A Fast Translucency Appearance Model for Real-Time Applications

Francesco Banterle*
University of Bristol

Alan Chalmers†
University of Bristol

Algorithm comparison: (left) traditional rendering with light behind the Happy Buddha, the light does not pass through the object, and (right) our algorithm, the light can scatter inside the object, and so we achieve sub surface scattering effects.

Abstract

Realistic rendering of natural materials such as marble, jade, human skin, etc. requires the simulation of subsurface scattering of light. Such high-fidelity rendering, however, takes a significant amount of computation time, precluding its use in real-time application. This paper introduces a new technique for rendering the appearance of translucent material in real-time based on spherical harmonics. The technique consists of two passes, first the irradiance of the object is projected on a spherical harmonic basis, then in the second pass the exitant radiance is evaluated. This technique is not physically based, but it ensures high frame rates. It also supports deformable objects and with no precomputation time, so it could be used in interactive applications including videogames.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: SubSurface Scattering, GPU Technique, Real-Time Rendering, Spherical Harmonics Lighting.

1 Introduction

Translucent objects, including marble, jade, human skin, etc, are common in our daily life. Translucency is caused by light which enters in an object and is scattered according to the physical property of the object's material. This process, called subsurface scattering, makes the small surface detail smoother and lets light pass through the object. Traditionally, translucency has been approximated with Lambertian diffuse reflection model or with modified versions such as Wrap or Rim Lighting [Pharr 2002], but these approximations do not work with strongly translucent material. In the last few years many efficient algorithms to model subsurface scattering have been presented, such as [Dorsey et al. 1999], [Jensen et al. 2001], and [Jensen and Buhler 2002]. These are based on BSSRDFs (bidirectional surface scattering distribution function), which are a generalization of BRDFs with possibly distinct light incidence and exitance points. However the problem with these algorithms is that they are designed for off-line systems, so they are not suitable for interactive or real-time applications.

In this paper we present a new subsurface scattering model, faster to evaluate than the dipole's methods on a GPU and not limited in terms of light condition as the precomputed radiance transfer. The model is not physically based, as the objective is to simulate the appearance of translucent material in real time. The idea of such simulation of translucency is not new, for example [Fleming et al. 2004], where the study was focused on the perception of the translucency by the human eye using image contrast operators.

1.1 Previous Work

Recently researchers have introduced many algorithms to simulate subsurface scattering in real-time or interactively. Most of these algorithms are GPU adaptations of the dipole approximation [Jensen et al. 2001], a fast analytic solution to subsurface scattering, which eliminates the need for numerical simulation.
In [Dachsbacher and Stamminger 2003] a shadow map was extended to store depth and incident light information, and compute subsurface scattering effects by filtering the shadow map neighborhood using a hierarchical approach. The technique reaches 30 frames per second with models made of 10k triangles on an Intel Pentium4 2.4Ghz PC with an ATI Radeon9700, but it is constrained to directional light.
[Lensch et al. 2002] approximated the subsurface scattering by filtering the incident illumination stored in a texture atlas. The shape of these kernels is surface dependent and precomputed before lighting is applied. They also approximate forward scattering by precomputing a vertex-to-vertex throughput factor, which resembles a form factor. Forward scattering is rendered by performing a step

*e-mail: banterle@cs.bris.ac.uk
†e-mail: alan.chalmers@bris.ac.uk

similar to radiosity gathering, by collecting for a given vertex, the irradiance from the other vertices scaled by the throughput factor. This technique runs in real-time, but it does not allow deformable objects, as the texture atlas is precomputed. The authors achieved an interactive frame rate of 6 frames per second with a model made of 4K triangles using a dual Intel Xeon 1.7Ghz PC with a GeForce3.
In [Carr et al. 2003] a multiple-scattering subsurface light transport was examined to resemble a single radiosity gathering step, using a hierarchical radiosity algorithm on a GPU. They achieved a real-time frame rate of 30 frames per second employing a model made of 70K triangles on a PC with a NVIDIA GeforceFX5900, but the technique does not support deformable meshes, because it precomputes an hierachical mesh atlas.
A hierarchical boundary element method was used in [Mertens et al. 2003] to solve the integral describing subsurface scattering. The techinque runs at 5 frames per second on a dataset with 132K triangles, using a dual Intel Xeon 2.4GHz PC with an ATI Radeon 9700. It also allows users to change object geometry, lighting's settings, and material's properties.
An approximation of subsurface scattering using only local illumination was presented in [Hao and Varshney 2004], by blending illumination from neighbour vertices to approximate local back scattering. They precompute a scattering term for source lighting expressed in a piecewise linear basis. They reconstructed scattering per-vertex from directional light by linearly interpolating these terms computed from the nearest surrounding samples. This technique runs in real-time with a frame rate of 30 frames per second with a model made of 150K triangles on a Intel Pentium4 2.0Ghz PC with a NVIDIA GeForce3.
A different approach was presented in [Sloan et al. 2003], indeed for each vertex of the model the radiance transport is precomputed (including subsurface scattering) and projected into a spherical harmonics basis in order to relight the objects. The main problem of this method is that it allows only low-frequency lighting and static model. Recently in [Sloan et al. 2005] this algorithm was extended in order to allow deformable model but it is still limited to low-frequency lighting. The algorithm runs at 100-200 frames per second (with a precomputation time of less than 30 minutes) on an Intel Pentium4 3.0Ghz PC with a NVIDIA GeForce6800.

2 A simplified model of scattering

The key idea of the technique is a new simplified model of light scattering. It assumes that, when light enters in an object from an incoming point x_{in}, it is all focused in a point near the surface called accumulation point x_a. Then light is transmitted from x_a, in the object according to its index of refraction, and it leaves object from an outgoing point x_{out}. There is no phase function, because the light is scattered only forward (Figure 1). Note that in the physically based model proposed in [Jensen et al. 2001], light enters and scatters in the translucent material is allowed to move along more complex paths as shown in Figure 2. This model had been developed keeping in mind to reduce expensive computation, indeed in the classic evaulation presented in [Jensen et al. 2001] we must solve the equation of exitant radiance:

$$L^e(x_{out},\vec{\omega}_{out}) = \frac{1}{\pi}F_t(\eta,\vec{\omega}_{out})B(x_{out}) \tag{1}$$

$$B(x_{out}) = \int_S E(x_{in})R_d(x_{in},x_{out})dx_{in} \tag{2}$$

where $F_t(\eta,\omega_{out})$ is the fresnel term, $E(x_{in})$ is the irradiance at point x_{in}, $B(x_{out})$ is the radiosity term, $R_d(x_i,x_{out})$ is the dipole approximation of diffuse equation. To speed up the computation we may approximate this equation and achieve an appearance of translucency. The integral may divided as follows:

$$L^e(x_{out},\vec{\omega}_{out}) = \frac{1}{\pi}F_t(\eta,\vec{\omega}_{out})R_d(x_{in},x_{out})B(x_{out}) \tag{3}$$

$$B(x_{out}) = \int_S E(x_{in})dx_{in} \tag{4}$$

the second term $B(x_{out})$ is computed during the accumulation step using spherical harmonics. Then the first term $\frac{1}{\pi}F_t(\eta,\omega_{out})R_d(x_{in},x_{out})$ is computed during the evaluation step, where x^*_{in} is the point intersected by a ray which has got refraction as direction and the point at x_{out} as origin.

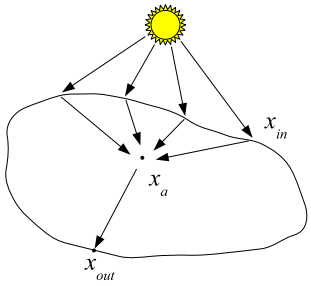


Figure 1: The simplified model of light subsurface scattering: the incoming light from the surface is accumulated in the accumulation point x_a, and transmitted along the refracted direction.

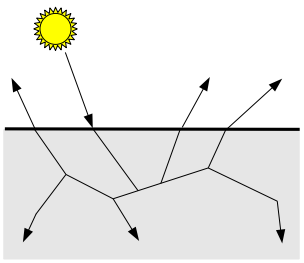


Figure 2: The general model of light subsurface scattering: light enters in the material and it is scattered in various direction according to material's properties.

2.1 Accumulation Step: Spherical Harmonics Projection

The first step of the technique is the calculation of irradiance sample points of the model (x_{in}), then these sample points will be scattered into the material. In [Dachsbacher and Stamminger 2003] sample points are calculated using a render to texture from the light point of view, but in our technique we use an external cubemap of the object, because the projection to a spherical harmonics basis is simplified. It is possible to use dual paraboloid render to texture, but in this case we take care with the distorsions.
An external cubemap is a cubemap calculated from six external cameras having the center of the object as the target view and aligned with the object's bounding box axes (Figure 3). The external cubemap is rendered with only the illuminated object without the environment.

The light incident at a surface point x_{in} is scattered into the material according to the Fresnel term $F_t(\eta,\vec{\omega}_{in})$. So the irradiance of each

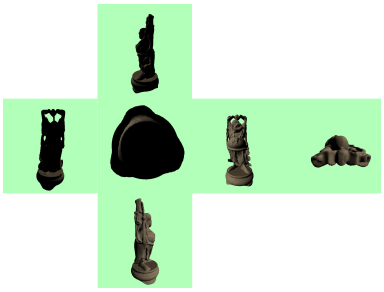


Figure 3: The external cubemap of the Happy Buddha.

pixel in the cubemap is scaled by the transmittance Fresnel term $F_t(\eta, \vec{\omega}_{in})$, that can be approximated as proposed in [Schlick 1994].

After we have computed the external cubemap of the object, we have to accumulate the irradiance in x_a, the accumulation points. We should also calculate an x_a point for each vertex of the model and this operation is computationally expensive. If we have a convex object we can approximate these points with only one sample point (placed in the center of bounding box of the object), which is parameterized by a direction. Moreover we can calculate the $B(\vec{n}_{out})$ (now parametrized by n_{out} the normal of point we want to shade) using [Ramamoorthi and Hanrahan 2001a], a fast approximation of the analytic solution. Which solves:

$$B(\vec{n}_{out}) = \int_{\Omega(\vec{n}_{out})} E(\vec{\omega})(\vec{r}(\vec{n}_{out}) \cdot \vec{\omega}) d\vec{\omega} \tag{5}$$

in $O(n)$ rather than $O(n^2)$ of the brute force integration, where $r(\vec{n})$ is refraction vector which depends by n_{out}. Finally we project the irradiance of an environment map into a spherical harmonic basis functions using only 9 coefficients, which are enough for a diffuse effect. This is achieved by integrating the cubemap against the spherical harmonic basis functions:

$$E_{lm} = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} E(\theta,\phi) Y_{lm}(\theta,\phi) \sin\theta d\theta d\phi \tag{6}$$

where $E(\theta,\phi)$ is the irradiance of the external cubemap, $Y_{lm}(\theta,\phi)$ are the spherical harmonics functions. Using the parametrization of a direction vector $s = (x,y,z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ the basis is defined as:

$$Y_{lm}(\theta,\phi) = K_{lm} e^{im\phi} P_l^{|m|} \cos\theta \qquad l \in \mathbb{N}, \quad -l \leq m \leq l \tag{7}$$

where $P_l^{|m|}$ are the associated Legendre polynomials and K_{lm} are the normalization constants:

$$K_{lm} = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{8}$$

Since we are interested on real valued functions we can define a real-valued basis, and it is given by the follow equation:

$$Y_{lm} = \begin{cases} \sqrt{2}\mathrm{Re}(Y_{lm}) & m > 0 \\ \sqrt{2}\mathrm{Im}(Y_{lm}) & m < 0 \\ Y_{l0} & m = 0 \end{cases} \tag{9}$$

It is worth noticing that this process is done one time for each color channel in the RGB color space.

2.2 Evaluation Step

In order to obtain the term $B(\vec{n}_{out})$ we evaluate the spherical harmonics using the refraction vector $\vec{r}$ for the look up. We use the equation presented in [Ramamoorthi and Hanrahan 2001a] for efficiently evaluating spherical harmonics for system optimized for matrix and vector operations such as the GPU:

$$B(\vec{n}_{out}) = \vec{r}(\vec{n}_{out})^t M \vec{r}(\vec{n}_{out}) \tag{10}$$

where M is a symmetric matrix 4×4 (there are three matrices, one for each RGB channel), in which each entry depends on the coefficients E_{lm}:

$$M = \begin{pmatrix} c_1E_{22} & c_1E_{2-2} & c_1E_{21} & c_2E_{11} \\ c_1E_{2-2} & -c_1E_{22} & c_1E_{2-1} & c_2E_{1-1} \\ c_1E_{21} & c_1E_{2-1} & c_3E_{20} & c_2E_{10} \\ c_1E_{11} & c_2E_{2-1} & c_2E_{10} & c_4E_{00} - c_5E_{20} \end{pmatrix}$$
$$c_1 = 0.429043 \quad c_2 = 0.511664$$
$$c_3 = 0.743125 \quad c_4 = 0.886227 \quad c_5 = 0.247708 \tag{11}$$

Finally, we scale the term $B(\vec{n}_{out})$ calculated with equation (10), in order to take into account of the density of material:

$$L^e(\vec{x}_{out}, \vec{\omega}_{out}, \vec{n}) = \frac{1}{\pi} F_t(\eta, \vec{\omega}_{out}) \frac{(\sigma_{tr} d + 1) e^{-d\sigma_{tr}}}{\sigma_t' d^3} B(\vec{n}) \tag{12}$$

where σ_t' is the reduced extinction coefficient, σ_{tr} is the effective extinction coefficient, and d is the distance from point x_{in} to point x_{out} (Figure 5). This attenuation function was inspired by the dipole approximation equation R_d [Jensen et al. 2001].

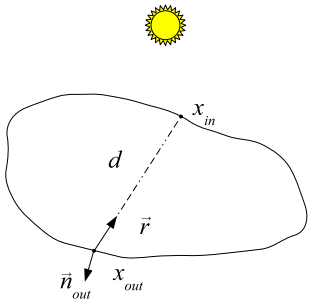
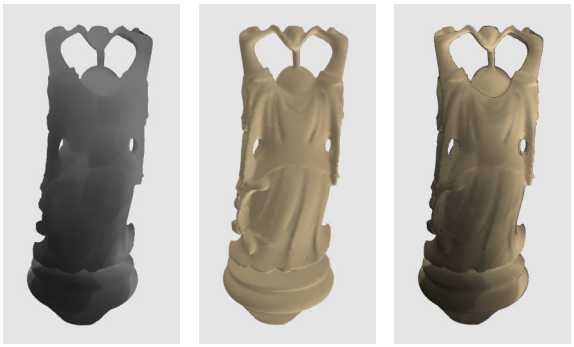


Figure 4: The evaluation step: the refraction vector $\vec{r}$ is used to fetch spherical harmonics to get radiosity term, which is scaled by a function of thickness d.

The distance d is calculated using a shadow map [Williams 1978] captured from the light's position. We fetch the depth value of x_{in} from the shadow map using the position of the current vertex, then we subtract this value by the distance from the vertex to the light position x_{out}. With this step we get the thickness of the object.

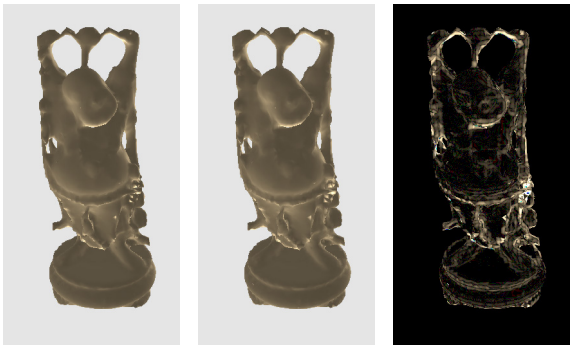


(a) (b) (c)

Figure 5: The evaluation step: a) the attenuation function based on the thickness of the object. b) the radiosity term evaluated using spherical harmonics. c) the radiosity term scaled by the attenuation function.

3 Implementation

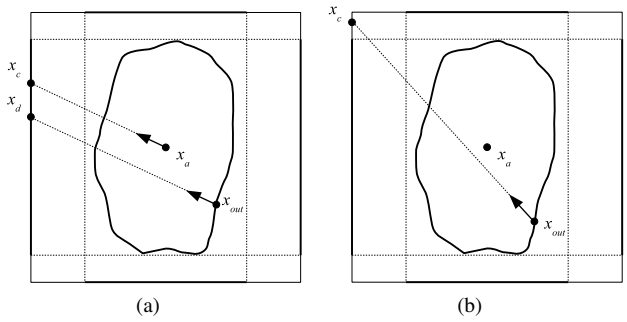
We implemented the proposed technique on GPU in order to speed up spherical harmonics projection step and the evaluation step. We inserted this code in an already existing real-time 3d framework based on DirectX9c and HLSL.



(a) (b) (c)

Figure 6: The radiosity term evaluation of the Happy Buddha at low resolution(15K triangles): a) radiosity term evaluated per pixel. b) radiosity term evaluated per vertex. c) Difference scaled 10x.

For the spherical harmonics projection step, in an off-line process, we precomputed the coefficients Y_{lm} for the filtering and we stored them using textures. Then in real-time, for each frame, we calculate the coefficients E_{lm} and the accumulation of each coefficient is done by a pixel shader which performs a MIP-Mapping like scheme. For more detail refer to a similar algorithm presentend in [King 2005]. There are two problems during the evaluation of spherical harmonics, the *shift* and *blindness* problems. The first is caused by the evaluation of spherical harmonics in the accumulation point x_a (Figure 7.a), so direction is shifted from what we really want. In the second problem the refraction vector can point toward zones in which the object is not rendered during the creation of external cubemap, and so in these directions the spherical harmonics intensity will probably very low. The reason is that in the general case a bounding box is not a cube, so when we render external cubemap the faces of cubemap will not fit the bounding box(Figure 7.b). The *shift problem* was solved using a ray-box intersection (the origin of the ray is the point to be evaluated, and the direction the refraction vector in that point) in a pixel shader, in this way we capture the exact intersection point used to lookup in the cubemap. The *blindness problem* was solved in a simply way. We scaled the object during the render to texture for generating the external cubemap, in order to transform is bounding box into a cube (Figure 8).



(a) (b)

Figure 7: The two problems during evaluation of spherical harmonics: a) *shift problem* the refraction vector is shifted because we evaluate spherical harmonics in the accumulation point x_a. b) *blindness problem* the refraction vector can point toward zones in which the object is not rendered during the creation of external cubemap, and so in these directions the spherical harmonics intesity will probably very low.

For the spherical harmonics evaluation step, we send the matricies of E_{lm} to the GPU and we evaluate directly in shader equation (4), because GPUs are systems optimized for matrix and vector operations. If the object does not use bump mapping the spherical harmonics evaluation could be done in the vertex shader in order to speed-up, because the values vary slowy as a function of the refraction vector used to look up (Figure 6). The attenuation function is computed per-pixel because its value changes fast to be calculated per vertex.

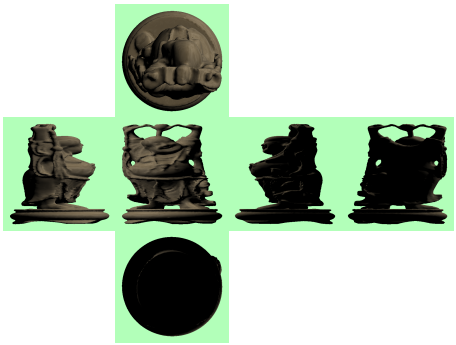


Figure 8: The external cubemap scaled for solving the *blindness problem*.

Note that if the object is concave we calculate an approximated thickness, because the information on occlusions are not saved (Figure 9). This can lead to errors and ghost like effects during the visualization of the object. So we solved this problem using depth of peeling [Everitt 2001] which is an image space technique that allows to get for each pixel the second nearest fragment, third nearest, and so on. In this case we need only the second depth value to check if the point that we are shading receives or not directly the scattering term. So only one pass of the algorithm is performed.

Finally the shadow map is filtered with a low pass filter. For this purpose we used a separated gaussian filter with a kernel of size 8×8, in this way we perform simple anti-aliasing (we can not apply PCF [Reeves et al. 1987], because we are working on thickness and not on a boolean property as shadow), the filtering is applied in hardware using a pixel shader to speed-up computation.

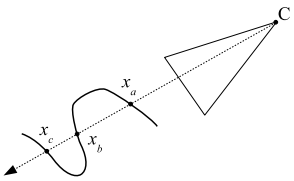


Figure 9: Calculation of thickness: for point x_b the thickness is correctly evaluted, however for point x_c if we calculate thickness respect light we get errors because the point is occluded by x_b. For point like x_c we do not calculate subsurface scattering in order to avoid ghosting effects.

Model	Triangles	fps
Bunny	69K	65.2
Happy Buddha	67K	67.9
Dragon	47K	83.6

Table 1: The results for the test models.

4 Results

For all the results we have used an Intel Pentium4 2.8 GHz PC, equipped with 1 GB RAM and an NVIDIA GeForce6800 graphics card with 128MB RAM. The size of the rendering viewport was 512×512 in window mode, the faces of cubemap were 64×64, with a shadow map of 512×512. We initially tried to use a higher resolution (128×128 and 256×256) for the faces of the cubemap to improve quality, but the values E_{lm} did not significantly change, so we chose the resolution of 64×64. Also we calculated equation (12) per pixel and not per vertex.

The precomputation of Y_{lm} coefficients was performed by CPU and took less than a second. These values could also be saved in a texture to avoid the calculation every time the application starts. Note that the object could be deformable (mesh morphing, skinning, free form deformation, etc.), and the light's parameters could change while the application is running. This is possible because the technique does not use precomputed information about the mesh, since it calculates at each frame the E_{lm} coefficients and the shadow map (only Y_{lm} coefficients are precomputed because they are always the same).

For the tests we used the Stanford's models Happy Buddha, Bunny and Dragon (Figure 10, Figure 11). In Figure 12 displays the technique with deformable mesh, we applied to Happy Buddha a sine animated deformer as a function of time and z coordinate, to show the capabilities with animated mesh.

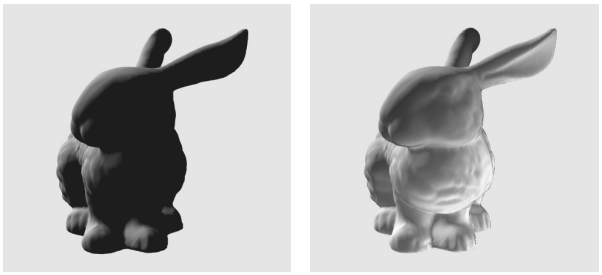
Finally, Figure 13 provides a visual comparison of the quality of our results, showing that our method produces a good appearance of subsurface scattering effects.

5 Conclusion and Future Work

In this paper we presented a fast, GPU friendly, translucency model suitable for convex objects. As the main projection step and evaluation can efficiently exploit the current graphics hardware, this technique allows the interactive applications to represent translucent materials in real-time at a low cost and without any constraint in terms of light conditions and animations (deformable meshes are supported). The technique could be used with concave objects be-

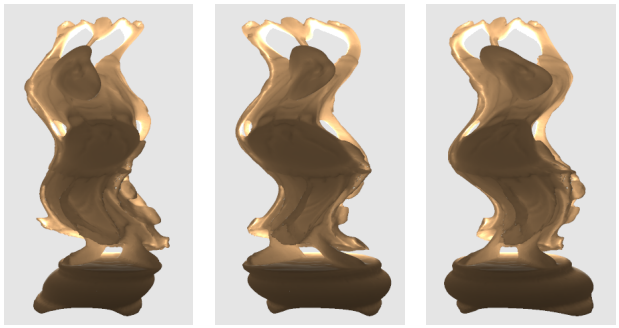
(a) (b)

Figure 10: The Dragon: a) direct lighting only. b) subsurface scattering.



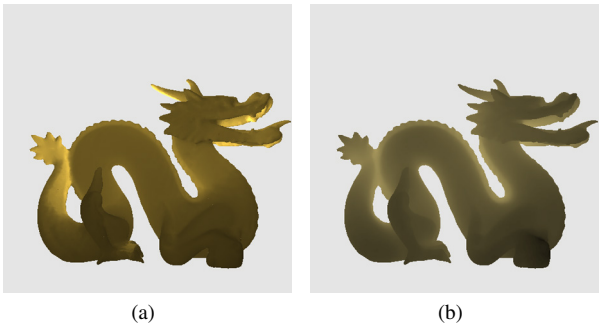
(a) (b)

Figure 11: The Bunny: a) direct lighting only. b) subsurface scattering.



(a) (b) (c)

Figure 12: Three frames of a simple animated sine deformer applied to Happy Buddha, in order to show capabilities of the algorithm using deformable and animated meshes.



(a) (b)

Figure 13: A comparison figure: a) our technique b) the result using Precomputed Radiance Transer.

cause errors in the projection step are often visually unnoticeable.

In the future we would like to explore how to extend the model to achieve higher fidelity result. A possible extension would be to allow more accumulation points, and interpolating values during the evaluation step. We would also like to investigate the perceptual fidelity of our approximated models with the physically based solutions and thereby improve the perceptual quality of our approach. Finally we will work on improving the anti-aliasing for distant light with the Perspective Shadow Mapping [Stamminger and Drettakis 2002].

6 Acknowledgments

The 3D models are courtesy of the Stanford University Computer Graphics Lab. Thanks to Maurizio Sciglio, Emanuele Salvucci, and Marco Salvi for their helpful comments. This work reported in this paper has formed part of EPSRC grant EP/D032148 whose funding and support is gratefully acknowledged.

References

CARR, N. A., HALL, J. D., AND HART, J. C. 2003. Gpu algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 51–59.

DACHSBACHER, C., AND STAMMINGER, M. 2003. Translucent shadow maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 197–201.

DALY, S. 1993. The visible differences predictor: an algorithm for the assessment of image fidelity. 179–206.

DORSEY, J., EDELMAN, A., JENSEN, H. W., LEGAKIS, J., AND PEDERSEN, H. K. 1999. Modeling and rendering of weathered stone. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 225–234.

EVERITT, C. 2001. Interactive order-independent transparency. In *white paper: http://developer.nvidia.com/object/order_ independent_ transparency*.

FLEMING, R. W., JENSEN, H. W., AND BüLTHOFF, H. H. 2004. Perceiving translucent materials. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, ACM Press, New York, NY, USA, 127–134.

HANRAHAN, P., AND KRUEGER, W. 1993. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 165–174.

HAO, X., AND VARSHNEY, A. 2004. Real-time rendering of translucent meshes. *ACM Trans. Graph. 23*, 2, 120–142.

JENSEN, H. W., AND BUHLER, J. 2002. A rapid hierarchical rendering technique for translucent materials. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 576–581.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 511–518.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA.

KING, G. 2005. *GPUGems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, ch. 10 - Real-Time Computation of Dynamic Irradiance, 167–176.

LENSCH, H. P. A., GOESELE, M., BEKAERT, P., KAUTZ, J., MAGNOR, M. A., LANG, J., AND SEIDEL, H.-P. 2002. Interactive rendering of translucent objects. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 214.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385–392.

MARTIN, T., AND TAN, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *Rendering Techniques*, 153–160.

MERTENS, T., KAUTZ, J., BEKAERT, P., SEIDELZ, H.-P., AND REETH, F. V. 2003. Interactive rendering of translucent deformable objects. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 130–140.

PHARR, M., AND HANRAHAN, P. 2000. Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 75–84.

PHARR, M. 2002. *RenderMan in Production Siggraph 2002 Course*. http://www.renderman.org, ch. 3 - Light/Surface Interaction, 55–72.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 497–500.

RAMAMOORTHI, R., AND HANRAHAN, P., 2001. the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 283–291.

SCHLICK, C. 1994. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum 13*, 3, 233–246.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3, 382–391.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. Graph. 24*, 3, 1216–1224.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 557–562.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 270–274.