The Visual Computer
The final publication is available at Springer via http://dx.doi.org/10.1007/s00371-016-1207-2

Interactive Directional Subsurface Scattering and Transport of Emergent Light

Alessandro Dal Corso · Jeppe Revall Frisvad · Jesper Mosegaard · J. Andreas Bærentzen

Received: date / Accepted: date

Abstract Existing techniques for interactive rendering of deformable translucent objects can accurately compute diffuse but not directional subsurface scattering effects. It is currently common practice to gain efficiency by storing maps of transmitted irradiance. This is however not efficient if we need to store elements of irradiance from specific directions. To include changes in subsurface scattering due to changes in the direction of the incident light, we instead sample incident radiance and store scattered radiosity. This enables us to accommodate not only the common distance-based analytical models for subsurface scattering but also directional models. In addition, our method enables easy extraction of virtual point lights for transporting emergent light to the rest of the scene. Our method requires neither preprocessing nor texture parameterization of the translucent objects. To build our maps of scattered radiosity, we progressively render the model from different directions using an importance sampling pattern based on the optical properties of the material. We obtain interactive frame rates, our subsurface scattering results are close to ground truth, and our technique is the first to include interactive transport of emergent light from deformable translucent objects.

Keywords subsurface scattering · global illumination · interactive rendering · translucent objects · turbid media

A. Dal Corso (✉) · J. R. Frisvad · J. A. Bærentzen
Technical University of Denmark, Kgs. Lyngby, Denmark
E-mail: alcor@dtu.dk

J. Mosegaard
The Alexandra Institute, Aarhus, Denmark

1 Introduction

Subsurface scattering of light is a physical phenomenon that occurs in translucent materials. Milk, honey, skin, marble, and candle wax are just a few examples of translucent materials. It is possible to produce the qualitative appearance of translucency using interactive volume rendering techniques [32], but such techniques are not quantitatively accurate. With the advent of analytical models for subsurface scattering [26], it became feasible to build more accurate techniques for interactive rendering of translucent objects. The first technique of this kind [33], and more recent ones that also work for deformable objects (see Section 2), consider diffuse subsurface scattering only. In practice, this means that subsurface scattering is computed by evaluating an integral over the object surface of an analytic dipole model [26] that only depends on the distance between the points of incidence and emergence. Single scattering and other dependencies of the subsurface scattering on the direction of the incident light are neglected. Recent work in offline rendering however shows that the directional effects are not negligible [50,20,10,14].

We present an interactive technique that supports *directional* subsurface scattering without relying on precomputation or a grid for volumetric light propagation. To the best of our knowledge, our method is the first of its kind. Since the method does not rely on texture parameterization, it works for deformable and even procedurally generated geometry.

Due to reciprocity of light transport, we would ideally treat the directions of incident and emergent light equally. This is however too costly for an interactive technique. To achieve interactivity, we need caching of subsurface scattering computations. Existing techniques typically cache transmitted irradiance [25,33]

diffuse subsurface scattering | transport of emergent light | directional subsurface scattering

Fig. 1 Deforming translucent candle rendered interactively as with existing techniques (left block), with our transport of emergent light (middle block), and including directional subsurface scattering (right block). Our method is the first to support interactive rendering of the results in the right block (6 frames per second). For this scene, we use 28 scattered radiosity maps, 45 samples per direction, and 80 virtual point lights.

(total incoming light in a surface point) and use a precomputed filter to evaluate the subsurface scattering [33, 5, 29]. These techniques require that the subsurface scattering depends on distance only, whereas we need to use the direction of the incoming light. To cache another quantity, we note that subsurface scattering partly diffuses the light even if the incident light and the scattering are highly directional. Every ray of incoming light gives rise to a (non-diffuse) lobe of emergent light at all surface points. Adding up these lobes, the emergent light is in practice nearly diffuse. We therefore store scattered radiosity (outgoing light) instead of transmitted irradiance. Some of the directional subsurface scattering models also neglect dependency on the direction of emergence but still achieve improved accuracy [50, 20, 14]. Out of these, we can directly use the ones that do not rely on precomputation [20, 14].

In some existing techniques [33, 36, 4, 42], scattered radiosity is stored per vertex. To accommodate more detailed directional effects, we use more detailed maps of the scattered radiosity. We obtain these maps without requiring texture parameterization of the translucent object by rendering the object from multiple views using orthographic cameras. For each of these views, we compute a map of scattered radiosity. We can then efficiently render the translucent object from any view by look-ups into the scattered radiosity maps.

The scattered radiosity maps have two other important advantages. As long as the light source and the object are stationary, we can blend scattered radiosity maps and thereby progressively improve the rendering. Moreover, we can compute the transport of emergent light to the surrounding scene [40, 42]. To include these light paths while keeping the translucent object deformable, we generate a distribution of virtual point lights on the surface of the translucent object and set their intensity according to the scattered radiosity. These virtual point lights enable us to render the transported light using a many-light method [8]. Since we include transport of emergent light, our method is very useful for interactive rendering of scenes with the light source hidden behind a translucent object. Indirect illumination of a scene by light that has scattered through candle wax is one use case (Fig. 1). Another interesting example is light scattering through translucent lamp shades or light bulbs. To the best of our knowledge, we present the first interactive technique for transport of light emerging from deformable translucent objects.

2 Related Work

One way to obtain interactive subsurface scattering is by means of precomputation. Several early techniques rely on precomputed scattering factors that enable subsurface light transport between surface patches or from patch to vertex [33, 23, 4, 24]. These factors resemble form factors in radiosity algorithms and specify transport of transmitted irradiance to scattered radiosity. An extension of these radiosity-like techniques is to include transport of emergent light [42]. Other work is based on precomputed radiance transfer [43, 47, 49, 46], and some of this includes directional effects such as single scattering in the rendered result [43, 47, 49]. Another approach is to precompute a grid that can be used with a fast diffusion computation to render subsurface scattering in real-time [45, 48]. As opposed to our work,

all these precomputation-based methods cannot interactively render deformable translucent objects.

Some finite element methods are fast enough to enable interactive rendering of deformable translucent objects [36,34]. However, as these methods rely on diffuse incoming light (transmitted irradiance) and a multiresolution mesh (triangular or tetrahedral), they are not easily adapted for directional subsurface scattering and would typically require some mesh preprocessing.

Volume rendering techniques can quite convincingly produce the qualitative appearance of translucency at high frame rates [32,3,2,13]. While such methods are inspired by the volume rendering equation [30], they only provide a rather rough approximation of its solution. In addition, the accuracy of the subsurface scattering is limited by the resolution of the volume or the grid. Some of the more advanced methods [3,13] also propagate light using low-order spherical harmonics that effectively diffuse the subsurface scattering contribution. Other techniques, which are based on separable filtering and a depth map, also achieve real-time subsurface scattering by aiming at the qualitative appearance and sacrificing quantitative accuracy [19,18].

Fast filtering techniques can be constructed so that they approximate diffuse subsurface scattering more accurately [12,5,21,29]. The filtering is done in texture space and thus requires texture parametrization of the object surface. To avoid texture space problems, similar filtering techniques are available for light space [9] and screen space [35,27,28,29]. The performance of all these filtering techniques, however, depends heavily on the assumption that the subsurface scattering is diffuse so that the convolution kernel is only a function of the distance between the points of incidence and emergence. Our work uses light space sampling [9], but removes the assumption that subsurface scattering is diffuse. If we were to remove this assumption from texture or screen space filtering techniques and adapt them for directional subsurface scattering, they would become texture space or screen space variations of the technique that we propose. The former variation would require texture parametrization of the object surface, the latter would be view-dependent.

Another interesting approach to interactive rendering of deformable translucent objects is based on splatting [41,6]. In this approach, surface points seen from the light source are splatted as screen-aligned quads. These splats contribute according to the subsurface scattering model where they overlap surface points in the geometry buffer of the camera. On first inspection, this seems an ideal approach for interactive rendering of directional subsurface scattering. However, the directional model requires larger splats as it varies not only with distance, and it is more expensive to evaluate as tabulation is impractical. We therefore found the splatting approach too expensive.

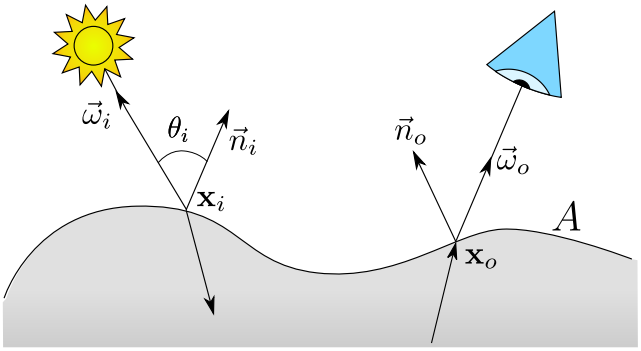

Fig. 2 BSSRDF configuration on an object surface A. The diagram illustrates the notation we use: bold font as in $\boldsymbol{x}_o$ denotes a point, while arrow overline as in $\vec{\omega}_i$ denotes a normalized direction vector.

3 Method

We render translucent objects using a *bidirectional scattering-surface reflectance distribution function* (BSSRDF). In most BSSRDFs, a translucent material is defined by the following spectral optical properties: refractive index η, absorption coefficient σ_a, scattering coefficient σ_s, and asymmetry parameter g. As is common in graphics, we use trichromatic optical properties (rgb). In addition, the BSSRDF depends on the position $\boldsymbol{x}_i$ and the direction $\vec{\omega}_i$ of the incident light as well as the position $\boldsymbol{x}_o$ and the direction $\vec{\omega}_o$ of the emergent light. The configuration is illustrated in Fig. 2. When rendering a translucent object, we obtain the outgoing radiance L_o by evaluating the following integral over all $\boldsymbol{x}_i$ in the surface area A and over all $\vec{\omega}_i$ in the hemisphere around the surface normal $\vec{n}_i$ at $\boldsymbol{x}_i$ [26]:

$$L_o(\boldsymbol{x}_o,\vec{\omega}_o) = L_e(\boldsymbol{x}_o,\vec{\omega}_o) + \int_A\int_{2\pi} S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o)L_i(\boldsymbol{x}_i,\vec{\omega}_i)\cos\theta_i\,\mathrm{d}\omega_i\,\mathrm{d}A_i\,, \quad (1)$$

where $\cos\theta_i = \vec{\omega}_i\cdot\vec{n}_i$, L_i is incident radiance, L_e is emitted radiance, and S is a BSSRDF. Disregarding surface reflection, as this can be incorporated using well-known techniques, the analytical BSSRDF can be written in the form:

$$S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o) = F_t(\vec{\omega}_o)(S_d(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o)+S^*)F_t(\vec{\omega}_i)\,, \quad (2)$$

where F_t is Fresnel transmittance, S_d is the diffusive part, which is typically modeled by a dipole, and S^* (dependencies omitted) is the remaining light transport, that is, the part not included with S_d.

As in other interactive subsurface scattering techniques that are not based on precomputation, we now

assume that S^* is insignificant. For most BSSRDF models [26,11,20], this means that single scattering is excluded entirely. However, if we use the directional dipole model [14], most single scattering is included with S_d. We therefore get a more accurate result with this model as the neglected S^* contains a significantly smaller part of the scattered light.

In existing interactive techniques, it is common practice to move the BSSRDF outside the integration over directions of incidence $\vec{\omega}_i$ (in Equation 1) and define transmitted irradiance by [33,9,36,35,5,41,6,34,29]

$$E(\boldsymbol{x}_i) = \int_{2\pi} L_i(\boldsymbol{x}_i, \vec{\omega}_i) F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \,. \tag{3}$$

We would however like to support BSSRDFs that include directional effects [20,14]. Since such BSSRDFs depend on $\vec{\omega}_i$, we cannot perform this separation, but we can define scattered radiosity by

$$B(\boldsymbol{x}_o) = \pi \int_A \int_{2\pi} S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) L_i(\boldsymbol{x}_i, \vec{\omega}_i) F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \, \mathrm{d}A_i \,. \tag{4}$$

This is an important quantity as the rendering equation (1) becomes

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = L_e(\boldsymbol{x}_o, \vec{\omega}_o) + \frac{1}{\pi} F_t(\vec{\omega}_o) B(\boldsymbol{x}_o) \,, \tag{5}$$

which enables view-independent rendering of translucent objects if we store scattered radiosity B. We note that L_o is not fully view independent because of the Fresnel term F_t, but this is an inexpensive term that we can evaluate per pixel per frame at very little cost.

For simplicity, our initial assumption is of a scene consisting of a single object illuminated by a single directional light. In Section 3.3, we extend to point lights, and in Section 4, we show an example of using multiple lights. For surface points lit by a directional light with radiance L_ℓ and direction $\vec{\omega}_\ell$, we have

$$L_i(\boldsymbol{x}_i, \vec{\omega}_i) = L_\ell \, V(\boldsymbol{x}_i, -\vec{\omega}_\ell) \, \delta(\vec{\omega}_i + \vec{\omega}_\ell) \,, \tag{6}$$

where V is visibility and δ is a Dirac delta function that makes the inner integral disappear, yielding

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_{A_{\text{lit}}} S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, \mathrm{d}A_i \,, \tag{7}$$

where $\cos\theta_\ell = -\vec{\omega}_\ell \cdot \vec{n}_i$ and A_{lit} is the directly lit area of the surface (for unlit areas $L_i = V = 0$). Since we only need to integrate over the directly lit part of the surface area, we perform the integration in a geometry buffer (G-buffer) rendered from the point of view of the light source (a translucent shadow map [9]). Since we have a directional light, our G-buffer is an orthographic projection of the scene into the light's view plane, which has $\vec{\omega}_\ell$ as its normal.

In order to distribute samples in the G-buffer according to a distance r and an angle α, we assume a planar surface normal to the light direction and rewrite the integral in polar coordinates with origin $\boldsymbol{x}_o$:

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_0^{2\pi} \int_0^\infty S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r \, \mathrm{d}r \, \mathrm{d}\alpha \,, \tag{8}$$

where $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and α is the angle between $\boldsymbol{x}_o - \boldsymbol{x}_i$ and the first basis vector of the light's view plane. This assumption is clearly often violated, but it is commonly used in derivation of BSSRDF models [26,11].

We evaluate the integral in Equation 8 by Monte Carlo integration. Our estimator for scattered radiosity is

$$B_N(\boldsymbol{x}_o) = \frac{\pi L_\ell}{N} \sum_{j=1}^{N} \frac{S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r_j}{p(r_j, \alpha_j)} \,, \tag{9}$$

where $p(r, \alpha)$ is the joint probability density function from which we draw the sample pairs (r_j, α_j). Starting from $\boldsymbol{x}_o$ transformed to the texture space of the light's camera, each sample pair corresponds to a texture space offset for looking up $\boldsymbol{x}_i$ and $\vec{n}_i$ in the light's G-buffer.

3.1 Sampling Distribution

BSSRDFs decay exponentially with the distance $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$. In particular, the asymptotic exponential falloff of the standard and directional dipoles [26,14] is $\exp(-\sigma_{\text{tr}} d)$, where $d \to r$ for $r \to \infty$ and σ_{tr} is the *effective transport coefficient* defined by

$$\sigma_{\text{tr}} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)} \,. \tag{10}$$

It is therefore highly beneficial to importance sample according to this exponential decay. We do importance sampling by choosing

$$p_{\text{exp}}(r, \alpha) = p(r)p(\alpha) = \sigma_{\text{tr}} e^{-\sigma_{\text{tr}} r} \frac{1}{2\pi} \,, \tag{11}$$

which is easily sampled by

$$(r_j, \alpha_j) = \left(\frac{-\log \xi_1}{\sigma_{\text{tr}}}, 2\pi\xi_2 \right) . \tag{12}$$

The symbols $\xi_1, \xi_2 \in [0, 1]$ denote canonical uniform random variables, which we obtain on the fly using a linear congruential pseudorandom number generator.

It is important to note that the effective transport coefficient σ_{tr} is different for different color bands. As a consequence, we use a separate set of position samples for each color band. In this way, we avoid color shifts, especially for materials with very different scattering coefficients in the different color bands (ketchup, for example).

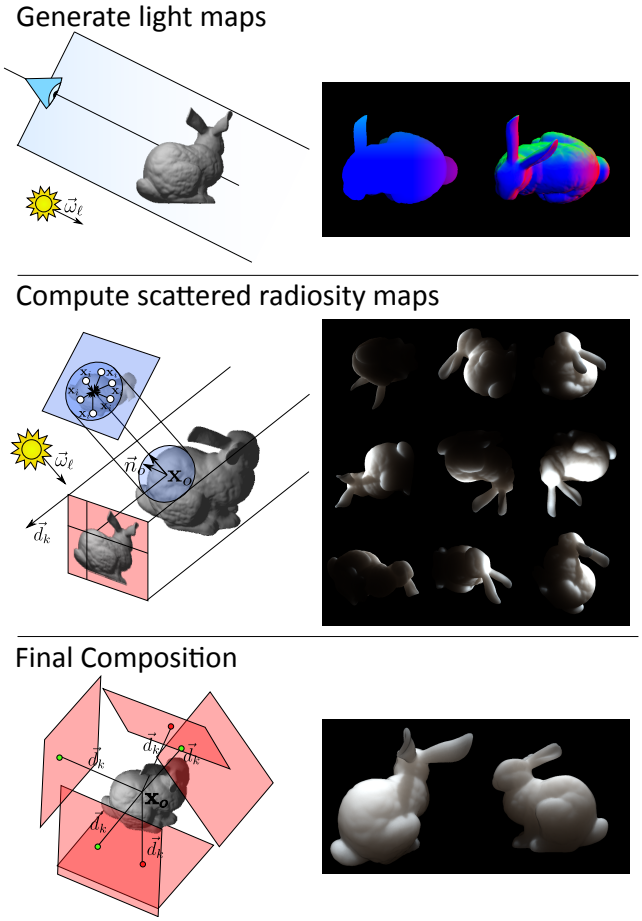

Fig. 3 Our three-step multipass technique for interactive rendering of directional subsurface scattering in deformable translucent objects. The scattered radiosity maps enable view-independence and transport of emergent light.

3.2 Rendering Technique

The diffusive part of the standard dipole BSSRDF depends only on $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and is therefore easily tabulated and used at runtime at nearly no expense. In directional subsurface scattering, on the other hand, the diffusive part of the BSSRDF depends on both $\boldsymbol{x}_o$, $\vec{n}_o$, $\boldsymbol{x}_i$, $\vec{n}_i$, and $\vec{\omega}_i$. This means that it is impractical to tabulate it and thus expensive to evaluate it. To limit the number of times that we need to evaluate the BSSRDF at runtime, we chose to exploit the opportunity to have view-independence by storing scattered radiosity in maps. In fact, as we noted in Equation 5, the scattered radiosity does not depend on the view direction $\vec{\omega}_o$. With view-independence, it is convenient to also make the update of the scattered radiosity maps progressive. By doing so, the rendered result improves over time if we are only moving the camera. Our technique is easily made progressive by adding more samples for each frame. This means that we have two render modes: (a) converged translucency with real-time fly-through and (b) fully flexible translucency rendered at interactive frame rates.

Our rendering technique is based on the rasterization pipeline of the graphics processing unit (GPU). In fully flexible mode, we use the three-step multipass algorithm illustrated in Fig. 3. In the first step, we create a G-buffer for each light source. In the second step, we compute scattered radiosity maps using these light G-buffers. In the third step, we sample the scattered radiosity maps and combine the look-ups. If nothing changed except the camera position, we also accumulate radiosity map results with the ones from the previous frames. When convergence is reached, we switch to converged mode and perform the third step only. In the following, we provide the details of the three steps.

In the first step, as in translucent shadow mapping [9], we render a G-buffer from the point of view of the light. For each pixel, we store positions and normals, as well as a material index (for global illumination purposes, Section 3.4). Each directional light has an orthographic camera and an associated G-buffer stored in a layered 2D texture. We compute all the light G-buffers in a single rendering pass, where each triangle is fed to each layer of a 2D layered texture in a geometry shader.

In the second step, we render the translucent object from K directions using orthographic cameras. The number of directions is chosen so that the surface of the model is covered well. We place the cameras randomly on the bounding sphere of the object using a quasi-random Halton sequence [22]. We then configure the cameras to look at the center of the bounding sphere with a frustum that encapsulates the sphere. Also in this step, we use layered rendering in order to efficiently render scattered radiosity into the different maps in a single pass. For each fragment of the translucent object observed by an orthographic camera, we compute the scattered radiosity by generating N samples per color band on-the-go (Equation 12), looking up into the light G-buffers with those samples to get $\boldsymbol{x}_i$ and $\vec{n}_i$, and using those to evaluate Equation 9. To avoid pattern repetition artifacts, we choose a seed for the random points using the pixel index in the scattered radiosity map as well as the current map and frame numbers.

To progressively update the scattered radiosity maps, we first perform a depth-only pass and then we render the model with writing into the depth buffer disabled. During the second step of the algorithm (except when the light condition is changing or the object is deforming), blending is enabled to allow accumulation in the scattered radiosity maps. We also generate mipmaps for the scattered radiosity maps so that we have the oppor-

Algorithm 1: Estimating the scattered radiosity in $\boldsymbol{x}_o$ using K maps (step 3 of Fig. 3). Each map has a direction $\vec{d}_k$ and a world-to-texture conversion matrix $\boldsymbol{P}_k$. The variable F counts the number of accumulated frames, which is needed to average the blending in step 2 of Fig. 3.

Data: $\boldsymbol{x}_o$, ϵ_{bias}, ϵ_{comb}, F, K
Result: B
$n = 0$
color $= (0, 0, 0)$
for $k \in [0, K)$ **do**
  $\cos\theta = \text{clamp}(\vec{n}_o \cdot \vec{d}_k, 0, 1)$
  $\bar{\boldsymbol{x}}_o = \boldsymbol{x}_o - \epsilon_{\text{comb}}(\vec{n}_o - \cos\theta\ \vec{d}_k)$
  $\bar{\boldsymbol{x}}_{o,\text{tex}} = \boldsymbol{P}_k\ \bar{\boldsymbol{x}}_o$
  $v = \texttt{multisampleVisibilityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}},\ \epsilon_{\text{bias}})$
  color $=$ **color** $+ v \cdot \texttt{sampleRadiosityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}}, \epsilon_{\text{bias}})$
  $n = n + v$
end
$B = \frac{\textbf{color}}{Fn}$

tunity to apply a cheap high-pass filter that smoothes high frequency noise.

In the third and final pass, we sample the scattered radiosity maps for each fragment of the translucent object observed by the actual camera. This process is described in the pseudo-code in Algorithm 1. We average the contributions from the various directions with the visibility of the point as a binary weight. In the third step of Fig. 3, the green and the red dots represent the visible and not visible contributions from the point $\boldsymbol{x}_o$, respectively. Storing depth with the scattered radiosity maps, we use shadow mapping to obtain a visibility function. To avoid artifacts, we choose a constant shadow bias ϵ_{bias} for the visibility function. Moreover, to avoid errors when sampling close to the borders of a scattered radiosity map, we multi-sample the shadow map and introduce an additional bias ϵ_{comb} that translates the sample position towards the negative normal direction $-\vec{n}_o$. After composition of the scattered radiosity B, we obtain outgoing radiance from Equation 5 and perform tone mapping to finalize the result.

Considering the procedure described in this section, we can get a better understanding of the parameter N. The total number of Monte Carlo samples used for computing the outgoing radiance (L_o) in a surface point observed by the camera is $3N$ times K times the number of frames used for progressive updates. From the point of view of a surface point, N can thus be thought of as the number of samples per frame per map direction per color band.

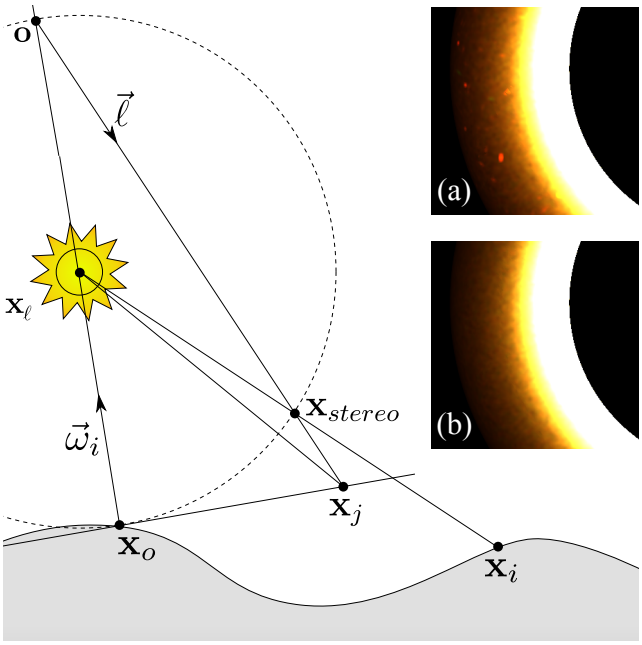

Fig. 4 Effect of stereographic correction when a translucent object surrounds a point light. With planar sampling (a), we look up into the light's cube map G-buffer using $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. With stereographic correction (b), we use $\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell$ instead. The insets (a and b) show how the correction improves the final result (torus, potato material).

3.3 Point Lighting

A point light at some distance from the translucent object works much in the same way as a directional light. The light's camera simply uses perspective instead of orthographic projection and intensity falls off with the distance squared. One particularly important application of our work is however simulation of the light coming through candles, candleholders, and lamp shades (Fig. 1, for example). In these cases, the point light is surrounded by the translucent object and we then use omnidirectional shadow mapping [15] with a cube map G-buffer for the light.

With a cube map captured for a point light at $\boldsymbol{x}_\ell$, one would first get a sampled point $\boldsymbol{x}_j$ by using (r_j, α_j) to offset $\boldsymbol{x}_o$ in its tangent plane. A look-up into the cube map with $\boldsymbol{x}_j - \boldsymbol{x}_\ell$ would then provide the sampled $\boldsymbol{x}_i$ and $\vec{n}_i$. However, when observing a translucent object surrounding the light source, this planar sampling of the light's G-buffer is no longer a good approximation. To have a better approximation that enables sampling of the entire cube map for each $\boldsymbol{x}_o$ (instead of only a hemisphere), we use an inverse stereoscopic projection. With this stereoscopic correction, the direction used for look-up into the cube map becomes

$$\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell = (\boldsymbol{x}_\ell - \boldsymbol{x}_o) - 2\left[(\boldsymbol{x}_\ell - \boldsymbol{x}_o) \cdot \vec{\ell}\right] \vec{\ell}, \qquad (13)$$

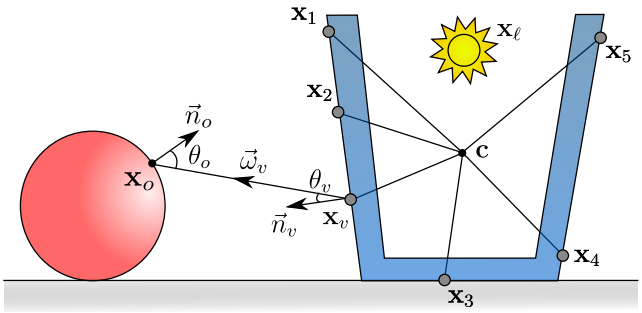

Fig. 5 Transport of emergent light from a translucent object (blue) to a diffuse object (red). We distribute VPLs (gray dots) on the outer surface of the translucent object, and use them to indirectly illuminate the remaining scene.

where

$$\vec{\ell} = \frac{(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)}{\|(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)\|}\,, \tag{14}$$

as illustrated in Fig. 4. The top right image (a) in Fig. 4 is an example of the sampling noise we get if we use $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. The middle right image (b) shows how the stereoscopic correction betters this problem.

3.4 Transport of Emergent Light

We further extend our method to account for transport of emergent light using virtual point lights (VPLs) [31]. We distribute a set of N_{vpl} points on the surface of the translucent object. Then, for each observed point $\boldsymbol{x}_o$, we add the contribution from all VPLs using

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = \sum_{v=1}^{N_{\text{vpl}}} f_r(\boldsymbol{x}_o, -\vec{\omega}_v, \vec{\omega}_o) \; G_b(\boldsymbol{x}_o, \boldsymbol{x}_v) V(\boldsymbol{x}_o, \boldsymbol{x}_v) I_v \tag{15}$$

with VPL intensity

$$I_v = \frac{1}{\pi} F_t(\omega_v) B(\boldsymbol{x}_v) A / N_{\text{vpl}}\,, \tag{16}$$

where A is the surface area across which the VPLs were distributed, G_b is the standard bounded geometry term [8], and B is obtained from the scattered radiosity maps using Algorithm 1.

As in the previous section, we now take special steps to accommodate our key use case of a point light surrounded by a translucent material. Our approach is illustrated in Fig. 5. In this particular case, the scene illuminated by emergent light will most commonly be shadowed from surface points of the translucent object that are directly lit (as the source is surrounded). We therefore approximate the visibility term V by distributing VPLs on backlit surfaces only. With this distribution of VPLs, we use the area of the bounding volume of the translucent object as an approximation of A. This is computed for each frame on the CPU.

Unfortunately, for a deformable object and a relatively small set of VPLs, the method is prone to flickering unless we ensure that the VPL positions are stable over time. Our solution is to render the outermost surface of the translucent object to a cube map whose center $\boldsymbol{c}$ coincides with the object's bounding box center. Each pixel in the cube map now contains the coordinates of a point on the surface of the translucent object. By sampling the cube map at a constant set of random directions, we obtain a stable set of surface positions that we use as VPL locations (Fig. 5).

4 Results

The implementation of our method interactively renders directional subsurface scattering in deformable objects and requires no preprocessing nor texture parameterization of the object surface. We use the diffusive part of the directional dipole [14] as S_d or the photon beam diffusion model [20] when evaluating Equation 9. The directional dipole is significantly faster, so we use this one unless noted otherwise. We define the translucent objects in our scenes using measured optical properties from different sources [26,37,17].

To validate our results, we compare with Monte Carlo ray tracing implemented on the GPU using OptiX [38]. In this reference method, we render directional subsurface scattering using the progressive direct Monte Carlo integration technique described by Frisvad et al. [14]. As prescribed, we use a Russian roulette based on the asymptotic exponential falloff of the model to accept or reject samples. However, we do not equidistribute the samples using a dart throwing technique as a more brute force uniform sampling of the object surface is more well-suited for a GPU ray tracer. This implementation gave us a ground truth for comparison both in terms of quality and performance. However, when comparing performance, one should keep in mind that unlike our method the reference method is view dependent.

In all the following examples, performance is at interactive rates. If nothing changes except the camera, our method will converge over a number of frames and then run in real-time. The implementation switches back to interactive rates when something other than the camera changes. By 'interactive' we mean a rendering time below 166 milliseconds per frame (6 frames per second, fps), as specified by Akenine-Möller et al. [1]. All the tests were performed on an NVIDIA GeForce GTX 780 Ti graphics card (2880 cores). Unless otherwise indi-

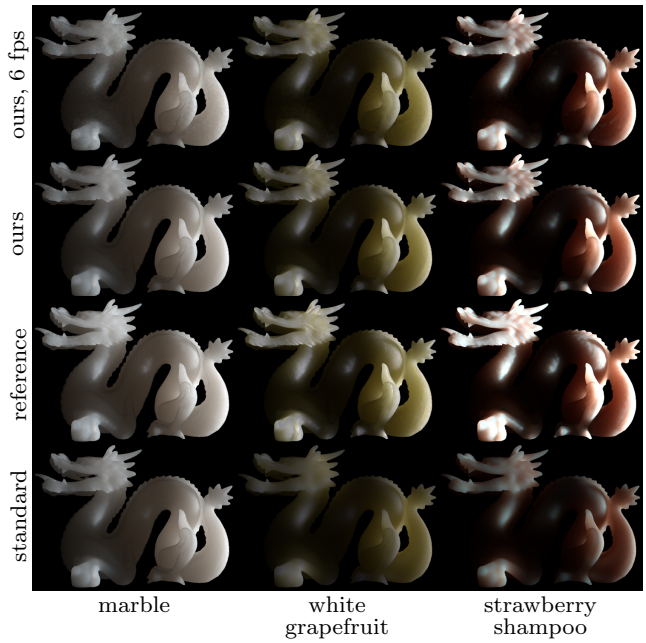

Fig. 6 Comparison of our method (rows 1 and 2) with the reference method (row 3) and diffuse subsurface scattering (row 4) for different materials. Row 1 is our results for a single frame at 6 fps, while row 2 is our view-independent result after convergence. All results use 31 maps.

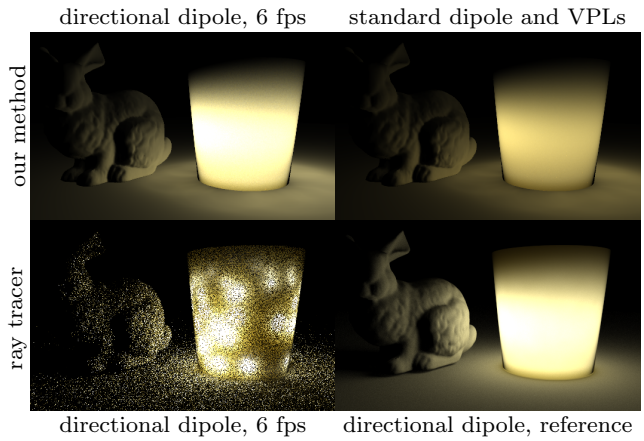

Fig. 7 Equal time comparison (left column) of our method with the reference method and qualitative comparison with diffuse subsurface scattering (upper right) and the converged reference solution (lower right). The scene is lit by a point light in a white grapefruit candle holder.

cated, our results use a 512×512 frame resolution for both radiosity and light maps.

Fig. 6 allows a visual comparison with ground truth (results obtained with the reference method). We chose one highly scattering material with isotropic phase function ($g = 0$), namely marble, and two forward scattering materials ($g > 0$), namely white grapefruit juice and strawberry shampoo. At convergence (second row), our method compares favorably to the directional dipole reference (third row). Our method improves the details

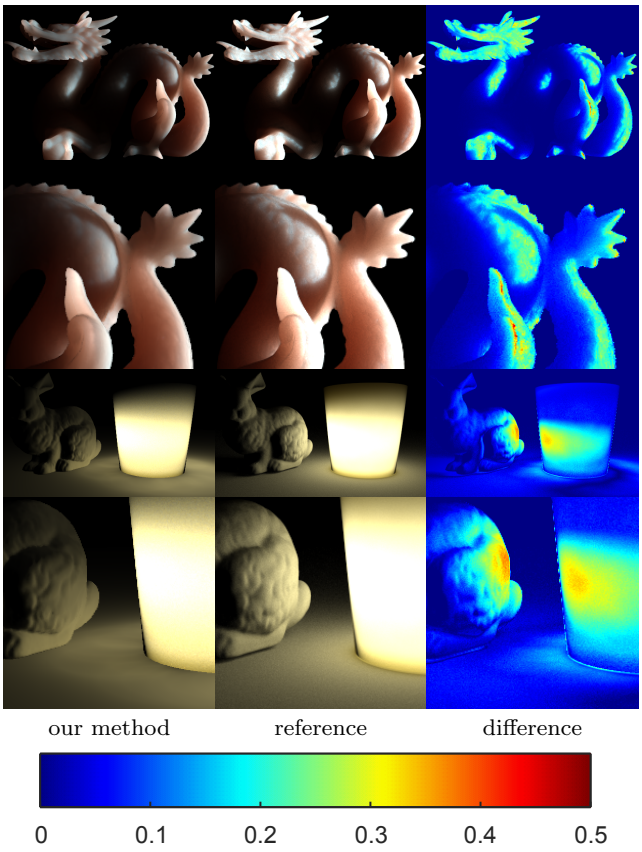

Fig. 8 Zoom-ins and differences from Figures 6 and 7. Root-mean-squared error of the color bands $\sqrt{\Delta r^2 + \Delta g^2 + \Delta b^2}$ is used as error metric in the difference images.

of the subsurface scattering when compared with diffuse subsurface scattering, that is, the standard dipole [26] (fourth row), especially for white grapefruit juice and strawberry shampoo. We also show the results of our method after one frame rendered at interactive frame rates (first row). These results are similar to our converged solution except that there is a slight bit of sampling noise, which we reduce using mipmap filtering.

Fig. 7 compares the transport of emergent light obtained with our method to that obtained with the reference method. While the 200 VPLs used here do not provide a highly accurate result, they do provide something better than a constant ambient term. At 6 fps, our solution is similar to the reference and converges very quickly to a better result, while the OptiX solution has both high-frequency and low frequency noise, is view dependent, and converges very slowly.

Fig. 8 provides zoom-ins and difference images from Figs. 6 and 7. Our results in general seem to be missing a part of the light transport. As revealed by the difference images, the missing contribution is due to undersampling of the surface at grazing incidence and missing interreflections. This undersampling is the reason why

our method ray tracer reference

Fig. 9 Stanford bunny with marble material at different scales (from top to bottom the scale is: 0.01, 0.1, and 1 meter). The left and middle columns show equal time results for our method and the ray tracer (1 frame at 6 fps). The right column shows the ray traced results after convergence. Here we use 16 maps and a 1024×1024 light map.

Mertens et al. [35] chose to sample in screen space instead of light space. However, sampling in screen space has other problems, as not all samples are lit. When considering transport of emergent light, the zoom-ins and difference images show missing shadows and inaccuracies due to the small number of VPLs. However, as graphics hardware improves, we will be able to use more VPLs and one of several fast VPL visibility techniques [8] to get better accuracy while retaining interactive frame rates.

In Fig. 9, we compare the quality reached by our solution with the quality reached by the ray traced solution in equal time. We perform this comparison for a marble bunny at three different scales. Generally, our method has a uniform behavior for different scales. For materials that are not optically thin (not at low scale), our method converges faster. The highly scattering materials (mid and high scale) are the more important cases to render well, as these are inside the range of materials for which the analytic subsurface scattering models are valid. At high scales, scattering effects become more localized, so our method is better at capturing the effect than the ray traced solution. At low scales, fewer G-buffer samples hit the object, which leads to a more noisy result with our solution.

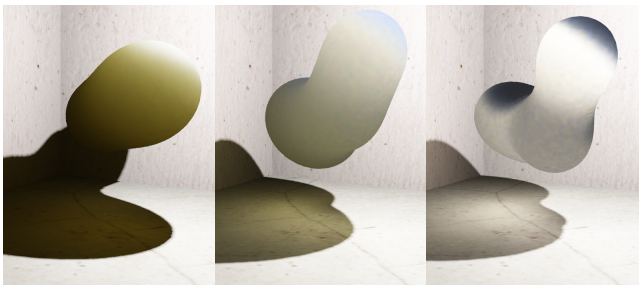

Fig. 10 Rendering with our method and a dynamically generated 3D surface ('blob') and transport of emergent light for three materials. The blob renders at 6 fps with 50 VPLs and 1500 samples per map in 6 maps. Materials from left to right: white grapefruit juice, soy milk, and glycerine soap.

In order to test the method using dynamically generated geometry, we created an implicit 3D surface [44] as the sum, $\Phi_t = \Sigma_i \phi_{i,t}(\vec{p})$, of 4 blobs,

$$\phi_{i,t}(\vec{p}) = \exp(-\sigma \|\vec{p} - \vec{p}_i(t)\|^2) \,,$$

where the position of each blob, $\vec{p}_i(t)$, is a periodic function. Since the periods are different, the period of the aggregate implicit Φ_t is potentially very large, and precomputation of the light transport inside the object would not be practical. Our method however applies, as it does not rely on precomputation, but we do need to rasterize the object. To do this, we compute a triangle mesh for an isosurface of Φ_t using dual contouring [16] implemented in a geometry shader. This is done in a pre-pass to each frame where the geometry shader evaluates Φ_t and its gradient directly based on the current time. The output triangle strips are streamed back to a vertex buffer object using transform feedback. Fig. 10 presents a rendered blob using different materials.

To justify our claimed need for scattered radiosity maps, we compare our method with an implementation without caching of subsurface scattering computations (as in translucent shadow mapping [9]). Note that this approach as opposed to ours is view dependent and pixel bound, and that unobserved VPLs would be more expensive to evaluate. Fig. 11 compares performance without considering view dependency and VPLs. Caching of scattered radiosity in maps is more efficient as soon as the translucent object occupies more than 5.8% of a 1024×1024 image.

The candle scene in Fig. 1 demonstrates the usefulness of our method. We scaled the optical properties of glycerine soap to approximate the scattering properties of candle wax. Our method creates a soft 'caustic' on the ground with varying intensity depending on the shape of the candle model. We thus enable a more realistic lighting of the scene than is obtainable with existing interactive techniques.

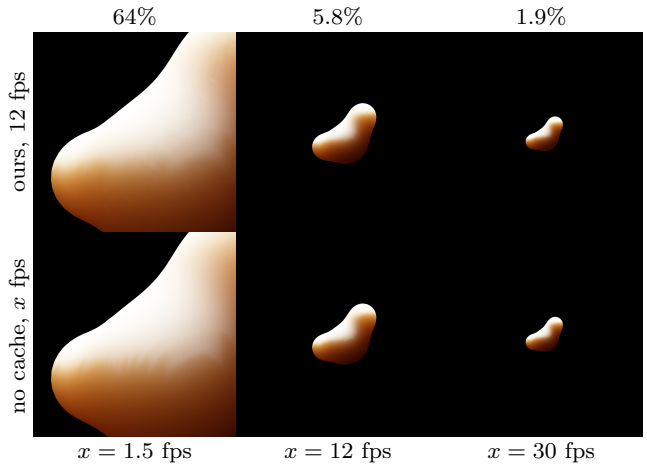

Fig. 11 Chocolate milk blob occupying different percentages of the image (noted at the top). We compare our method (ours) with a view-dependent, caching-free implementation (no cache, meaning no scattered radiosity maps). We use 1000 samples per map in 10 maps when caching, per pixel when not caching. Equal frame rates (12 fps) occur when occupancy is 5.8% of the image.

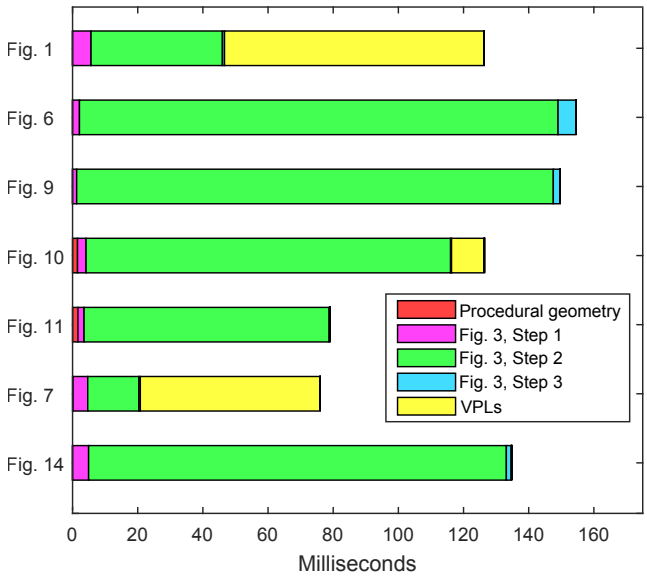

Fig. 12 Timing breakdowns for some of our renderings. Initialization times were negligible and were thus included with step 1 of Fig. 3. The evaluation of the BSSRDF and the VPLs (when present) dominate the rendering times.

To provide a performance breakdown of our technique, Fig. 12 lists render times dedicated to the different steps of our algorithm in our various results. BSSRDF evaluation (step 2 of Fig. 3) dominates all the timings, with the exception of Figs. 1 and 7, where the transport of emergent light dominates. Fig. 13 provides timings and coverage improvement of a bunny rendering with increasing K. The first seven directions cover most of the surface, while the remaining directions are necessary to cover small holes in the shading.

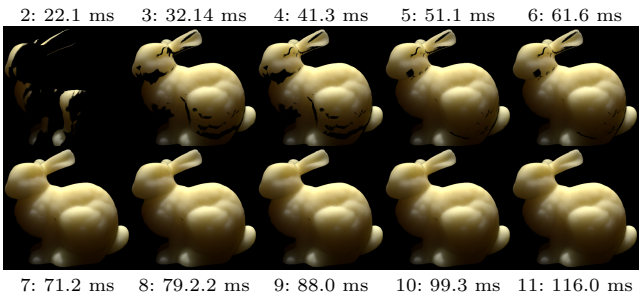

Fig. 13 Converged renderings of a potato bunny ($N = 30$) and timings for increasing number of scattered radiosity maps K. We list K followed by rendering time in milliseconds (ms) for each result. The first 7 maps cover most of the surface, while the following 4 cover small details (the small area just to the left of the bunny's hind leg, for example).

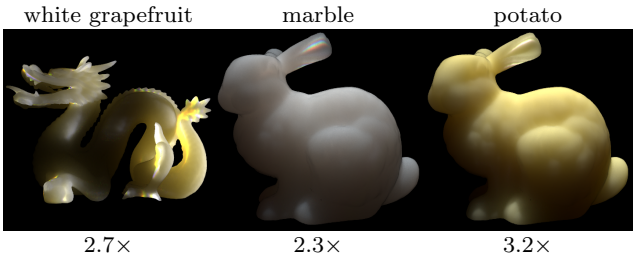

Fig. 14 Converged results with scenes and parameters as in other figures, but this time rendered using the photon beam diffusion model [20]. For each rendering, we provide the factor that this model is slower than if we use the directional dipole.

To underline the versatility of our approach, Fig. 14 has a set of results rendered using the photon beam diffusion model [20]. The weak singularities in this model lead to fireflies (overly bright pixels) with our sampling approach. We avoid this problem by clamping the distance d_r to a minimum of $0.25/(\sigma_a+\sigma_t)$ when it is used in a denominator. Factors that photon beam diffusion is slower than the directional dipole are included in the figure. These factors double if we use a graphics card with 512 cores (GTX 580) instead of 2880 cores.

Finally, Fig. 15 presents results with multiple directional lights. To approximate an environment light, we sample a number of representative directional light sources from the environment map using the method described by Pharr and Humphreys [39]. Contributions from all the directional lights are cached in the same scattered radiosity maps. In this example, we add specularly reflected light by looking up into the environment map using the direction of the reflected ray and multiplying by Fresnel reflectance.

5 Discussion

The resolution of a light's G-buffer (a light map) should be chosen carefully. If the range of the scattering ef-

Fig. 15 Stanford Bunny illuminated by an environment map. The map was importance sampled and converted to eight different directional lights. Potato material, 16 maps.

fects (roughly $1/\sigma_{\mathrm{tr}}$) is smaller than the size of one pixel in the light map, the contributions from the directional dipole tend to cluster and form 'pearling' artifacts. A possible solution would be a variation of cascaded shadow maps [51] to provide a higher resolution light map when needed. Generally, a light map of 512×512 pixels is an acceptable size that can be brought to 1024×1024 in problematic cases.

User parameters of our method include the resolutions of the light map and the scattered radiosity maps, the two biases ϵ_{comb} and ϵ_{bias}, the number of samples N, the number of scattered radiosity maps K, and the number of VPLs N_{vpl}. We now provide some guidelines for setting parameters. The size of the light map was already discussed in the previous paragraph. For the scattered radiosity maps, a size of 512×512 is fine for most application, and $K = 16$ directions generally provide enough coverage for simple models (the dragon, with its complicated geometry, required $K = 31$ directions). Performance scales linearly with K (Fig. 13), as we spend most of the time evaluating the BSSRDF (Fig. 12). The two biases ϵ_{comb} and ϵ_{bias} need to be tweaked manually. The numbers N and N_{vpl} are usually set manually to get the desired performance once the other parameters have been settled.

For most of our results, we choose the directions $\vec{d}_k$ of the scattered radiosity maps automatically. This works well for objects that are roughly convex, but for more oddly shaped concave objects some part may be left uncovered. Tearing artifacts caused by insufficient coverage appear in the mouth of the dragon in Fig. 6 and in the supplementary video. Fig. 13 also illustrates the problem, and shows that increasing the number of directions or manually choosing them can often ease this problem.

The memory consumption of our technique is comparable to that of the texture space filtering techniques [12,5,21,29]. As such, the maps and buffers that we use easily fit in the memory of modern GPUs. We surprisingly use more memory than the volumetric techniques [32,3,2,13]. The reason is that they make do with very low resolution volumes (32^3 or 64^3). It is however important to note that the added directionality and quality of details that we achieve cannot be achieved with such low resolution volumes. High resolution volumes would be needed with these techniques, which would lead to performance and memory issues.

Since we cache scattered radiosity, we cannot directly use a BSSRDF that fully depends on the direction of emergence $\vec{\omega}_o$ (the dual-beam model [10], for example). For such a BSSRDF, we would have to rely on the assumption that the emergent radiance integrates to a nearly diffuse distribution. We would then carry out a cosine-weighted integral over $\vec{\omega}_o$ when computing the scattered radiosity maps and otherwise use the same method. On the other hand, our concept of caching scattered radiosity instead of transmitted irradiance might be of interest in offline rendering techniques such as multiresolution radiosity caching [7]. This would enable use of directional subsurface scattering and inexpensive transport of emergent light in a movie production rendering solution.

6 Conclusion

We have presented a novel technique for interactive rendering of directional subsurface scattering. The method is view independent and applicable to deformable 3D models without requiring a texture parameterization of the object surface. While our method takes the direction of incident light into account, it also relies on the assumption that emergent light is not directional. This enables us to cache emergent light in so-called scattered radiosity maps. These maps enable us to control the output quality, to render progressively, and to illuminate the scene with light that has scattered through a translucent object.

Acknowledgements We would like to thank Christian Esbo Agergaard, Technical Director, Sunday Studios for the melting candle model. The Stanford Bunny and the Stanford Dragon models are courtesy of the Stanford University Computer Graphics Laboratory (http://graphics.stanford.edu/data/3Dscanrep/). The HDR environment map in Fig. 15 is courtesy of Tobias Grønbeck Andersen.

References

1. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-Time Rendering, 3rd edn. A K Peters (2008)
2. Bernabei, D., Hakke-Patil, A., Banterle, F., Benedetto, M.D., Ganovelli, F., Pattanaik, S., Scopigno, R.: A parallel architecture for interactively rendering scattering and

refraction effects. IEEE Computer Graphics and Applications **32**(2), 34–43 (2012)
3. Børlum, J., Christensen, B.B., Kjeldsen, T.K., Mikkelsen, P.T., Noe, K.Ø., Rimestad, J., Mosegaard, J.: SSLPV: Subsurface light propagation volumes. In: Proceedings of ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11), pp. 7–14 (2011)
4. Carr, N.A., Hall, J.D., Hart, J.C.: GPU algorithms for radiosity and subsurface scattering. In: Proceedings of Graphics Hardware 2003, pp. 51–59 (2003)
5. Chang, C.W., Lin, W.C., Ho, T.C., Huang, T.S., Chuang, J.H.: Real-time translucent rendering using GPU-based texture space importance sampling. Computer Graphics Forum (Proceedings of Eurographics 2008) **27**(2), 517–526 (2008)
6. Chen, G., Peers, P., Zhang, J., Tong, X.: Real-time rendering of deformable heterogeneous translucent objects using multiresolution splatting. The Visual Computer **28**(6–8), 701–711 (2012)
7. Christensen, P.H., Harker, G., Shade, J., Schubert, B., Batali, D.: Multiresolution radiosity caching for efficient preview and final quality global illumination in movies. Tech. Rep. Pixar Technical Memo #12-06, Pixar (2012)
8. Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., Novák, J.: Scalable realistic rendering with many-light methods. Computer Graphics Forum **33**(1), 88–104 (2014)
9. Dachsbacher, C., Stamminger, M.: Translucent shadow maps. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 197–201 (2003)
10. d'Eon, E.: A dual-beam 3D searchlight BSSRDF. In: ACM SIGGRAPH 2014 Talks, p. 65 (2014)
11. d'Eon, E., Irving, G.: A quantized-diffusion model for rendering translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011) **30**(4), 56:1–56:13 (2011)
12. d'Eon, E., Luebke, D., Enderton, E.: Efficient rendering of human skin. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2007), pp. 147–157 (2007)
13. Di Koa, M., Johan, H.: ESLPV: enhanced subsurface light propagation volumes. The Visual Computer **30**(6–8), 821–831 (2014)
14. Frisvad, J.R., Hachisuka, T., Kjeldsen, T.K.: Directional dipole model for subsurface scattering. ACM Transactions on Graphics **34**(1), 5:1–5:12 (2014)
15. Gerasimov, P.S.: Omnidirectional shadow mapping. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 12, pp. 193–203. Addison Wesley (2004)
16. Gibson, S.F.F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI'98, *Lecture Notes in Computer Science*, vol. 1496, pp. 888–898. Springer (1998)
17. Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., Levin, A.: Inverse volume rendering with material dictionaries. ACM Transactions on Graphics **32**(6), 162:1–162:13 (2013)
18. Gosselin, D.R., Sander, P.V., Mitchell, J.L.: Real-time texture-space skin rendering. In: W. Engel (ed.) ShaderX^3: Advanced Rendering with DirectX and OpenGL, chap. 2.8, pp. 171–184. Charles River Media (2004)
19. Green, S.: Real-time approximations to subsurface scattering. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 16, pp. 263–278. Addison Wesley (2004)
20. Habel, R., Christensen, P.H., Jarosz, W.: Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. Computer Graphics Forum (Proceedings of EGSR 2013) **32**(4), 27–37 (2013)
21. Hable, J., Borshakov, G., Heil, J.: Fast skin shading. In: W. Engel (ed.) ShaderX^7: Advanced Rendering Techniques, chap. 2.4, pp. 161–173. Charles River Media (2009)
22. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. Communications of the ACM **7**(12), 701–702 (1964)
23. Hao, X., Baby, T., Varshney, A.: Interactive subsurface scattering for translucent meshes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D 2003), pp. 75–82 (2003)
24. Hao, X., Varshney, A.: Real-time rendering of translucent meshes. ACM Transactions on Graphics **23**(2), 120–142 (2004)
25. Jensen, H.W., Buhler, J.: A rapid hierarchical rendering technique for translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) **21**(3), 576–581 (2002)
26. Jensen, H.W., Marschner, S.R., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: Proceedings of ACM SIGGRAPH 2001, pp. 511–518 (2001)
27. Jimenez, J., Sundstedt, V., Gutierrez, D.: Screen-space perceptual rendering of human skin. ACM Transactions on Applied Perception **6**(4), 23:1–23:15 (2009)
28. Jimenez, J., Whelan, D., Sundstedt, V., Gutierrez, D.: Real-time realistic skin translucency. IEEE Computer Graphics and Applications **30**(4), 32–41 (2010)
29. Jimenez, J., Zsolnai, K., Jarabo, A., Freude, C., Auzinger, T., Wu, X.C., von der Pahlen, J., Wimmer, M., Gutierrez, D.: Separable subsurface scattering. Computer Graphics Forum (2015). To appear
30. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. Computer Graphics (Proceedings of ACM SIGGRAPH 84) **18**(3), 165–174 (1984)
31. Keller, A.: Instant radiosity. In: Proceedings of ACM SIGGRAPH 97, pp. 49–56 (1997)
32. Kniss, J., Premože, S., Hansen, C., Ebert, D.: Interactive translucent volume rendering and procedural modeling. In: Proceedings of IEEE Visualization 2002, pp. 109–116 (2002)
33. 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 (PG 2002), pp. 214–224 (2002)
34. Li, D., Sun, X., Ren, Z., Lin, S., Tong, Y., Guo, B., Zhou, K.: TransCut: Interactive rendering of translucent cutouts. IEEE Transactions on Visualization and Computer Graphics **19**(3), 484–494 (2013)
35. Mertens, T., Kautz, J., Bekaert, P., Reeth, F.V., Seidel, H.P.: Efficient rendering of local subsurface scattering. In: Proceedings of Pacific Graphics (PG 2003), pp. 51–58 (2003)
36. Mertens, T., Kautz, J., Bekaert, P., Seidel, H.P., Reeth, F.V.: Interactive rendering of translucent deformable objects. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 130–140 (2003)
37. Narasimhan, S.G., Gupta, M., Donner, C., Ramamoorthi, R., Nayar, S.K., Jensen, H.W.: Acquiring scattering properties of participating media by dilution. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) **25**(3), 1003–1012 (2006)

$$2C_1 \approx \begin{cases} 0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 \\ \qquad + 4.98554\eta^4 - 1.36881\eta^5, \eta < 1 \\ -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 \\ \qquad - 2.54396\eta^4 + 0.254913\eta^5, \eta \geq 1 \end{cases} \quad 3C_2 \approx \begin{cases} 0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 \\ \qquad + 0.236494\eta^4 + 0.145787\eta^5, \eta < 1 \\ -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} + 1213.67\eta \\ -568.556\eta^2 + 164.798\eta^3 - 27.0181\eta^4 + 1.91826\eta^5, \eta \geq 1 \end{cases}$$

Fig. 16 Approximate fits for $2C_1$ and $3C_2$ by d'Eon and Irving [11].

38. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: OptiX: a general purpose ray tracing engine. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2010) **29**(4), 66:1–66:13 (2010)
39. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation, second edn. Morgan Kaufmann/Elsevier (2010)
40. Rushmeier, H.E., Torrance, K.E.: Extending the radiosity method to include specularly reflecting and translucent materials. ACM Transactions on Graphics **9**(1), 1–27 (1990)
41. Shah, M.A., Konttinen, J., Pattanaik, S.: Image-space subsurface scattering for interactive rendering of deformable translucent objects. IEEE Computer Graphics and Applications **29**(1), 66–78 (2009)
42. Sheng, Y., Shi, Y., Wang, L., Narasimhan, S.G.: A practical analytic model for the radiosity of translucent scenes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2013), pp. 63–70 (2013)
43. Sloan, P.P., Hall, J., Hart, J., Snyder, J.: Clustered principal components for precomputed radiance transfer. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) **22**(3), 382–391 (2003)
44. Velho, L., Gomes, J., de Figueiredo, L.H.: Implicit objects in computer graphics. Springer (2002)
45. Wang, J., Zhao, S., Tong, X., Lin, S., Lin, Z., Dong, Y., Guo, B., Shum, H.Y.: Modeling and rendering of heterogeneous translucent materials using the diffusion equation. ACM Transactions on Graphics **27**(1), 9:1–9:18 (2008)
46. Wang, R., Cheslack-Postava, E., Wang, R., Luebke, D., Chen, Q., Hua, W., Peng, Q., Bao, H.: Real-time editing and relighting of homogeneous translucent materials. The Visual Computer **24**(7), 565–575 (2008)
47. Wang, R., Tran, J., Luebke, D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) **24**(3), 1202–1207 (2005)
48. Wang, Y., Wang, J., Holzschuch, N., Subr, K., Yong, J.H., Guo, B.: Real-time rendering of heterogeneous translucent objects with arbitrary shapes. Computer Graphics Forum (Proceedings of Eurographics 2010) **29**(2), 497–506 (2010)
49. Xu, K., Gao, Y., Li, Y., Ju, T., Hu, S.M.: Real-time homogenous translucent material editing. Computer Graphics Forum **26**(3), 545–552 (2007)
50. Yan, L.Q., Zhou, Y., Xu, K., Wang, R.: Accurate translucent material rendering under spherical Gaussian lights. Computer Graphics Forum **31**(7), 2267–2276 (2012)
51. Zhang, F., Sun, H., Nyman, O.: Parallel-split shadow maps on programmable GPUs. In: GPU Gems 3, chap. 10. Addison-Wesley (2007)

A The Directional Dipole Model

[This appendix is not in the final publication.]

As other BSSRDF models, the directional dipole uses a number of inputs that are based on the optical properties of the translucent material (η, σ_s, σ_a, g):

$$\begin{array}{lll} \sigma_t = \sigma_s + \sigma_a\,, & \sigma'_s = (1-g)\sigma_s\,, & \sigma'_t = \sigma'_s + \sigma_a\,, \\ D = 1/(3\sigma'_t)\,, & d_e = 2.131\,D\sqrt{\sigma'_t/\sigma'_s}\,, & \sigma_{\mathrm{tr}} = \sqrt{\sigma_a/D}\,, \\ A = \frac{1-C_{\boldsymbol{E}}}{2C_\phi}\,, & C_\phi = \frac{1}{4}(1-2C_1)\,, & C_{\boldsymbol{E}} = \frac{1}{2}(1-3C_2)\,, \end{array}$$

where C_1 and C_2 are functions of η listed in Fig. 16. The directional dipole formulas for S_d are [14]:

$$S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) = S'_d(\boldsymbol{x}_o - \boldsymbol{x}_i, \vec{\omega}_{12}, d_r) - S'_d(\boldsymbol{x}_o - \boldsymbol{x}_v, \vec{\omega}_v, d_v)$$

and

$$\begin{aligned} &S'_d(\boldsymbol{x}, \vec{\omega}_{12}, r) = \\ &\tfrac{1}{4C_\phi(1/\eta)}\tfrac{1}{4\pi^2}\tfrac{e^{-\sigma_{\mathrm{tr}}r}}{r^3}\Bigg[C_\phi(\eta)\left(\tfrac{r^2}{D} + 3(1+\sigma_{\mathrm{tr}}r)\,\boldsymbol{x}\cdot\vec{\omega}_{12}\right) \\ &- C_{\boldsymbol{E}}(\eta)\Bigg(3D(1+\sigma_{\mathrm{tr}}r)\,\vec{\omega}_{12}\cdot\vec{n}_o \\ &- \left((1+\sigma_{\mathrm{tr}}r) + 3D\tfrac{3(1+\sigma_{\mathrm{tr}}r)+(\sigma_{\mathrm{tr}}r)^2}{r^2}\boldsymbol{x}\cdot\vec{\omega}_{12}\right)\boldsymbol{x}\cdot\vec{n}_o\Bigg)\Bigg], \end{aligned}$$

where the various S'_d arguments are based on positions ($\boldsymbol{x}_i$, $\boldsymbol{x}_o$), direction of incidence ($\vec{\omega}_i$), and normals ($\vec{n}_i$, $\vec{n}_o$). For the real source, we have

$$\begin{aligned} \vec{\omega}_{12} &= \frac{1}{\eta}((\vec{\omega}_i\cdot\vec{n}_i)\vec{n}_i - \vec{\omega}_i) - \vec{n}_i\sqrt{1 - \frac{1}{\eta^2}(1-(\vec{\omega}_i\cdot\vec{n}_i)^2)} \\ d_r^2 &= \begin{cases} |\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + D\mu_0(D\mu_0 - 2d_e\cos\beta), \text{ for } \mu_0 > 0 \\ |\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + 1/(3\sigma_t)^2 \qquad\qquad\quad \text{, otherwise} \end{cases} \\ \mu_0 &= -\vec{n}_o\cdot\vec{\omega}_{12} \\ \cos\beta &= -\sqrt{\frac{|\boldsymbol{x}_o - \boldsymbol{x}_i|^2 - (\boldsymbol{x}\cdot\vec{\omega}_{12})^2}{|\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + d_e^2}}\,, \end{aligned}$$

and for the virtual source,

$$\begin{aligned} \boldsymbol{x}_v &= \boldsymbol{x}_i + 2Ad_e\vec{n}_i^* \\ \vec{\omega}_v &= \vec{\omega}_{12} - 2(\vec{\omega}_{12}\cdot\vec{n}_i^*)\vec{n}_i^* \\ \vec{n}_i^* &= \begin{cases} \vec{n}_i & \text{, for } \boldsymbol{x}_o = \boldsymbol{x}_i \\ \dfrac{\boldsymbol{x}_o - \boldsymbol{x}_i}{|\boldsymbol{x}_o - \boldsymbol{x}_i|} \times \dfrac{\vec{n}_i \times (\boldsymbol{x}_o - \boldsymbol{x}_i)}{|\vec{n}_i \times (\boldsymbol{x}_o - \boldsymbol{x}_i)|} & \text{, otherwise}\,. \end{cases} \end{aligned}$$