9	233.6	157.8	102.8	47.2	26.6	16.7
Parasaur	7685	874.2	485.0	291.8	199.1	129.8	57.3	31.9	20.3
Igea	10000	702.9	377.8	238.0	163.2	107.1	48.8	26.9	16.8
Santa	10000	734.1	402.5	248.9	170.1	110.6	50.6	28.3	16.7

Table 3: *Performance of the integration stage (the third stage of run-time process) for different models in fps.*

Figure 6: *Result images with settings in Table 3. (a) MaxPlanck with skimmilk and 900 samples in 47.2 fps (b) Parasaur with skin1 and 1600 samples in 31.9 fps (c) TexHead with marble and 2500 samples in 19.5 fps.*

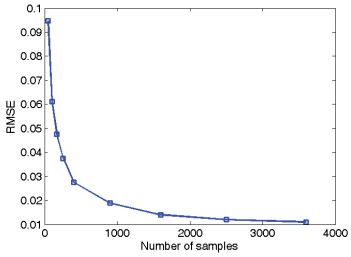


Figure 7: *RMSE of 30mm Santa model rendered with skin 2 material using different number of samples.*

of 10mm Igea with marble material in different views. Because Mertens et al. rendered the irradiance map from camera view, irradiance is caught poorly when the camera turns to the other side of the light source (see the second column in Figure 8). Besides, Mertens et al. applied importance sampling over the projection plane based on the diffusion function $R_d(r)$, samples need to be regenerated for each rendered pixel and the contribution is scaled by the normal direction and depth of each sample. Therefore, their method needs more instructions in the final stage and the computation cost is about 10 times of our method.

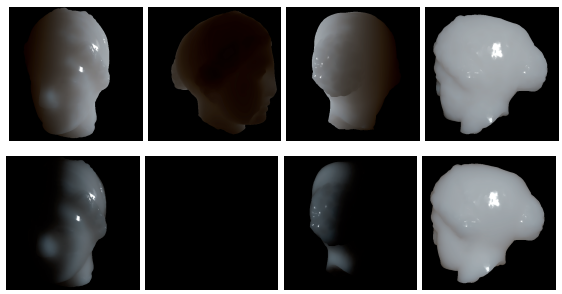
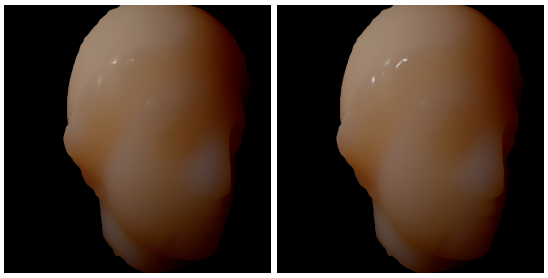


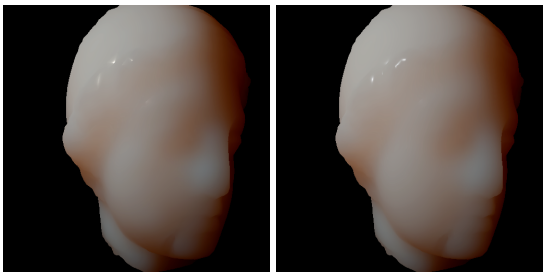
Figure 8: *Comparison with an interactive-rate rendering approach. Images in the top row are rendered by our method using 1600 samples in 26.9 fps. Images in the bottom row are rendered by the method of Mertens et al. [MKB*03a] under the same condition in 2.6 fps. Our method renders back-lit translucency correctly and its performace is 10 times faster.*

To quantitatively evaluate the image quality of our results, we also compare the proposed texture space importance sampling approach to a non-real-time method proposed by Jensen and Buhler [JB02]. Figure 9 and 10 show the images rendered by these two methods with different model scales and materials. In Figure 9(a)(c), 1600 samples are used and images are rendered in 19.9 fps. The root mean

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

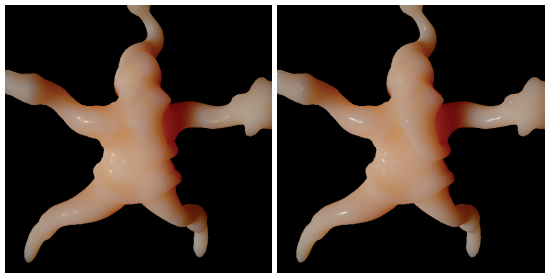


(a) Ours (skin1, 1600 samples) (b) Jensen & Buhler (skin1)

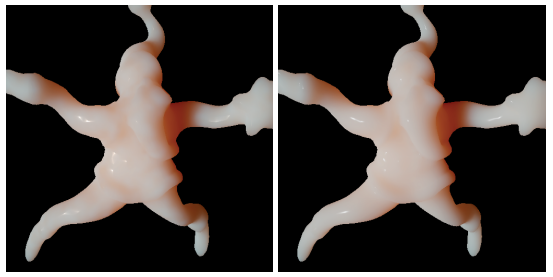


(c) Ours (skin2, 1600 samples) (d) Jensen & Buhler (skin2)

Figure 9: *Images of 10mm Igea rendered by our method and by Jensen and Buhler's method [JB02].*



(a) Ours (skin1, 2500 samples) (b) Jensen & Buhler (skin1)



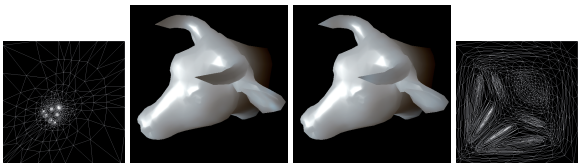
(c) Ours (skin2, 2500 samples) (d) Jensen & Buhler (skin2)

Figure 10: *Images of 30mm Santa rendered by our method and by Jensen and Buhler's method [JB02].*

square error (RMSE) between images of these two methods are 0.0096 and 0.0098 for skin1 and skin2, respectively. In Figure 10(a)(c), 2500 samples are used and images are rendered in 16.5 fps. The RMSE between images of these two methods are 0.0138 and 0.0119 for skin1 and skin2, respectively. Figure 9(b)(d) and Figure 10(b)(d) are rendered by the non-real-time method [JB02] with a rapid hierarchical octree data structure under 0.3 error threshold in 0.016 fps.

Our approach can render dynamic translucent materials by extending the 1D texture of the diffusion function $R_d(r)$ to a 2D texture. We linearly interpolate the scattering coefficients and the model size from one setting to another. Figure 11(a) shows the result images of objects with dynamically varying model size. The number of samples and rendering speed in Figure 11(a) range from 400 to 48400 samples and 140 to 1.1 fps respectively (from left to right images). In Figure 11(b), both model size and material are dynamically varied. 3600 samples are used in all images and rendering speed ranges from 150 to 18 fps.

To study the effects of paramerization on our translucency rendering approach, we tested our approach with different parameterization methods. Our experiment shows that the RMSE of the rendering results of a 10mm IGEA model with skin1 material is 0.0124 for Floater's non-area-preserving parameterization method [Flo97] and 0.0089 for Yoshizawa's area-preserving parameterization method [YBS04]. This suggests that for models without sharp geometric features, area-preserving parameterization generates slightly better results than non-preserving parameterization



(a) (b) (c) (d)

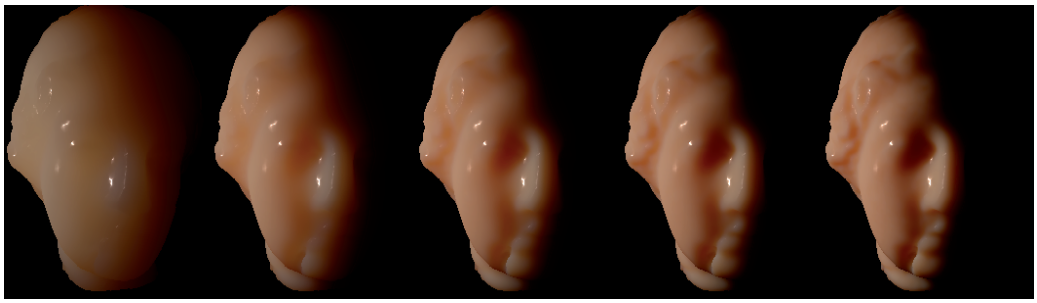
Figure 12: *(b,c) Images of cow head with marble material. (a) Texture derived by Floater's parameterization [Flo97]. (d) Texture derived by an area-preserving parameterization proposed by Yushizawa et al. [YBS04]. There are significant errors around the horn in (b) due to the undersampling problem.*

does. For models with sharp corners, non-area-preserving parameterization may introduces the under-sampling problem. Figure 12(a) shows an example in which the areas of triangles around the horn are shrunk too much due to non-area-preserving parameterization. The under-sampling problem induces significant errors around the horn when compared with the result generated using Yoshizawa's area-presering parameterization method as shown in Figure 12(c).

5.2. Hybrid Method

Our hybrid method decomposes the diffusion function into a local term and a global term. This decomposition ameliorates the problem that the required number of samples increases rapidly when local translucency effects dominate the

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

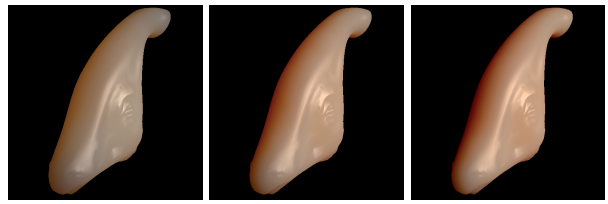


(a) 10 mm skin1 to 70 mm skin1

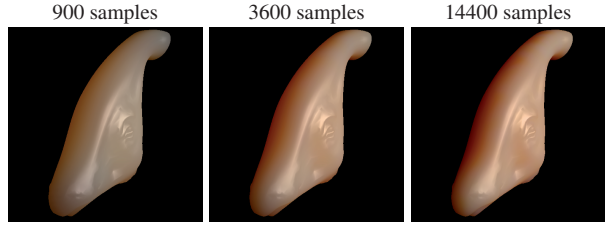
(b) 10 mm skimmilk to 40 mm skin1

Figure 11: *Rendering results of objects with (a) dynamically varying model size and (b) dynamically varying model size and material. The intermediate results are generated by linearly interpolating the model size and scattering coefficients.*

rendering results, either due to large model size or material properties. Figure 13 shows the rendered images of 10mm, 20mm and 30mm Parasaur with skin1 material. The number of samples needed for good image quality increases to 14400 in the texture space importance sampling method, while the hybrid method can achieve similar image quality with much fewer number of samples (compare (c) and (f) in Figure 13 and 14). In the hybrid method, we use 900 samples and 401 samples for the global and local effect, respectively. Since large L and C value would increase the rendering time while decreasing RMSE, we tested different values and found that $L = C = 20$ is a good compromise for the rendering speed and accuracy. Figure 14 is another example for three different materials: skin1, marble, and cream. Using images generated by Jensen and Buhler's method [JB02] as the ground truth, the RMSE in Figure 14(a) and (d) are 0.0107 and 0.0144, respectively. These results show that the rendering speed of our hybrid method can be maintained for different model sizes and object materials.



(a) 10mm 55.0fps 900 samples | (b) 20mm 13.4fps 3600 samples | (c) 30mm 3.3fps 14400 samples



(d) 10mm 17.3fps 900+401 samples | (e) 20mm 17.3fps 900+401 samples | (f) 30mm 17.3fps 900+401 samples

Figure 13: *Images of Parasaur with skin1 material and varying model sizes. (a)(b)(c) are rendered by our texture space importance sampling method. (d)(e)(f) are rendered by our hybrid method.*

6. Conclusion and Future Work

In this paper, we have presented a GPU-based texture space importance sampling method for rendering translucent materials. The proposed method converts the integration over the model surface into an integration over a texture space. This conversion makes GPU implementation of importance sampling feasible such that translucency rendering with good accuracy can be archived in real-time. Our comparison experiments show that the proposed approach has faster speed and better back-lit translucency effects than the interactive-rate rendering approach proposed by Mertens et al. [MKB*03a]. Moreover, the proposed texture space importance sampling approach achieves real-time frame rate (16 to 20 fps) with less than 0.014 RMSE when compared to Jensen and Buhler's non-real-time approach [JB02].

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

(a) skin1 18.2fps 2500 samples | (b) marble 3.0fps 14400 samples | (c) cream 0.6fps 62500 samples

(d) skin1 21.8fps 900+401 samples | (e) marble 21.6fps 900+401 samples | (f) cream 22.2fps 900+401 samples

Figure 14: *Images of Santa with 30mm size and different materials. (a)(b)(c) are rendered by our texture space importance sampling method. (d)(e)(f) are rendered by our hybrid method.*

The proposed texture space importance sampling method may require more samples for objects with apparent local translucency effects. This problem also happens on approaches that apply uniform sampling on model surface [JB02]. We overcome this problem by proposing a hybrid method in which translucent rendering is decomposed into local part and global part. Our experiments show that the proposed hybrid method can effectively reduce number of samples and render translucency efficiently.

In the future, we plan to exploit the temporal coherence to reduce the number of samples in rendering consecutive frames when the light source moves or the model size changes. Besides, area-preserving parameterization does not take the variation of irradiance value into account and hence may still introduce under-sampling problems. We will investigate how to incorporate the signal-specialized parameterization into our texture space importance sampling scheme. Furthermore, mesh dissection is often used to reduce the parameterization distortion. Issues regarding the importance sampling over texture atlas will also be investigated. Finally, we would like to explore the possibility to do importance sampling according to the product of irradiance and diffusion function. This is a more challenging problem for real-time translucent rendering, but might lead to a unified approach that is capable of handling both local and global translucent effect.

References

[CHH03] CARR N. A., HALL J. D., HART J. C.: Gpu algorithms for radiosity and subsurface scattering. In *Proceedings of Graphics hardware '03* (2003), pp. 51–59.

[DBB02] DUTRE P., BALA K., BEKAERT P.: *Advanced Global Illumination*. A. K. Peters, Ltd., 2002.

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Eurographics workshop on Rendering '03* (2003), pp. 197–201.

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 4 (1997), 231–250.

[HBV03] HAO X., BABY T., VARSHNEY A.: Interactive subsurface scattering for translucent meshes. In *Symposium on Interactive 3D Graphics '03* (2003), pp. 75–82.

[HK93] HANRAHAN P., KRUEGER W.: Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH '93* (1993), pp. 165–174.

[HV04] HAO X., VARSHNEY A.: Real-time rendering of translucent meshes. *ACM Trans. Graph. 23*, 2 (2004), 120–142.

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. In *Proceedings of SIGGRAPH '02* (2002), pp. 576–581.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of SIGGRAPH '01* (2001), pp. 511–518.

[KLC06] KENG S.-L., LEE W.-Y., CHUANG J.-H.: An efficient caching-based rendering of translucent materials. *The Visual Computer 23*, 1 (2006), 59–69.

[KW86] KALOS M. H., WHITLOCK P. A.: *Monte Carlo methods. Vol. 1: Basics*. Wiley-Interscience, 1986.

[LGB*02] LENSCH H. P. A., GOESELE M., BEKAERT P., KAUTZ J., MAGNOR M. A., LANG J., SEIDEL H.-P.: Interactive rendering of translucent objects. In *Proceedings of Pacific Graphics '02* (2002), pp. 214–224.

[MKB*03a] MERTENS T., KAUTZ J., BEKAERT P., REETH F. V., SEIDEL H.-P.: Efficient rendering of local subsurface scattering. In *Proceedings of Pacific Graphics '03* (2003), pp. 51–58.

[MKB*03b] MERTENS T., KAUTZ J., BEKAERT P., SEIDELZ H.-P., REETH F. V.: Interactive rendering of translucent deformable objects. In *Eurographics workshop on Rendering '03* (2003), pp. 130–140.

[NRH*22] NICODEMUS F. E., RICHMOND J. C., HSIA J. J., GINSBERG I. W., LIMPERIS T.: Geometrical considerations and nomenclature for reflectance. 94–145.

[Sta95] STAM J.: Multiple scattering as a diffusion process. In *Eurographics Workshop on Rendering '95* (1995), Hanrahan P. M., Purgathofer W., (Eds.), pp. 41–50.

[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.

[YBS04] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of the Shape Modeling International 2004* (2004), pp. 200–208.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.