EG UK Theory and Practice of Computer Graphics (2009)
Ik Soo Lim, Wen Tang (Editors)

Real-time rendering of deformable translucent objects

Nadir BENMOUNAH, Vincent JOLIVET and Djamchid GHAZANFARPOUR

XLIM, Universite de Limoges, France

Abstract

This paper introduces an efficient real-time shading model to simulate light transport in translucent materials. In this work no pre-processing step is needed, allowing us to deform translucent objects at interactive rates. The proposed technique avoids the pre-processing of texture atlas and texture coordinates that may entail distortions and self-occlusion errors. We create a cube map texture atlas on the GPU by a specific projection providing accurate access to neighborhood information is accurate. The texture atlas is generated on-the-fly *each time the mesh geometry is modified, allowing us to deform the object geometry in real time keeping valid the subsurface light transport computation.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Rendering techniques

1. Introduction

Many common real objects that we encounter in our daily life are neither opaque nor transparent: they are translucent, presenting a certain light penetration inside the material, while it does not permit us to see through it. Subsurface scattering describes this phenomenon. When light strikes a translucent material it penetrates the body, piercing the surface. This light penetration brings to the material a soft and a smooth appearance, hiding and blurring small geometric details.

To handle light interaction with translucent materials, Nicodemus [NRH*77] has proposed the Bidirectional Subsurface Reflectance Distribution Function (BSSRDF). The BSSRDF proposed by Nicodemus is an expensive model in terms of computing times. A ray of light entering the material and then scatters around before leaving the surface at a different location. Bidirectional Reflectance Distribution Functions (BRDF) are a less general formulation of the BSSRDF and are not suitable for modeling light interaction with translucent materials. BRDFs give the material a rough aspect, assuming that light enters and leaves the surface at the same point. Fig. 8 shows a comparison between an object rendered with subsurface scattering using our method (left), and the same object neglecting the internal light transport (right).

Many recent techniques have implemented subsurface scattering interactively on the GPU. Unfortunately, they need to pre-compute adequate texture coordinates parametrization capturing the translucent object geometry in a texture atlas. In this texture atlas, each texel corresponds to a unique location on the mesh. Obviously, the pre-computations limit the possibility of deforming translucent objects. To compute the BSSRDF, each mesh location needs to acquire its neighborhood information; Moreover, the texture parametrization remains empirical and entails many distortion and self-occlusion errors, decreasing the accuracy of the subsurface transport computations.

Our method deals with most cases of distortions and self-occlusions that may occur during projection. The method we present in this paper proposes alternative and accurate solutions to allow deformation of translucent objects and to enhance the perception of translucency. We depict an intuitive and a formal method for acquiring neighborhood data. A cube map is generated *on-the-fly* with a specific projection on the axis-aligned bounding box containing the mesh. The projection allows a direct and an intuitive access to the texel that corresponds to each mesh location and its neighborhood. The projection and the look-ups are implemented on the GPU to achieve real-time rates. Moreover, it is possible the deform objects since no pre-computing phase is needed.

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/189-198

The remaining parts of the paper are organized as follows. We present previous work on translucent rendering in Section 2. In Section 3 we introduce the translucent illumination model based on the BSSRDF model. Then, Section 4 presents our methods. In Section 5, we discuss the results obtained and finally, Section 6 describes future work.

2. Previous work

A wide range of methods and techniques have been proposed for translucent rendering. Most researchers aspired to obtain real-time rates in solving the high-consuming volume rendering equation for participating media. Many solutions dealing with the participating media have been proposed; finite element method [RT87, RT90] , path tracing [HK93, LW96] and photon mapping [Jen96, DEJ*99]. Hanrahan and Krueger [HK93] have derived a BRDF model for subsurface reflection that analytically computes the subsurface light transport for a homogeneous material. They take into consideration the biological properties of skin layers to achieve realistic rendering. This technique is able to produce accurate subsurface effects, but is very time consuming.

The diffusion equation is an alternative for subsurface scattering calculation, which is a close approximation of the volume rendering. Jensen *et al.* [JMLH01] exposed an approximation of Nicodemus's BSSRDF by introducing the dipole diffusion approximation that consists in positioning two point sources near the surface respecting the required boundary conditions. The dipole is a diffusion simulation of the multiple scattering. Jensen and Buhler [JB02] modify this method [JMLH01], to reduce computing times by gathering the irradiance from external light, then transferring these irradiances to the other surface patches using a hierarchical integration. Unfortunately, this technique does not appear to allow interactive rendering.

Haber *et al.* [HMBR05] have proposed a numerical solution of the diffusion equation. They succeeded using an octree discretization and a multigrid solver to efficiently and numerically compute the diffusion equation in complex objects topology. Their method manipulates any concave or convex geometry, but it is not suited to real-time, and may show some artifacts on dense materials.

Many real-time methods have been proposed: the subsurface equation involves calculating matrix radiosity as shown by Lensch *et al.* [LGB*02] where the formulation of Jensen's BSSRDF resembles a single radiosity gathering step. Here, subsurface scattering is approximated by filtering the incident illumination stored in a texture atlas. Interactive rates have been reached by their method but the use of the texture atlas induces many problems, such as self-occlusions and distortions. It has been shown [CHH03] that Lensch *et al.* work can be completed by introducing a hierarchical mesh atlas to perform the radiosity gathering step. This method is half executed on the GPU, which allows interactivity, but it does not support deformable meshes because of a necessary pre-computation phase.

Mertens *et al.* [MKB*03] use a clustering hierarchy, which is re-built in real-time to render deformable translucent objects interactively. They also propose a local subsurface scattering algorithm that integrates the samples in image space at interactive rates using hardware acceleration. However, this hardware acceleration does not concern their hierarchical radiosity clustering. The method deals only with fixed mesh hierarchy, and fails to handle adaptive or refinement meshing, where the irradiance is not constant over a triangle.

A different approach has been presented in [SHHS03]. In this approach the radiance transport and subsurface scattering of each vertex are pre-computed and projected onto 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 geometrically static models.

Another interactive method is depicted in [HBV03], where Hao *et al.* implement a real-time rendering approach, by pre-computing for each vertex light irradiance expressed in a spherical harmonics basis. Irradiances are then accumulated from the nearest vertices. Their technique is per-vertex, which may show artifacts and shading discontinuities, especially in low resolution meshes. Moreover, the method is not suited to deformable meshes.

Wang *et al.* [WTL05] proposed the first interactive method capable of pre-computing subsurface light transport of translucent materials lit under all-frequency lighting, using complex environment maps. Their method needs a pre-computing phase and fails to render deformable translucent objects.

D'Eon *et al.* [dLE07] present an efficient rendering algorithm of human skin based on polynomial multiplications and additions over Gaussians. Instead of performing the irradiance convolution on texture space by means of image filtering, the radiosity function is replaced by a sum of Gaussians. This generates many images, and the convolution of irradiances is then approximated by summing the generated images. The proposed Gaussians approximation produces real-time translucent rendering of human skin, but their technique has many limitations; The Gaussians approximation they propose is empirical; The method requires a texture parametrization, which is not an intuitive process. In our method, we do not need to pre-compute

texture coordinates.

3. Theory

The BRDF is a simplification of the BSSRDF, for which light enters and leaves the materials at the same location. To compute the BSSRDF, we need to integrate irradiances over all incident directions and for every location in the neighbourhood. The size of the neighbourhood is discussed in Sec. 5.

A formulation of the outgoing radiance at the point x_o in direction ω_o due to subsurface light transport is given by:

$$L_o(x_o,\omega_o) = \int_A \int_\Omega S(x_i,\omega_i;x_o,\omega_o) L_i(x_i,\omega_i)(N_i.\omega_i) d\omega_i dA(x_i) \quad (1)$$

where L_o is the outgoing radiance at point x_o in direction ω_o, L_i is the incident radiance at point x_i in direction ω_i, and finally S is the BSSRDF. Integration is done over all locations on the mesh in all the considered directions.

The dipole simplification brought by Jensen reduced the BSSRDF from an 8 to a 4 dimensions function, directionally independent and related only to the outgoing and the incoming surface positions:

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

$$B(x_o) = \int_S E(x_i) R_d(x_i;x_o) d(x_i) \quad (3)$$

Here, $B(x_o)$ is the radiosity at x_o due to the dipole source. The function R_d is known as the subsurface diffuse reflectance, and encodes the geometric data of the object and the physical and optical properties of its material.

$$E(x_i) = \int_\Omega L_i(x_i,\omega_i) F_t(\eta,\omega_i) |N_i.\omega_i| d\omega_i \quad (4)$$

$E(x_i)$ is the irradiance function, which gathers incoming light striking a location x_i scaled with F_t. F_t is the Fresnel transmittance. Finally, the definition of the function R_d is depicted on Table 1.

The diffuse reflectance function R_d is the most costly part of the translucency estimation since integration over all surface points is very expensive. Many techniques take up Jensen's model and act toward obtaining interactive rendering results.

$$R_d(x_i,x_o) = \frac{\alpha'}{4\pi}[z_r(\sigma S_r+1)\frac{e^{-\sigma S_r}}{{S_r}^3} + z_v(\sigma S_v+1)\frac{e^{-\sigma S_v}}{{S_v}^3}]$$

$$z_r = 1/\sigma_t'$$
$$z_v = z_r(1+4AD)$$
$$S_r = ||x_r - x_o||; \text{ with } x_r = x_i - z_r.N_i$$
$$S_v = ||x_v - x_o||; \text{ with } x_v = x_i - z_v.N_i$$
$$A = \frac{(1+F_{dr})}{(1-F_{dr})}$$

$$F_{dr} = \frac{-1.44}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta$$
$$D = 1/3\sigma_t', \ \sigma = \sqrt{3\sigma_a\sigma_t'}$$
$$\sigma_t' = \sigma_a + \sigma_s', \ \alpha' = \frac{\sigma_s'}{\sigma_t'}$$

σ_s'= reduced scattering coefficient.
σ_a=absorption coefficient.
η relative index of refraction.
$F_t(\eta,\omega)$ = Fresnel transmittance factor.

Table 1: *The dipole source BSSRDF model.*

4. Real-time rendering of translucent objects

In order to compute the outgoing radiance of a mesh location, we perform an accumulation of the irradiances of neighboring locations, modulated by the associated R_d. This accumulation is performed as a per-texel filter kernel in texture space. Our method needs no pre-computations, and the irradiances are computed *on-the-fly* to allow dynamic lighting and object deformation.

Two main subsurface scattering gathering philosophies have influenced our work, the first one is depicted in Dachsbacher *et al.* [DS03] and concerns the global scattering where a pre-computed shadow map is used to store irradiances at each mesh location, these data are then captured in a render target texture. Their method is not suitable for our requirements. Since, the pre-computation of the shadow map does not allow us to deform the meshes. The second philosophy is presented by Banterle *et al.* in [BC06]. Here irradiance is projected onto a spherical harmonics basis, using an external light cube map on which they project the object geometry, then the outgoing radiance is evaluated. The method enables subsurface rendering of deformable objects, but the simplified approximation of the diffuse dipole equation they propose fails to produce realistic results. The projection of the object geometry on cube map caught our attention, and we tried to adopt it to compute local scattering.

4.1. Irradiances computation

The translucent objects we consider are illuminated with single point lights or environment lighting.

4.1.1. Irradiance computation from a single-point light source

The irradiances are computed at run-time, in a pixel shader, to allow dynamic lighting. To compute the irradiance of a mesh location, we need its position and normal data. The computed irradiances are stored in textures.

4.1.2. Irradiance from environment maps

To compute the irradiance at each mesh location (Eq. 4), we need to integrate over all incoming directions. $E(x_i)$ is approximated in terms of spherical harmonics coefficients [Gre03]. The spherical harmonics projection reduces the integral to a simple dot product of a small number of spherical harmonics coefficients. For that, we project $L_i(x_i,\omega_i)$, $F_t(\eta,\omega_i)$ and $G = |N_i.\omega_i|$ onto the spherical harmonics basis $Y_l^m(\theta_i,\phi_i)$.
The spherical harmonics *basis* Y_l^m is given by :

$$Y_l^m(\theta_i,\phi_i) = \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}} P_l^m(cos\theta_i) e^{im\phi_i} \quad (5)$$

The total irradiance is computed as the dot product of the obtained coefficients. The $Y_l^m(\omega_i)$, the $L_i(x_i,\omega_i)$ and the $F_t(\eta,\omega_i)$ do not depend on the geometry (normals). Their m spherical harmonics components can be estimated in an offline step and stored in textures.

To project the incoming radiance $L_i(x_i,\omega_i)$ (Eq. 6) stored in the environment map we perform a summation of each incoming radiance multiplied by $Y_l^m(\omega_i)$ and the solid angle $A(\omega_i)$ occupied by the pixel on the unit sphere around the center of the cube.

$$L_{l,m} = \sum_{face} \sum_{i=1}^{size} \sum_{j=1}^{size} L_{face}(i,j) Y_l^m(\omega_i) A(\omega_i) \quad (6)$$

The Fresnel transmittance term is projected using the following equation.

$$F_{l,m} = \int_{\theta_i} \int_{\phi_i} F_t(\eta,(\theta_i,\phi_i)) Y_l^m((\theta_i,\phi_i)) sin\theta_i d\theta_i d\phi_i \quad (7)$$

The cosine term function depends on the geometry and should ideally be projected *on-the-fly*. Estimating the spherical harmonics coefficients of the cosine term on the GPU is possible, but is time-consuming. We can notice that the same normal always returns the same spherical harmonics coefficients values of G. We propose to compute these values for any normal in an offline process, and store the results in m 3D textures, indexed by the surface normal.

The computation of irradiance at a mesh location is performed with a dot product of the m light, Fresnel transmittance and the SH basis function coefficients (See Eq. 8). The dot product is performed in a pixel shader, which have to look up for coefficients in the different textures.

$$E(x_i) = \sum_{l,m} G_{l,m} F_{l,m} L_{l,m} Y_{lm} \quad (8)$$

4.2. Projection and texture lookups

The pre-computation of the texture atlas involves many drawbacks. The projection of the mesh on a single plane may introduce some distortions and self-occlusions problems. Moreover, the pre-definition of the texture coordinates does not allow object deformation in real-time. The algorithm we present reduces significantly distortion and self-occlusion problems, and removes the need to pre-calculate texture coordinates.

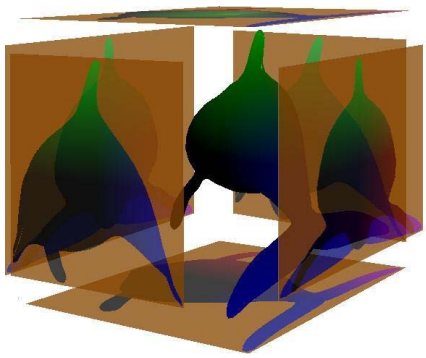

Figure 1: *Projection on cube map.*

The mesh is projected orthogonally on each cube map face of the axis-aligned bounding box that encloses its geometry (see Fig. 1). For both types of illumination, the irradiances are rasterized into textures on which we project the object. The pixel shader that computes irradiances renders the mesh to the cube map. Six cameras looking at each face of the cube map are used to store the cube map in textures. These textures can later be sent to the a second pixel shader using render targets.
The localization of each fragment on the selected face texture is then intuitive. It is simply the texel that corresponds to the perpendicular projection of the mesh location on the considered face. Thus, the texture coordinates are defined

on-the-fly and in a formal way.

A second pixel shader is used to perform irradiances accumulation and the outgoing radiance calculation. This pixel shader is fed with the six faces of the cube map (irradiances cube map). To achieve the accumulation step, we need to look for neighborhood data of each point to render in the cube map: The normal of the fragment to render defines a plane tangent to the mesh surface at this fragment Fig. 2. The tangent plane divides the 3D space into two half-spaces. We assume that the face storing this fragment (selected face) and its neighborhood data is one of the faces (chosen faces) having at least one vertex belonging to the upper half-space. Scalar products are performed of the fragment normal and the normals of each chosen face. The face with the greatest scalar product is selected. As the scalar product increases in magnitude, the projection becomes more faithful.

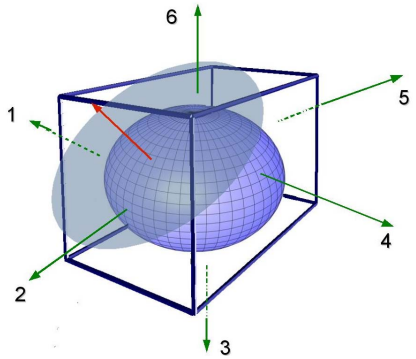

Figure 2: *The tangent plane defines two half-spaces. Only the faces 1,2,4,6 are in interaction with the upper half-space, and are chosen.*

4.3. Local scattering

To compute the local outgoing radiance of a mesh location, we need to accumulate the contributions of irradiances at neighboring locations (Eq. 3). This accumulation is performed as a per-texel filter kernel in the immediate neighborhood for each texel of the selected face.

The projection on the cube map may induce some self-occlusions and distortions problems. We propose to deal with these problems in the two following subsections.

4.3.1. Self-occlusions

The chosen faces are sorted according to the associated scalar products, from the greatest to the smallest, as shown in Fig. 3. Self-occlusions happen when a mesh location is not visible on the selected face. Fig. 4 illustrates this case. When we are to render the location M, we look for its irradiance on the selected face in C location, but the location C occludes M.

During the creation of the irradiances cube map, the scalar product of the normal and the position of each fragment is stored in the alpha component of its corresponding texel in the cube map. The self-occlusion is detected by performing a test comparing the scalar product of the real 3D position and the normal of the point in the mesh, and the eventual scalar product stored in the alpha component of the selected face. If the test fails, the next face likely to contain the mesh location and its neighborhood data, is the face with the next greatest associated scalar product; The process is iteratively repeated until the true face is selected.

Because of some kind of concavities, a point may not be visible on any face of the bounding cube. In this case, we assume that its local contribution is negligible, since it does receive any irradiance. It's important to note that these hidden locations provide a global scattering contribution and is performed efficieltly as explained in Sec. 4.4.

4.3.2. Distortions

The cube map projection we propose in this paper reduces significally distortions since the geometry is projected on six different planes. Moreover, the selected face shows the weakest inclination with the projected surface, therefore the projection is more faithful.

However, if the selected face has a small scalar product, far texels may be brought close due to the inclined projection. In [JB02] authors showed that the maximum distance a photon travels into a material, is the mean-free path given by $l_d = 1/\sigma_t'$. The mean free path corresponds to the radius of the kernel filtering. We propose to use an adaptive kernel filtering size. Instead of using a fixed kernel size (maximum kernel size) $k_{max} = l_d$, the adaptive kernel size is obtained by multiplying the mean free path by the scalar product associated with the selected face $k = sp.l_d$. Here, the kernel size is corrected by the associated scalar product sp.

4.4. Global scattering

The points located on opposite surfaces of thin regions may be projected on many distinct faces, and will be far in texture space. The contribution of near points located on the other side of a thin area must be included. Fig. 5 shows that case. The contribution of the global response is lower than the local response, but neglecting it as in [MKB*03] gives the object a bad realism and inhibits the perception of depth, essentially for high translucent materials. We propose a fast

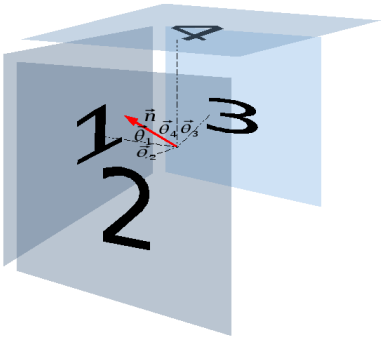

Figure 3: *Faces sorting,* $\Theta_1 \langle \Theta_2 \langle \Theta_3 \langle \Theta_4$.

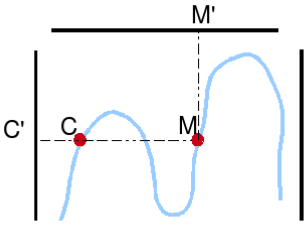

Figure 4: *The M location is projected on the upper and the left face. Only the upper face is selected.*

approximation of the global scattering. The approximation we performed needs no convolution process.

The dipole approximation introduced in [JMLH01] replaced the true source with two different sources, the real light source and the other, the virtual source near the surface on the contributing locations; In our method, we modify the diffuse reflectance R_d due to dipole source by considering the real source as the true source, and cancelling the virtual source effect. The global diffuse reflectance R_g is then:

$$R_g(x_s,x) = \frac{\alpha'}{4\pi}[z_s(\sigma S_s+1)\frac{e^{-\sigma S_s}}{{S_s}^3}] \qquad (9)$$

With x_s the source position. To estimate the radiosity $B(x)$ a simple product between the irradiance $E(x)$ at x and R_g is done:

$$B(x) = E(x)R_d(x_s;x) \qquad (10)$$

This modification allows every surface location, even the buried locations to receive their global scattering response.

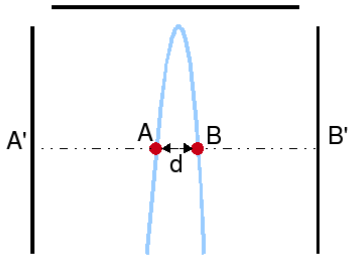

Figure 5: *Near locations projecting on far texture positions. The distance d is small enough (less than the mean free path) to consider the points A and B as far locations. Here, our global scattering algorithm corrects the subsurface estimation.*

To obtain the final rendering, a simple addition is then performed of the global and the local terms, see Fig.9.

4.5. Deformability

The pre-processing of texture atlases presents a drawback for real-time rendering of translucent deformable objects. In our method, the cube map projection has been implemented on the GPU, and permits detection of every mesh location displacement at run time. The irradiances are stored in render to target textures, which are updated at each frame. For each vertex displacement, the new values of irradiances are re-computed and stored in cube maps. Once the render irradiances cube map is created, it is sent to the second fragment shader that performs the irradiances accumulation to evaluate the outgoing radiance *on-the-fly*. For deformable models, this computation has to be performed at each frame.

To test our methods, we have implemented a morphing between a mesh model and its bounding sphere. Our algorithm keeps correct the subsurface transport computation while the object geometry is altered. In Fig. 10, a high translucent buddha object, lit from behind with a single-point light source is deformed, we notice that thin areas always look more lit than the thick regions. We also tested our method on low translucent material such as porcelain. This is done by introducing the scattering and absorption factors corresponding to that material. On Fig. 11, we show 3 frames of a morphing sequence of a porcelain object, illuminated by an environment map.

5. Implementation and results

The methods we proposed can be summarized as shown in the following pseudocode:
Compute irradiances:
Send pre-computed spherical harmonics components

to the first pixel shader
Perform irradiances
Render irradiances in cube map

For each mesh location do
Send the irradiances cube map to the second pixel shader
Local scattering:
The normal defines 2 half-spaces
Choose faces that may contain the mesh location
Sort the chosen faces according to the Scalar Products
For the greatest SP to the smalest SP
If the mesh location is present on the face
The chosen face is selected
Break
End If
End For
Compute the optimal kernel filtering size
Retrieve irradiance
Perform irradiances filtering
Compute the local scattering
Global scattering:
Compute global scattering

Final rendering:
The mesh location color = local scattering + global scattering
Render mesh in 3D
End For

The algorithm is executed every frame, whether the geometry is being modified or not.

Our method is very parameterizable. We can change optical parameters such as scattering and absorption coefficients, corresponding to different materials.
Since irradiances are captured at run-time, lighting and mesh geometry can be changed interactively. The validity and quality of our technique can be examined on the various figures below.

Depending on the considered neighborhood size, i.e. the maximum kernel filtering size, the frame rate is more or less important. Here, we want to find the ideal kernel size (without the product scalar correction) to apply. We noticed that a maximum 5x5 kernel is visually correct. Image analysis of translucent objects images has been used to validate perception of translucency. As noted in [FJH04] the more the object is translucent, the less the contrast in its image is emphasized, involving a uniformity in the intensity distribution. When the object is altered from translucent to opaque, its histogram changes shape, from uniformly distributed to a skewed shape. Fig. 6 presents a translucent object with different maximum kernel sizes, and the corresponding histogram, we notice that the intensities distribution tends to remain the same, from the 5x5 kernel.

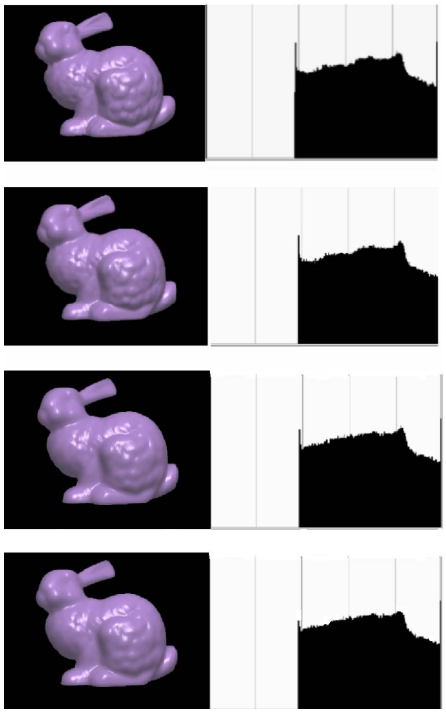

Figure 6: *From up to down, a translucent object using a 3x3, 4x4, 5x5, 6x6 kernel size. The two last images have the same intensity distribution.*

Our algorithm deals efficiently with high translucent material, where global scattering contribution is more important than the local one such as wax. Fig. 7 shows a wax candle lit by its flame (single-point light source). Wax tends to scatter light in different directions, producing a light-saturated material.

Figure 7: *Red wax candle*

Table 2 shows the performance of our implementation of

the subsurface algorithm. The timings were collected on an AMD Athlon 64 X2 Dual Core with 2.2 GHZ and NVidia GeForce 7900 GTX graphics card and confirm the interactive aspect of our method. The high resolution buddha mesh is rendered at 22 fps while it is deformed, even if the rate is somewhat low, the fluidity of rendering is ensured as is shown in our animations.

Model	Triangles	FPS with deformation	FPS without deformation
Hand	7700	72	120
NVidiaDemon	48174	49	81
Bunny	72223	43	71
neptune	227220	25	41
Buddha	381500	22	28

Table 2: *This table lists the frames per second, the number of triangles for each models we tested with our implementation, on deformable and non deformable meshes.*

6. Discussion and future work

We have presented an efficient and real-time method to accurately render deforming translucent materials based on the dipole approximation. The rendering is interactive, under dynamic viewing and illumination conditions. At run time, our method is able to keep correct translucency information, while the geometry is changed. It is also possible to change scattering and absorption coefficients at run time, which can be useful for the user to tune those parameters until he reaches the desired material aspect.

No pre-processing step is needed, which allows an access to a large range of translucent rendering applications, and accelerates by the way, the execution of the implemented algorithms without caring about memory storage.

The solution we brought may fail to gather the neighborhood information for some topologies. Some internal visibility problems caused by concavities and voids are the important drawback of the dipole approximation. This problem is noticed on topologies containing holes inside the mesh, even if our algorithms can't compute local scattering of the locations present at those hidden parts of meshes, the global scattering estimates a part of their radiance. Moreover, for hidden parts, local scattering response tends to zero, because they are not visible from light sources.

In the future, we wish to improve our translucent rendering engine by solving numerically in a 3D grid the diffusion equation without caring about the visibility problems.

References

[BC06] BANTERLE F., CHALMERS A.: A fast translucency appearance model for real-time applications. In *SCCG 2006 - Spring Conference on Computer Graphics* (April 2006), ACM SIGGRAPH.

[CHH03] CARR N. A., HALL J. D., HART J. C.: Gpu algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 51–59.

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

[dLE07] D'EON E., LUEBKE D., ENDERTON E.: Efficient rendering of human skin. In *SR '07 Rendering Techniques* (Grenoble, France, 2007), Kautz J., Pattanaik S., (Eds.), Eurographics Association, pp. 147–157.

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 197–201.

[FJH04] FLEMING R. W., JENSEN H. W., HEINRICH: Perceiving translucent materials. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization* (New York, NY, USA, 2004), ACM Press, pp. 127–134.

[Gre03] GREEN R.: Spherical harmonic lighting: The gritty details. *Archives of the Game Developers Conference* (March 2003).

[HBV03] HAO X., BABY T., VARSHNEY A.: Interactive subsurface scattering for translucent meshes. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM, pp. 75–82.

[HK93] HANRAHAN P., KRUEGER W.: Reflection from layered surfaces due to subsurface scattering. In *Proc. of SIGGRAPH-93: Computer Graphics* (Anaheim, CA, 1993), pp. 165–174.

[HMBR05] HABER T., MERTENS T., BEKAERT P., REETH F. V.: A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects. In *GI '05: Proceedings of Graphics Interface 2005* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005), Canadian Human-Computer Communications Society, pp. 79–86.

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. In *SIG-*

GRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 2002), ACM Press, pp. 576–581.

[Jen96] JENSEN H. W.: Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)* (New York, NY, 1996), Springer-Verlag/Wien, pp. 21–30.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 511–518.

[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 *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2002), IEEE Computer Society, p. 214.

[LW96] LAFORTUNE E. P., WILLEMS Y. D.: Rendering participating media with bidirectional path tracing. In *Proceedings of the eurographics workshop on Rendering techniques '96* (London, UK, 1996), Springer-Verlag, pp. 91–100.

[MKB*03] MERTENS T., KAUTZ J., BEKAERT P., REETH F. V., SEIDEL H.-P.: Efficient rendering of local subsurface scattering. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 51.

[NRH*77] NICODEMUS F. E., RICHMOND J. C., HSIA J. J., GINSBERG I. W., LIMPERIS T.: *Geometrical Considerations and Nomenclature for Reflectance*. US Dept. of Commerce, National Bureau of Standards: for sale by the Supt. of Docs., US Govt. Print. Off., 1977.

[RT87] RUSHMEIER H. E., TORRANCE K. E.: The zonal method for calculating light intensities in the presence of a participating medium. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM, pp. 293–302.

[RT90] RUSHMEIER H. E., TORRANCE K. E.: Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Trans. Graph. 9*, 1 (1990), 1–27.

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

[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.

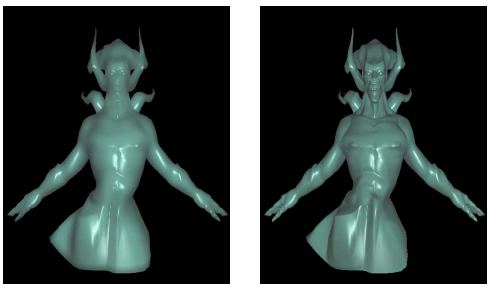

Figure 8: *At the left, a translucent mesh is rendered using* $\sigma_a = (2.58, 2.54, 1.6)$ *and* $\sigma_s^{'} = (0.00162, 0.00053, 0.00024)$. *At the right, a non translucent object is rendered using* $\sigma_s^{'} = (0.0, 0.0, 0.0)$

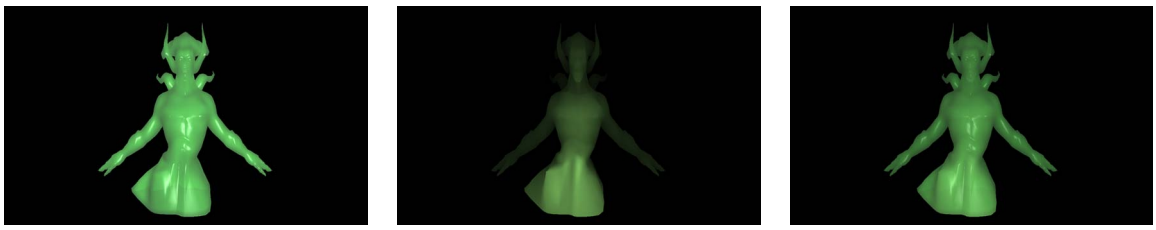

Figure 9: *The image at left shows an object with local scattering, the thickness of demon's horn does not look realistic. By adding the global term shown in the central image, the right image exhibits more correct rendering.*

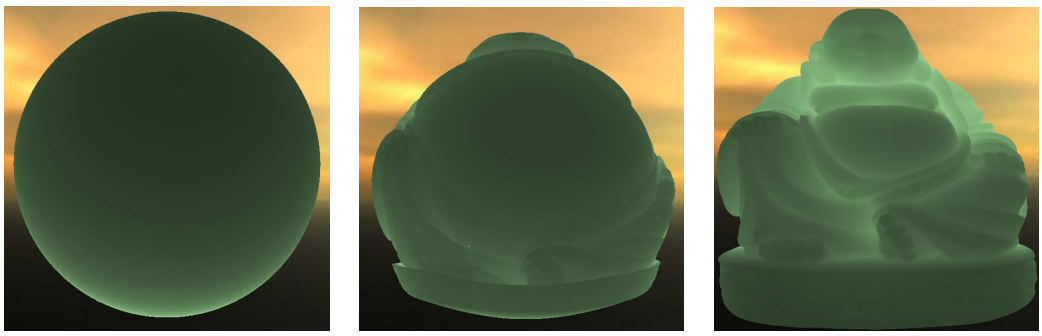

Figure 10: *A morphing of a high translucent object. The object is lit from behind with single-point light source.*

Figure 11: *A morphing of a porcelain object illuminated by an environment map.*