HAL open science

Accurate computation of single scattering in refractive media

Nicolas Holzschuch

To cite this version:

Nicolas Holzschuch. Accurate computation of single scattering in refractive media. [Research Report] RR-8312, 2013, pp.16. hal-00829237v2

HAL Id: hal-00829237

https://inria.hal.science/hal-00829237v2

Submitted on 11 Jun 2013 (v2), last revised 25 Jun 2015 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Accurate computation of single scattering in refractive media

Nicolas Holzschuch

RESEARCH REPORT
N° 8312
juin 2013
Project-Team Maverick

ISSN 0249-6399 ISRN INRIA/RR--8312--FR+ENG

Accurate computation of single scattering in refractive media

Nicolas Holzschuch*

Project-Team Maverick

Research Report n° 8312 — juin 2013 — 16 pages

Abstract: Many materials exhibit sub-surface scattering: light enters them, is scattered inside and leaves the material in a different place. Most of the research on scattering effects has focused on multiple scattering, and used simple approximations for single scattering [JMLH01]. Recent research [WZHB09] has shown that single scattering produces highly detailed and complex effects, not captured by this approximation. Accurate computation of single scattering effects is computationally expensive. It requires many samples along each pixel ray, especially for finely tesselated meshes with interpolated normals. The existing algorithm for accurate computation of single scattering requires a position-normal tree along with many samples along the ray. In this paper, we present a new method for accurate computations of single scattering effects. Our method relies only on local information, available at the triangle level. It is faster and more accurate than the existing method [WZHB09].

Key-words: Illumination simulation, translucent materials, scattering, participating media, refraction

* INRIA Grenoble Rhône-Alpes and LJK (CNRS and Université de Grenoble)

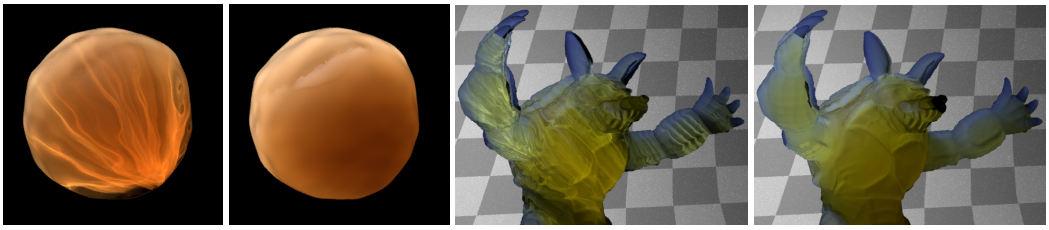
RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Calcul précis des événements de *single scatter* dans les matériaux réfractifs translucides

Résumé : Dans de nombreux matériaux, comme le marbre, la peau humaine, la pomme, le lait, la lumière qui les atteint pénètre à l'intérieur et interagit plusieurs fois avec l'intérieur du matériau. Cet effets donne à ces matériaux un aspect translucide, caractéristique. De nombreux travaux se sont intéressés à la modélisation de ce phénomène, mais ils se sont surtout concentrés sur ce qui se passe à la limite, lorsque la lumière a été dispersée un grand nombre de fois. Ces effets sont désormais bien connus. Comparativement, on connait peu de modèles pour le comportement de la lumière qui pénètre à l'intérieur du matériau et n'interagit qu'une seule fois avant de ressortir. Lorsque le matériau a un indice différent de 1, la lumière est réfractée à chaque interface, ce qui rend la localisation précise de ces événements difficile. La plupart des travaux antérieurs négligent l'une des réfractions, ce qui simplifie les calculs mais ne donne pas le bon résultat. Ce rapport décrit en détail une technique de calcul spécifique pour ce phénomène. Cette technique est plus rapide et plus précise que l'état de l'art. En outre, elle ne nécessite aucune structure de donnée annexe, ce qui la rend facile à implémenter au sein d'un logiciel de calcul de l'éclairage.

Mots-clés : Simulation de l'éclairage, milieux participants, dispersion, réfraction



(a) Our algorithm (6.6 mn) (b) Straight line (2.08 s) (c) Our algorithm (22.1 mn) (d) Straight line approximation (3.5 s)

Figure 1: Single scattering inside refractive translucent materials results in caustics (a and c). These caustics cannot be computed using the classical straight-line approximation [JMLH01] (b and d).

1 Introduction

Translucent materials, such as marble, amber or coffee, exhibit scattering effects: instead of just reflecting on the surface, incoming light enters the material, is scattered inside the material and leaves the object at a different point. These scattering effects result in smoother lighting. Traditionally, research on scattering effects separates between single and multiple scattering. Most of the existing research, including the dipole approximation [JMLH01] and Quantized Diffusion [DI11] have focused on multiple scattering. As it is mostly low frequency and diffuse it can be approximated using surface-based models.

There has been little research on single scattering. Existing implementations of scattering use a straight line approximation [JMLH01]. This approximation was developped for high-density materials, where the mean-free-path is small compared to the viewing distance and surface details. Although this approximation is not accurate for low density translucent materials, it is still widely used for lack of a viable alternative.

This lack of knowledge on single scattering has two consequences: first, accurate models of multiple scattering are combined with inaccurate representations of single scattering, resulting in lower combined accuracy. This impairs the comparison with ground truth computations and the validation of scattering models. Second, because we do not understand how single scattering behaves, it is difficult to develop fast but accurate algorithms.

To our knowledge, the only work to target single scattering effects in refractive media is Walter *et al.* [WZHB09]. Given a sample point inside the material, they compute all paths connecting this point to the light source and their contribution to single scattering. In this paper, we extend their algorithm: we focus on the part of the ray that intersects the translucent material. Instead of placing sample points on this ray, we directly compute the full contribution of each surface triangle. We quickly identify whether a triangle makes a contribution to the pixel, and on which segment of the ray. We sample only inside this segment. Our method gets more efficient for highly tesselated objects, as we can discard more triangles without sampling them. We are both faster and more accurate than the state-of-the-art method; as we do not require any extra data structure or pre-processing, our algorithm is easy to integrate into existing renderers.

Our paper is organized as follows: in the next section, we review previous work on scattering effects. In section 3, we present in detail our algorithm for computing single scattering effects. In the following section, we compare our results with previous works and give timings. Finally, we conclude in section 6.

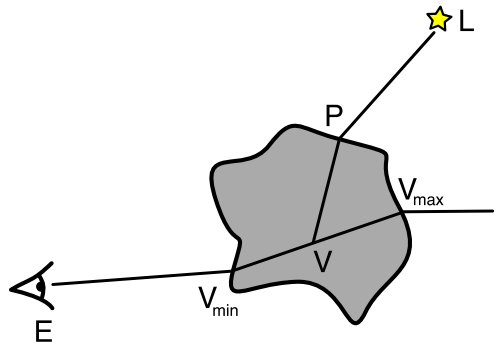


Figure 2: We consider a translucent object, under illumination from a light source L.

2 Previous works

Hanrahan and Krueger [HK93] computed scattering effects for materials with a small mean-free-path. They approximated single scattering with a BRDF, and used a full Monte-Carlo solution for multiple scattering. Jensen *et al.* [JMLH01] introduced the dipole approximation for multiple scattering, and a straight line approximation for single scattering effects. Single scattering events are connected to the light source by a straight line, ignoring refraction on that path.

Donner and Jensen [DJ05] extended this idea to multi-layered materials with the multipole model. D'Eon and Irving [DI11] introduced the Grosjean approximation for multiple scattering in isotropic participating media, and used it for accurate multiple scattering effects. Their quantized diffusion is an accurate representation of multiple scattering, even near the point of entry, especially for illumination orthogonal to the material surface. D'Eon [D'E12] simplified this approach for an improved dipole model. All these methods still relied on the straight line approximation for single scattering effects.

Habel *et al.* [HCJ13] used the Grosjean approximation in a beam-tracing framework, for accurate computations of multiple scattering, even with non-perpendicular illumination. They also introduced a new approximation for single-scattering effects, *diffuse single scatter*. They represent single scattering on the surface using a diffuse component.

These works make the hypothesis that the mean free path inside the material is relatively small, so that light undergoes multiple scattering events before leaving. Outgoing light is then diffuse and can be computed more efficiently. It also means that single scattering effects are less visible. Donner *et al.* [DLR*09] specifically targeted translucent materials a larger mean-free-path (compared to object size). They showed that in that case, the dipole approximation is not valid, and provided a more accurate BSSRDF model.

Walter *et al.* [WZHB09] specifically targeted single scattering effects in participating media. They showed that single scattering can result in highly visible caustics and provided an algorithm for computing these caustics. Our work builds on theirs and extends it.

Sun *et al.* [SRNN05] designed an efficient method for rendering non-refractive participating media. It does not generalize readily to the case of refractive media.

3 Single scattering computations

We consider a translucent object, under illumination from a light source L (see Figure 2). The object is made of a participating material, with a mean-free-path $\ell = \sigma_T^{-1}$ and albedo α. We assume that the material is refracting light rays at the surface of the object, with index of refraction η. Light is scattered

inside the material with a phase function ϕ.

We focus on single scattering effects. For each pixel, we consider the intersection between the eye-ray and the translucent object. Our goal is to compute the integral of all single scattering effects along this ray. We assume that the boundary of our translucent object is made of triangles with interpolated normals, a common representation in scene formats.

3.1 Sample point solution [WZHB09]

First, we briefly review the algorithm of Walter *et al.* [WZHB09]. They place sample point V along each pixel ray. For each sample point, they compute all paths connecting V to the light source L (see Section 3.1.1), then compute their contribution and sum them (see Section 3.1.2).

3.1.1 Connecting sample point V to the light source L

Given a sample point V, light source L and a triangle with interpolated normals, we are looking for all points on the triangle that connect V to L with a path that follows the law of refraction. The key idea is to transform this into the search for zeros of a function $\mathbf{f}$. Given a point P on the triangle, with shading normal $\mathbf{n}_S$, we define the normalized half-vector $\widehat{\mathbf{h}}$ and the function $\mathbf{f}$:

$$\mathbf{h} = \eta\omega_V + \omega_L \tag{1}$$

$$\widehat{\mathbf{h}} = \frac{\mathbf{h}}{\|\mathbf{h}\|} \tag{2}$$

where :

$$\omega_V = \overrightarrow{PV}/PV$$

$$\omega_L = \overrightarrow{PL}/PL\mathbf{f}(P) = \widehat{\mathbf{h}} + \mathbf{n}_S$$

All points P such that $\mathbf{f}(P) = \mathbf{0}$ correspond to a path connecting V and L, with the refraction angle following Snell's law.

To find the zeros of $\mathbf{f}$, Walter *et al.* [WZHB09] used a Newton-Raphson iterative method, with one small change: since point P on the triangle uses barycentric coordinates (α,β), $\mathbf{f}$ is a function from $\mathbf{R}^2$ into $\mathbf{R}^3$. The Jacobian J of $\mathbf{f}$ is a non-square matrix and therefore not invertible. Instead, they used the pseudo-inverse J^+:

$$J^+ = \left(J^T J\right)^{-1} J \tag{3}$$

With this pseudo-inverse the iterative Newton-Raphson method becomes:

$$P_{n+1} = P_n - J^+\mathbf{f}(P_n) \tag{4}$$

For this specific problem, the method converges quickly to a solution. We could use other iterative methods such as the conjugate gradient, but we didn't need to since, in our experiments, the function is well behaved and converges in few iterations. We mark this as a possible avenue for future work.

3.1.2 Contribution from each path

Once we have a path connecting V and L, we compute its contribution to the pixel using the following formula:

$$\text{contribution} = \frac{I_e F A \phi}{D} \tag{5}$$

where I_e is the intensity of the light source L, F is the Fresnel factor at both interfaces, A is the volume attenuation (the integral of $e^{-\sigma_T x}$ along both segments inside the media), ϕ is the phase function at V and D is the distance correction factor.

In the absence of participating media and refractive interface, D is the square of the distance between V and L, $D = LV^2 = (d_V + d_L)^2$. With refractive media and constant normal n_g, D has a simple expression:

$$D = (d_V + \eta d_L)\left(\frac{|\omega_L \cdot \mathbf{n}_g|}{|\omega_V \cdot \mathbf{n}_g|} d_V + \frac{|\omega_V \cdot \mathbf{n}_g|}{|\omega_L \cdot \mathbf{n}_g|} d_L\right) \tag{6}$$

Where $d_V = PV$ and $d_L = PL$. With interpolated shading normals, D has a more complicated expression, based on ray differentials at the interface [Ige99]. We take two vectors perpendicular to ω_V and each other: $\mathbf{u}_\perp$ and $\mathbf{u}_\parallel$. We then compute how L changes if we perturb ω_V along these vectors. D is the cross-product of these derivatives:

$$D = \left\| \frac{\mathrm{d}L}{\mathrm{d}\mathbf{u}_\perp} \times \frac{\mathrm{d}L}{\mathrm{d}\mathbf{u}_\parallel} \right\| \tag{7}$$

A complete expression of D can be found in [WZHB09]. We refer the interested readers to this article for more information. The accurate geometric factor D plays an important role in illumination computations (see Figure 9).

3.1.3 Acceleration techniques

Although the Newton-Raphson method converges quickly for each sample point V and each triangle, it is still expensive to compute all possible paths through all triangles, for all sample points. Walter *et al.* [WZHB09] used several methods to prune the triangles used in the search:

- Sidedness agreement: V must be in the negative half-space of the triangle, and L in the positive half-space, for both the geometric normal $\mathbf{n}_g$ and the shading normal $\mathbf{n}_s$.

- Spindle test: the angle between the incoming and the refracted ray is between $\frac{\pi}{2} + \arcsin(1/\eta)$ and π. For a given segment $[VL]$, this restricts the potential solutions P to inside a surface of revolution around $[VL]$, called the *spindle*.

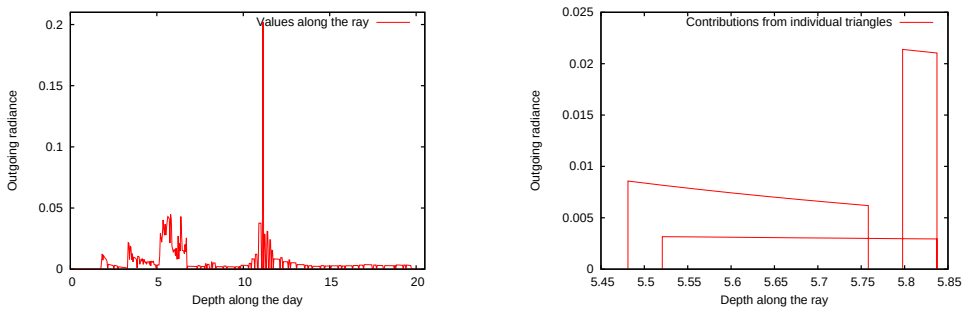
The spindle test is independent of the triangle normals, and can be used to prune large parts of the scene. For the sidedness agreement, Walter *et al.* [WZHB09] used a position-normal tree built from the triangles upward.

Together, these pruning methods reduced the computation time to something acceptable, a few minutes on a parallel, multi-core, cluster.

3.2 Our algorithm

3.2.1 Motivation: contributions along the ray

Although the algorithm converges quickly for objects made of flat triangles and moderately curved surfaces, Walter *et al.* [WZHB09] reported numerical stability issues for more complex scenes such as the bumpy sphere (see Figure 1a). They report having to use 128 samples along each pixel ray for this scene. To understand the reason, we plot the outgoing radiance as a function of depth along the ray (see Figure 3a for an example, corresponding to a single pixel). This function is highly irregular, with thin spikes. Thus the need many samples. If we separate the contributions from individual triangles, however, we see that each of them is regular, but with very small support (see Figure 3b). The irregular aspect of the overall response comes from the sum of many regular responses from individual triangles. The support of the response from any triangle is small, so regular samples along the segment are likely to miss it.



(a) Summed contributions from all triangles

(b) Individual contributions from 3 triangles

Figure 3: Outgoing illumination, measured along a pixel ray. The functions is highly irregular, requiring many samples for accurate integration. The contribution from each triangle is smooth, but with small support.

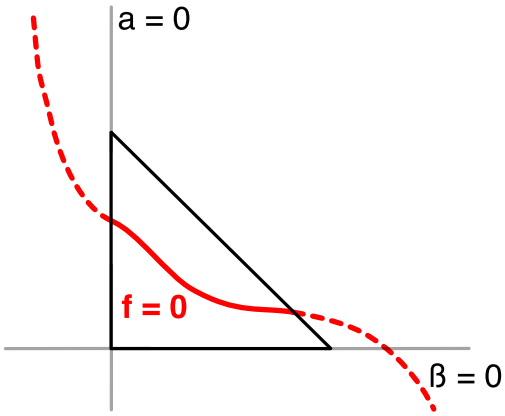


Figure 4: $\mathbf{f} = \mathbf{0}$ defines a curve in the triangle plane, parameterized by x. We search for intersections between this curve and the triangle (defined in barycentric coordinates).

3.2.2 Interval analysis along the ray

Our key idea is to start by identifying the limits of the contribution from each individual triangle, then sample only within these limits. This reduces the number of samples required along the ray.

To find these limits, we start with the same function $\mathbf{f}$, but with a third parameter, x, the coordinate of point V along the eye ray. x is already limited in range by the entry and exit points. $\mathbf{f}$ is now a function of three parameters: x and the barycentric coordinates (α, β) of point P on the triangle plane. In the general case, $\mathbf{f} = \mathbf{0}$ defines a polynomial surface in this 3D space.

We consider the plane of a single triangle, parameterized by its barycentric coordinates. $\mathbf{f} = \mathbf{0}$ defines a curve on this plane. We need the intersection between this curve and the triangle defined by: $\alpha > 0, \beta > 0, \alpha + \beta < 1$ (see Figure 4). The coordinates in x of these intersections will give us the support of function $\mathbf{f}$ along the ray.

For each triangle boundary, we find the intersection between the curve and this line. This becomes again a two-parameter problem, with x and one barycentric coordinate being the parameters, and the second barycentric coordinate being either fixed to 0 or linked to the first by $\alpha + \beta = 1$. We use the same technique as before for finding zeros of $\mathbf{f}$, with the pseudo-inverse. The only difference is that we

```
Start with interval [x_min, x_max]
numIntersections = 0;
repeat
    for each triangle boundary D
        Compute (x, α, β) intersection between D and curve
        Check if value of x is a min or a max (section 3.2.3)
        Update interval [x_min, x_max] accordingly
        If interval is empty: return 0
        If (α, β) inside triangle: numIntersections++;
    end for
until numIntersections = 2;
// Now, we sample the curve segment
Sum = 0
for x regularly sampled in [x_min, x_max]
     Find (α, β) such that f(x, α, β) = 0
     Sum += contribution from point (α, β)
end for
return Sum
```

Figure 5: Our algorithm for computing single-scattering effects

start with a 3×3 matrix, then convert it into a 3×2 matrix by removing one column, then compute the pseudo-inverse of this matrix.

Once we have an intersection, we compute the Jacobian of $\mathbf{f}$ at this point, to compute the local derivative of the curve (see section 3.2.3 for details). This tells us whether the value of x we found is an upper or a lower bound, and we update the interval for x accordingly. We treat all boundaries of the triangle, in a loop, until we either have an empty interval for x or a segment where the curve enters and leaves the triangle. If we have an empty interval, we move to the next triangle. If we have a valid segment, we sample it regularly and compute the contribution from each sample point.

Our algorithm is summarized in Figure 5. Most of the time, a single loop over the triangle edges is enough to find the entry and exit points. If the curve enters and leaves the triangle on the same edge, we need two loops over the triangle edges.

3.2.3 Derivatives of the curve at a sample point

We consider the camera position E and a ray from the camera $(E, \mathbf{u})$ where $\mathbf{u}$ is a unit vector.

We consider a point $P(\alpha, \beta)$ on the triangle, using barycentric coordinates. We take a sample point L on the light source. We define as ω_V and ω_L the unit vectors joining P and V (resp. L)):

$$\begin{aligned} \omega_V(x, \alpha, \beta) &= \overrightarrow{PV}/PV \\ \omega_L(\alpha, \beta) &= \overrightarrow{PL}/PL \end{aligned}$$

We define the normalized half-vector $\widehat{\mathbf{h}}$ and the function $\mathbf{f}(x, \alpha, \beta)$:

$$\begin{aligned} \mathbf{h}(x, \alpha, \beta) &= \eta\omega_V + \omega_L \\ \widehat{\mathbf{h}}(x, \alpha, \beta) &= \frac{\mathbf{h}}{||\mathbf{h}||} \end{aligned}$$

$$\mathbf{f}(x, \alpha, \beta) = \mathbf{h}(x, \alpha, \beta) + \mathbf{n}_S(\alpha, \beta)$$

For any normalized vector $\widehat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$, its derivative with respect to a parameter q is:

$$\frac{\partial \widehat{\mathbf{v}}}{\partial q} = \frac{1}{\|\mathbf{v}\|}\left(\frac{\partial \mathbf{v}}{\partial q} - \left(\widehat{\mathbf{v}} \cdot \frac{\partial \mathbf{v}}{\partial q}\right)\widehat{\mathbf{v}}\right) \tag{8}$$

Repeated applications of this rule give us the Jacobian of $\mathbf{f}$. We simplify part of the computation since ω_L and $\mathbf{n}_S$ do not depend on x.

For any parameter point (x, α, β), for any increment in parameter space $(\delta x, \delta\alpha, \delta\beta)$, we have:

$$\mathbf{f}(x + \delta x, \alpha + \delta\alpha, \beta + \delta\beta) \approx \mathbf{f}(x, \alpha, \beta) + J\begin{pmatrix} \delta x \\ \delta\alpha \\ \delta\beta \end{pmatrix} \tag{9}$$

If we consider a point (x, α, β) on the curve, we have $\mathbf{f}(x, \alpha, \beta) = \mathbf{0}$. Linearization in Equation 9 defines the tangent to the curve:

$$J\begin{pmatrix} \delta x \\ \delta\alpha \\ \delta\beta \end{pmatrix} = \mathbf{0} \tag{10}$$

The Jacobian J at a point on the curve is of rank at most 2. We can express one of the columns of J as a linear combination of the others:

$$\exists \lambda, \mu \text{ such that } \frac{\partial \mathbf{f}}{\partial x} = \lambda \frac{\partial \mathbf{f}}{\partial \alpha} + \mu \frac{\partial \mathbf{f}}{\partial \beta} \tag{11}$$

Our linear approximation of the curve (equation 10) becomes:

$$(\lambda\delta x + \delta\alpha)\frac{\partial \mathbf{f}}{\partial \alpha} + (\mu\delta x + \delta\beta)\frac{\partial \mathbf{f}}{\partial \beta} = \mathbf{0}$$

This gives us the following conditions:

$$\delta x = -\frac{1}{\lambda}\delta\alpha = -\frac{1}{\mu}\delta\beta \tag{12}$$

This equation tells us whether the point we found defines an upper or a lower bound on x, depending on the boundary of the triangle:

- For $\alpha = 0$, we have $\delta\alpha > 0$. x is a lower bound if $\lambda < 0$.
- For $\beta = 0$, we have $\delta\beta > 0$. x is a lower bound if $\mu < 0$.
- For $\alpha + \beta = 1$, we have $\delta\alpha + \delta\beta < 0$. Since $\delta\alpha + \delta\beta = -(\lambda + \mu)\delta x$, x is a lower bound if $\lambda + \mu > 0$.

3.2.4 Pruning triangles and restricting the interval

For each pixel ray, we have a segment $[V_{\min}, V_{\max}]$. Given a sample point L on the light source, we want to quickly test which triangles are potentially scattering light to any point on this segment.

We developped two strategies to restrict the search by pruning triangles:

- The first one is based on the spindle test and works on triangles and bounding volumes.
- The second is based on the shading normals and works only on triangles.

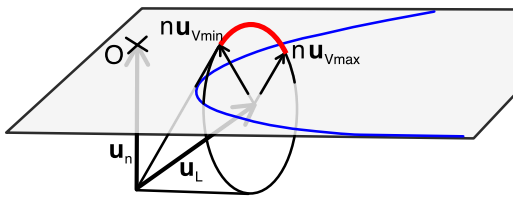


Figure 6: The half-vector $\widehat{\mathbf{h}}$ lives inside a cone of axis $\mathbf{u}_H$ and angle θ_H. $\mathbf{u}_H$ lives on an elliptical cone. This cone intersects the plane perpendicular to $\mathbf{u}_n$ in an hyperbola. Existence of a point such that $\mathbf{u}_n + \widehat{\mathbf{h}} = 0$ becomes a point-to-hyperbola distance in that plane.

We apply both strategies in a hierarchical descent of the scene hierarchy. Starting with the largest node containing the translucent object, we test whether it passes the first test. If yes, we iterate on its descendants. If not, we stop the search. We keep descending recursively until we reach triangles that have passed the first test. We then apply the second test to them, based on shading normals.

The output of the second test is both a boolean and a validity interval for x. If the boolean is true, we apply the extensive algorithm described in section 3.2.2.

For both tests, we start by building the bounding sphere of the object (whether it is a triangle or a bounding volume), with center C and radius r. We build bounding cones for ω_L and ω_V:

- ω_L is inside a cone of axis $\mathbf{u}_L$ and angle θ_L:

$$\begin{aligned} \mathbf{u}_L &= \overrightarrow{CL}/CL \\ \theta_L &= \arcsin(r/CL) \end{aligned}$$

- ω_V is bounded by a sweeping cone, whose normalized axis moves from $\mathbf{u}_{V\text{min}}$ to $\mathbf{u}_{V\text{max}}$ and whose angle is θ_V:

$$\begin{aligned} \mathbf{u}_{V\text{min}} &= \overrightarrow{CV_{\text{min}}}/CV_{\text{min}} \\ \mathbf{u}_{V\text{max}} &= \overrightarrow{CV_{\text{max}}}/CV_{\text{max}} \\ \theta_V &= \arcsin(r/d_{\text{min}}) \\ \text{where } d_{\text{min}} &= \min(\text{dist}(C, V)) \text{ for } V \in [V_{\text{min}}, V_{\text{max}}] \end{aligned}$$

Spindle-based test The spindle test tells us that the angle between ω_V and ω_L must be between $\frac{\pi}{2} + \arcsin(1/\eta)$ and π. Alternatively, the angle between ω_V and $-\omega_L$ must be smaller than $\frac{\pi}{2} - \arcsin(1/\eta)$.

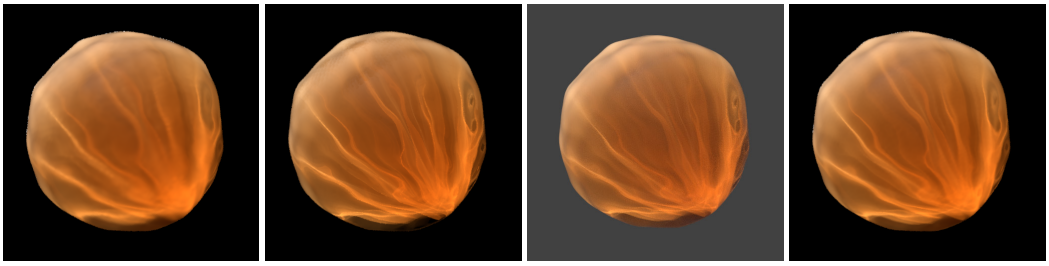
We compute the minimum angle between $-\mathbf{u}_L$ and $\mathbf{u}_V$ and compare it with $\frac{\pi}{2} - \arcsin(1/\eta) + \theta_V + \theta_L$. If it is larger, we can stop the search.

Shading-normal based test When testing against a triangle, we start by computing a sweeping cone bounding the half-vector $\widehat{\mathbf{h}}$. We also build a cone bounding the shading normal, with axis $\mathbf{u}_n$ and angle θ_n and check for intersection between the cones.

The extremity of the normalized vector ω_L lies inside a sphere of radius $r_L = 2\sin(\theta_L/2)$, centered on the extremity of $\mathbf{u}_L$ (similarly for ω_V). Thus, the extremity of $\mathbf{h} = \eta\omega_V + \omega_L$ is inside a sphere of radius $\eta r_V + r_L$, centered on a moving vector $\eta\mathbf{u}_V + \mathbf{u}_L$.

We note h_{min} the minimal value of the norm of $\eta\mathbf{u}_V + \mathbf{u}_L$. The normalized half-vector $\widehat{\mathbf{h}}$ lies inside a cone of axis $\mathbf{u}_H$ and angle θ_H:

$$\begin{aligned} \mathbf{u}_H &= \text{normalize}(\eta\mathbf{u}_V + \mathbf{u}_L) \\ \theta_H &= \arcsin\left((\eta r_V + r_L)/h_{\text{min}}\right) \end{aligned}$$



(a) Photon mapping (equal time, 2.2 M photons, 6.6 mn)

(b) Our algorithm (6.6 mn)

(c) Direct computation [WZHB09]

(d) Photon mapping (20 M photons, 48 mn)

Figure 7: Comparison between different methods for single scattering. (c) is reproduced from [WZHB09] with permission from the authors.

The axis $\mathbf{u}_H$ lies on a portion of an elliptic cone defined by $\mathbf{u}_L$ and the circle on which $\eta\mathbf{u}_V$ are (see Figure 6). It is possible to have $\widehat{\mathbf{h}} + \mathbf{n}_S = \mathbf{0}$ only if $-\mathbf{u}_H$ falls inside the cone of axis $\mathbf{u}_n$ and angle $\theta_n + \theta_H$.

Computing accurately the intersection of two cones is hard. Instead, we consider the plane perpendicular to $\mathbf{u}_n$, defined by $M \cdot \mathbf{u}_n = 1$. We place the origin of our coordinate system in that plane at the point $O = \mathbf{u}_n$. The intersection between this plane and the cone $(\mathbf{u}_n, \theta_n + \theta_H)$ is a circle, with center O and radius $r = \tan(\theta_n + \theta_H)$. The intersection between this plane and the elliptic cone carrying $\mathbf{u}_H$ is a hyperbola. The cones can intersect only if the curves intersect in the plane. Our problem (finding whether the two cones intersect) is transformed into computing the minimal distance between a point and a hyperbola. If that minimal distance is larger than r, we can discard the whole triangle for this segment.

We are actually interested in one of the two branches of the hyperbola, the one corresponding to $\mathbf{u}_n \cdot \mathbf{u}_H < 0$. We discard the other branch, which restricts the search interval for x. If O is outside the branch, we exploit the convexity of the hyperbola for faster rejection. Otherwise, we compute the minimal distance using dichotomy.

4 Results

Unless otherwise specified, all timings reported in this paper were recorded using our implementation of single scattering inside the Mitsuba renderer [Jak10] on an Intel Core2 (quad core) at 2.66 GHz.

4.1 Comparison with previous works

There has been comparatively few papers on single scattering effects, and they are difficult to compute even with existing global illumination methods [WZHB09]: bi-directional path tracing misses the single scattering effects with a point light source. Photon mapping requires a large number of volume photons and still provides a blurrier picture than direct computations.

Figure 1 provides a side-by-side comparison between our method and the straight line approximation [JMLH01] on two scenes (the bumpy sphere and an Armadillo model) The straight line approximation is two orders of magnitude faster than our method (and other methods), but does not provide all the caustics caused by single scattering.

Figure 7 provides a side-by-side comparison between our method, photon mapping and the Walter *et al* algorithm for single scattering [WZHB09]. We used the back lit bumpy sphere: it is a good test scene

for single scattering algorithms as it has several difficult points (thin caustics, many caustics converging on a single point, etc).

We used the implementation of photon mapping in the Mitsuba renderer [Jak10], which relies on the Beam Radiance Estimate by Jarosz *et al.* [JZJ08]. For roughly equal time computation (6.6 mn), photon mapping traces only 2.2 million photons, resulting in a blurry picture. With 20 millions photons, the single scattering effects are more clearly visible, but still blurrier than with our method. At that point, photon mapping is 7 times slower, and the memory cost is even larger. Note that we do not require any extra memory or data structure beyond what is already stored by the renderer: the geometry of the object, plus possibly a spatial hierarchical data structure.

Our results are almost identical to those of Walter *et al.* [WZHB09], but sharper. This gives our picture a better definition of the thin caustics. For a fair comparison of our algorithm with [WZHB09], we ran our algorithm on similar hardware. Walter *et al.* used a dual-processor with two Intel Xeon E5440 at 2.83 GHz. We used a less powerful dual-processor Intel Xeon E5345 at 2.33 GHz. Both machines have 8 cores in total. Our program took 261 s on the Bumpy Sphere scene, compared to 304 s for Walter *et al.* Our algorithm is thus at least 14 % faster for higher quality results.

4.2 Influence of the number of polygons

Name	# triangles	time (s)
Bumpy sphere	9680	396
Bunny (low res)	16301	568
Bunny (high res)	69451	1776
Armadillo	345944	2401

Table 1: Our test scenes: number of triangles and computation time recorded.

We ran our single scattering algorithm on several test scenes, from the bumpy sphere (10K triangles) to the armadillo model (350K triangles). For a given model, computation time is proportional to the number of pixels covered by the object on screen, and mostly indepedent of lighting conditions. Table 1 displays, for our test scenes, the number of triangles and computation time recorded.

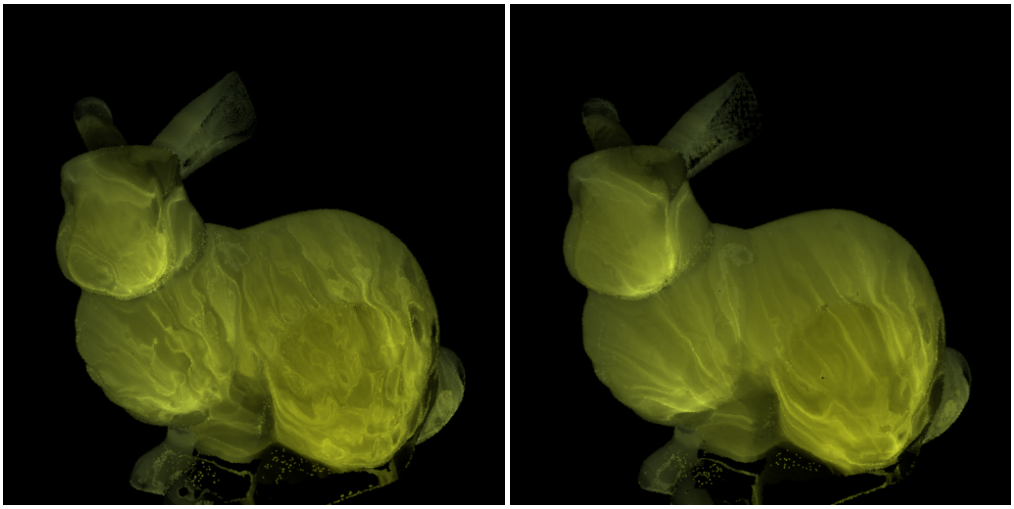
Figure 8 displays a comparison of our algorithm on two versions of the Stanford Bunny model: simplified (16K triangles) and high-resolution (69K triangles). Caustics are more complicated on the higher-resolution model.

4.3 Importance of the D factor

Figure 9 displays a side-by-side comparison of the bumpy sphere, computed using either the accurate D factor (described in section 3.1.2) or the approximate D factor based on geometric normals (Equation 6). Although both figures have computed the same paths through the triangles in the scene, the difference is strongly visible. The single scattering caustics are mainly due to light being concentrated by the curved surface.

4.4 Influence of the mean-free-path

Figure 10 displays a side-by-side comparison of single scattering on a translucent Stanford Bunny with varying scale. Changing the scale of the object is equivalent to changing the size of the object relative to the mean-free-path. Caustics caused by single-scattering only depend on geometrical factors. They do not move as we change scale, but their intensity varies.



(a) High-res. (69K tri, 22 mn). (b) Low-res. (16K tri, 7 mn).

Figure 8: Comparison between two versions of the bunny model, illuminated from behind.

For a fair comparison, we included transparency effects (how much of the background in visible through the object) and reflection. For objects of the same scale as the mean-free-path (scale = 2, left column), much of the light passes through the object and translucency effects are dominant. As we increase the object size relative to the mean-free-path (scale = 8 and 32), these effects become less noticeable and single-scattering effects are dominant. Finally, when the mean-free-path is much smaller than the object (scale = 128, right column), single scattering caustics become invisible because of absorption. For these materials, the straight line approximation provides a good approximation, although differences are still visible with our method.

4.5 Influence of accuracy parameters

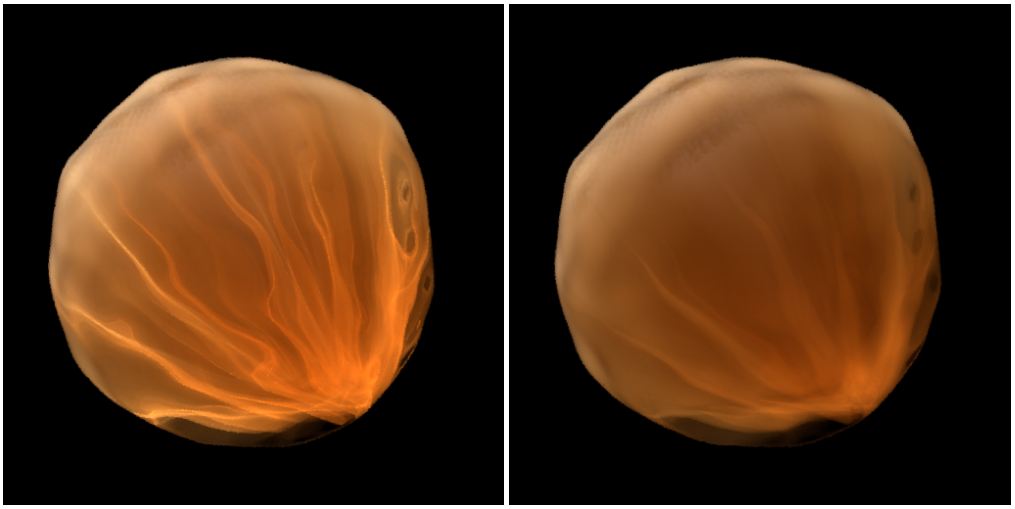
When we search for zeros of $\mathbf{f}$, we run our computations up to a certain accuracy. Typically, we stop the search if $\|\mathbf{f}\| < \varepsilon$. High values of ε (low accuracy) result in more noise, mainly through the D coefficient (see section 3.1.2). Since the search algorithm has quadratic convergence speed, reducing ε has little influence on the computation time (see Figure 11).

5 Acknowledgments

The Bunny and Armadillo models used in the figures of this paper are provided by the Stanford Computer Graphics Lab data repository. The Bumpy Sphere model used in the figures of this paper was kindly provided by Bruce Walter. This work was supported in part by a grant ANR-11-BS02-006 “ALTA”.

6 Conclusion

We have introduced a new algorithm for direct computation of single scattering effects in a translucent object, whose boundary is made of triangles with interpolated normals. Our algorithm is an extension of



(a) Full D factor
(based on shading normals).

(b) Approximate D
(no shading normals).

Figure 9: Importance of the D factor in single scattering computations. The approximate D factor fails to capture the thin caustics inside the material.

Walter *et al.* [WZHB09]. We restrict the influence of each boundary triangle to a small segment on the ray inside the object, then sample only inside this segment. Our algorithm is both faster and more accurate than the state-of-the-art [WZHB09]. It renders accurately the single-scattering caustics. Although we are slower than the straight line approximation classically used for fast rendering of scattering effects, we think that the added accuracy makes it worth using, especially for low density translucent materials. Our experiments show that single scattering effects are clearly visible when the mean-free-path is larger than a few percents of object size.

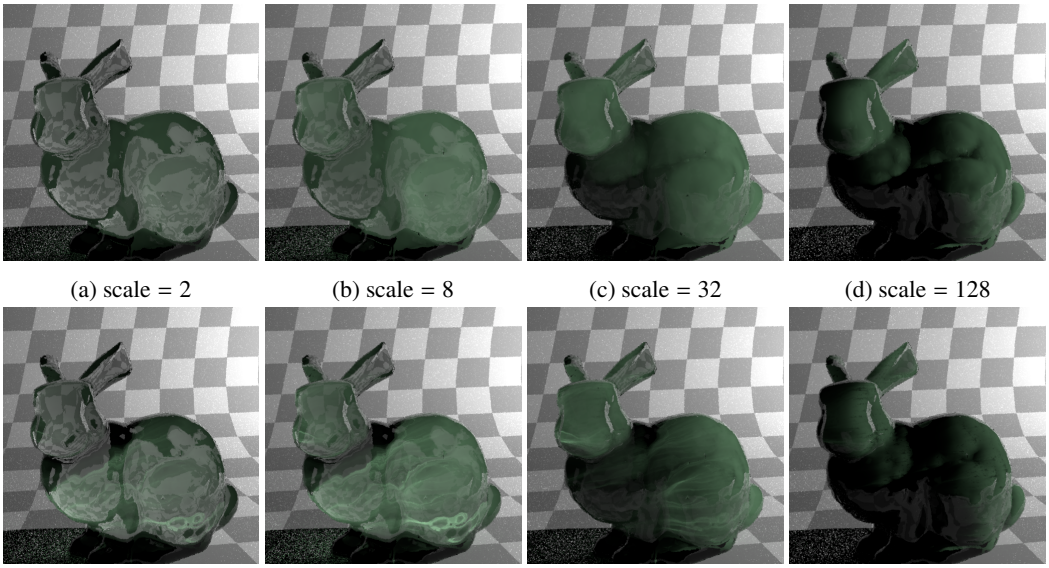
Our algorithm can be combined with any method for multiple scattering, such as the dipole or quantized diffusion. Since we do not require any data structure beyond what is usually present in a renderer, it is easy to integrate in any renderer. We hope that this algorithm will be useful to all researchers working on scattering effects. We are releasing the source code inside the Mitsuba renderer. We expect that this paper and implementation will help researchers in developing faster — but still accurate — computations of single scattering.

In future work, we want to exploit the added accuracy on single scattering for a better overall representation of all scattering effects, with both single and multiple scattering. We also want to use a hierarchical representation of surface details for faster computations. Finally, we want to port our code on the GPU for faster computations.

References

[D'E12] D'Eon E.: A better dipole. `http://www.eugenedeon.com/papers/betterdipole.pdf`, 2012.

[DI11] D'Eon E., Irving G.: A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph. 30*, 4 (July 2011), 56:1–56:14.



(a) scale = 2 (b) scale = 8 (c) scale = 32 (d) scale = 128

(e) scale = 2 (f) scale = 8 (g) scale = 32 (h) scale = 128

Figure 10: Influence of the scale parameter on single scattering effects. Top row: straight line approximation (10 s). Bottom row: our algorithm (9.46 mn).

[DJ05] Donner C., Jensen H. W.: Light diffusion in multi-layered translucent materials. *ACM Trans. Graph. 24*, 3 (July 2005).

[DLR*09] Donner C., Lawrence J., Ramamoorthi R., Hachisuka T., Jensen H. W., Nayar S.: An empirical bssrdf model. *ACM Trans. Graph. 28*, 3 (July 2009), 30:1–30:10.

[HCJ13] Habel R., Christensen P., Jarosz W.: Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Computer Graphics Forum 32*, 4 (2013).

[HK93] Hanrahan P., Krueger W.: Reflection from layered surfaces due to subsurface scattering. In *20th annual conference on Computer graphics and interactive techniques* (1993), SIGGRAPH '93, pp. 165–174.

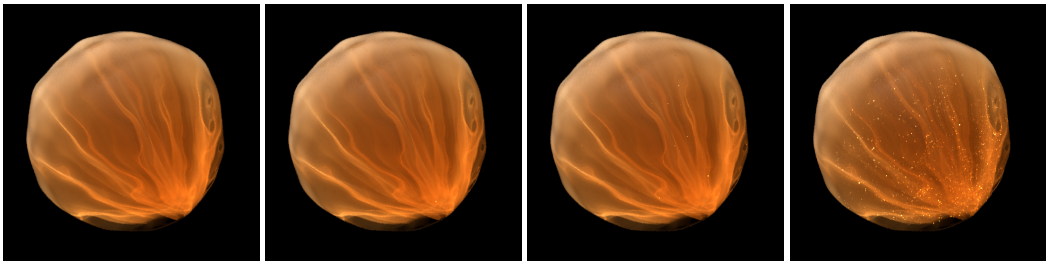
[Ige99] Igehy H.: Tracing ray differentials. In *26th annual conference on Computer graphics and interactive techniques* (1999), SIGGRAPH '99, pp. 179–186.

[Jak10] Jakob W.: Mitsuba renderer, 2010. `http://www.mitsuba-renderer.org`.

[JMLH01] Jensen H. W., Marschner S. R., Levoy M., Hanrahan P.: A practical model for subsurface light transport. In *28th annual conference on Computer graphics and interactive techniques* (2001), SIGGRAPH '01, pp. 511–518.

[JZJ08] Jarosz W., Zwicker M., Jensen H. W.: The beamradiance estimate for volumetric photon mapping. *Computer Graphics Forum 27*, 2 (Apr. 2008), 557–566.

[SRNN05] Sun B., Ramamoorthi R., Narasimhan S. G., Nayar S. K.: A practical analytic single scattering model for real time rendering. *ACM Trans. Graph. 24*, 3 (July 2005), 1040–1049.



(a) $\varepsilon = 10^{-5}$ (6.6 mn) (b) $\varepsilon = 10^{-4}$ (6.5 mn) (c) $\varepsilon = 10^{-3}$ (6.5 mn) (d) $\varepsilon = 10^{-2}$ (6.5 mn)

Figure 11: Reducing the accuracy of the computations increases the noise with little benefit on computation time.

[WZHB09] Walter B., Zhao S., Holzschuch N., Bala K.: Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph. 28*, 3 (July 2009), 92:1–92:8.

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399