HAL
open science

Single Scattering in Refractive Media with Triangle Mesh Boundaries

Bruce Walter, Shuang Zhao, Nicolas Holzschuch, Kavita Bala

To cite this version:

Bruce Walter, Shuang Zhao, Nicolas Holzschuch, Kavita Bala. Single Scattering in Refractive Media with Triangle Mesh Boundaries. ACM Transactions on Graphics, 2009, 28 (3), pp.92:1-8. 10.1145/1531326.1531398 . inria-00388481

HAL Id: inria-00388481

https://inria.hal.science/inria-00388481

Submitted on 26 May 2009

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.

Single Scattering in Refractive Media with Triangle Mesh Boundaries

Bruce Walter
Cornell University

Shuang Zhao
Cornell University

Nicolas Holzschuch
INRIA – LJK

Kavita Bala
Cornell University

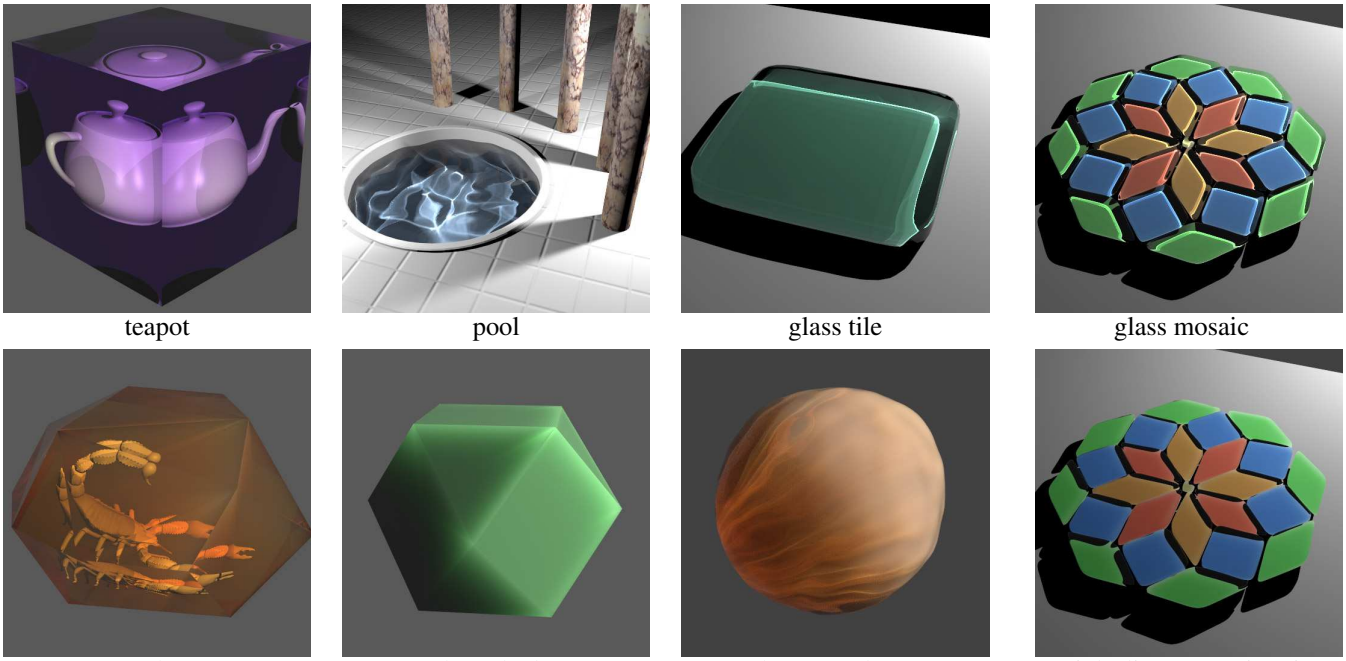

Figure 1: The bending and focusing of light in refractive media creates distinctive rich details. The top row shows single scatter surface caustics in glass and water. The bottom row shows complex volumetric refractive caustics in amber and glass. All images were generated using the method in this paper, except the bottom right which used the common straight-line approximation that neglects shadow ray refraction.

Abstract

Light scattering in refractive media is an important optical phenomenon for computer graphics. While recent research has focused on multiple scattering, there has been less work on accurate solutions for single or low-order scattering. Refraction through a complex boundary allows a single external source to be visible in multiple directions internally with different strengths; these are hard to find with existing techniques. This paper presents techniques to quickly find paths that connect points inside and outside a medium while obeying the laws of refraction. We introduce: a half-vector based formulation to support the most common geometric representation, triangles with interpolated normals; hierarchical pruning to scale to triangular meshes; and, both a solver with strong accuracy guarantees, and a faster method that is empirically accurate. A GPU version achieves interactive frame rates in several examples.

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

Keywords: Refraction, subsurface, single scatter

1 Introduction

Single (or low order) scattering is often an important effect in subsurface and volumetric materials such as amber or quartz and in optically thin materials. Unlike multiple scattering which generally acts as a blurring or low-pass filter, single scattering can produce high frequency effects such as volumetric caustics. Single scattering within a refractive medium is especially challenging because there may be zero, one, or many paths connecting a scattering location (inside the medium) and a light source (outside the medium). The refraction also can focus or defocus the light depending on the exact configuration. Figure 1 shows several complex examples.

The most common solution to this problem is to simply ignore it by directly connecting the light source and scattering point. This is cheap but inaccurate (see Figure 1, bottom right). Accurate approaches use random sampling, but this becomes arbitrarily expensive for small light sources or features. There is a lack of efficient, accurate solutions for computing these important types of paths.

In this paper we explicitly solve for refracted paths that connect a point inside and a point outside the medium. Our contributions are:

- Formulation of the problem based on the half-vector. This is necessary to support triangles with smooth shading normals, the most widely used geometry representation in graphics.
- Robust and efficient iterative techniques to find all such paths.
- Hierarchical pruning algorithms to scale to large meshes.
- Derivations and methods for computing the light contribution along a refracted path.

We will next briefly review previous work in Section 2, then formu-

late the problem and describe our path finding in Section 3. Section 4 describes how to compute the light contribution along each path. Finally, we present results in Section 5.

2 Previous Work

Our algorithm is related to previous work in several areas.

Subsurface scattering. Hanrahan and Krueger [1993] approximated single scattering by a BRDF, while using a full Monte-Carlo solution for multiple scattering. Jensen et al. [2001] simplified the multiple scattering case with their dipole approximation, but still use the BRDF approximation or use shadow rays that ignore refraction for single scattering.

Volume scattering. There has been a lot of work on fast volume scattering in non-refractive media such as Sun et al. [2005] and Nishita and Nakamae [1987], but this does not generalize easily to refractive media.

Accurate reflections from curved reflectors. Several researchers have used Fermat's principle to solve for reflection points on curved reflectors. Mitchell and Hanrahan [1992] used differential geometry and interval analysis to compute the exact reflection points. They discuss, but do not demonstrate, finding refraction points. Chen and Arvo [2000] compute a quick approximation of reflected rays using a Taylor expansion. Both these approaches are restricted to implicit surfaces with known equations.

Most models used in computer graphics use triangle meshes. Ofek and Rappoport [1998] computed the reflections of the vertices of such a triangular mesh. In recent work, similar methods have been used for interactive rendering, e.g. [Estalella et al. 2006; Roger and Holzschuch 2006; Szirmay-Kalos et al. 2005; Szécsi 2006]. However these methods find the reflection point by ray tracing and then solve for the approximate reflected point. This is a different problem than the one we are solving.

Beam tracing. The light refracted through the surface can be represented by a set of volumetric beams, but the beams have complicated shapes and intensity profiles. However fast caustic approximations can be constructed by sampling and interpolating intensity values in these beams, (e.g., [Nishita and Nakamae 1994; Iwasaki et al. 2003; Ernst et al. 2005]), though these methods are approximate and have trouble tracking shadowing of the caustics.

Monte Carlo. Caustics can, in principle, be computed by Monte Carlo path tracing, but the convergence tends to be very slow when shadow rays cannot be used, such as in refractive single scattering, especially if the light source is small. Metropolis Light transport [Veach and Guibas 1997] improves the Monte Carlo convergence rate by exploring perturbations about such paths once found, but still works best when the light source is large or shadow rays can be used.

Photon mapping. Photon mapping [Jensen 2001; Jarosz et al. 2008] can estimate the light at any point in space, but the result is blurred by the size of the kernel needed to reduce the noise. Thus very large numbers of photons can be necessary to reconstruct sharp features like caustics, especially with participating media. On the GPU, photon mapping can be done interactively [Sun et al. 2008] with impressive results for small scenes, but is still very data intensive and requires high photon counts.

Compared to our method, photon mapping handles a wider variety of lighting paths, with a cost that depends mostly on the finest feature resolution one is trying to reconstruct and the spatial extent of the scene, due to the view-independent nature of the photon map. Our method is purely view-dependent but handles only a specific class of light paths with cost that depends mainly on the medium boundary complexity and number of distinct convergent light paths.

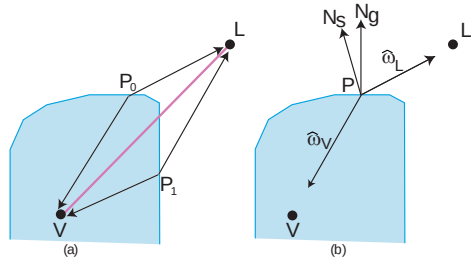

Figure 2: (a) Multiple light paths (black lines) can connect light L to scatter point V, while the non-refractive approximation (purple line) cannot account for these multiple paths. (b) Geometry of problem at a point **P** with geometric normal $\hat{\mathbf{N}}_g$ and shading normal $\hat{\mathbf{N}}_s$.

3 Finding Refracted Connecting Paths

Our goal is to compute the illumination at a point inside a refractive medium from a light source outside the medium. Specifically, we want to find all the ways that light from a point **L** outside the medium can reach a scattering point **V** inside the medium without scattering in between except for a single refraction event when it crosses into the refractive medium boundary (see Figure 2). We assume the index of refraction is piecewise constant with $\eta = \eta_{\text{in}}/\eta_{\text{out}}$. We also assume that the refraction obeys Snell's law except that it may bend relative to the local shading normal $\hat{\mathbf{N}}_s$ instead of the true geometric normal $\hat{\mathbf{N}}_g$.

To solve this problem, we need to find all the points **P** on the boundary which can refract light from **L** towards **V** and compute how much light travels along each such **LPV** path.

In this section, we first present a half-vector formulation of the problem that can be used with shading normals. Based on this formulation we then present fast, iterative solution techniques for: triangles without shading normals (Section 3.2) and with shading normals (Section 3.3). For the latter, we provide both a basic iterative solution that works well in practice and a more expensive interval-based modification that is guaranteed to find all solutions. And finally, we present efficient hierarchical algorithms to handle to media with complex triangle mesh boundaries (Section 3.4).

3.1 Half-vector problem formulation

One way to formulate this problem [Mitchell and Hanrahan 1992] is using Fermat's principle. Let us define the optical path length as $\eta\|\mathbf{V}-\mathbf{P}\|+\|\mathbf{L}-\mathbf{P}\|$, then the valid solutions for **P** are the extremal points (minima and maxima) of this equation under the restriction that **P** must lie on the boundary. Or equivalently, the gradient of this length (as a function of **P**) must be collinear with the geometric surface normal $\hat{\mathbf{N}}_g(\mathbf{P})$. However Fermat's principle does not hold when using shading normals, such as triangles with interpolated normals or normal maps. Instead we use an alternate formulation based on the refractive half-vector $\hat{\mathbf{H}}$ [Walter et al. 2007], which does generalize easily to handle varying shading normals (i.e., the geometric normal is not equal to the shading normal, $\hat{\mathbf{N}}_g \neq \hat{\mathbf{N}}_s$).

Let us define a direction (unit vector) $\hat{\mathbf{H}}$ as follows:

$$\hat{\omega}_V = \frac{\mathbf{V}-\mathbf{P}}{\|\mathbf{V}-\mathbf{P}\|} \tag{1}$$

$$\hat{\omega}_L = \frac{\mathbf{L}-\mathbf{P}}{\|\mathbf{L}-\mathbf{P}\|} \tag{2}$$

$$\hat{\mathbf{H}} = \frac{\eta\,\hat{\omega}_V + \hat{\omega}_L}{\|\eta\,\hat{\omega}_V + \hat{\omega}_L\|} \tag{3}$$

Then $\hat{\mathbf{H}}$ is collinear with the surface normal at all valid refraction points **P**, since requiring that the components of $\hat{\mathbf{H}}$ perpendicular to

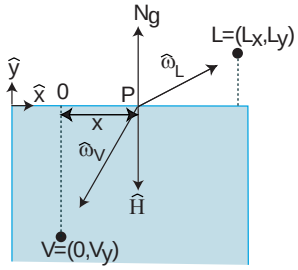

Figure 3: The setup for a triangle without shading normals which reduces the 2D search for **P** to a 1D search in the incidence plane.

the normal vanish is equivalent to Snell's law. Assuming the relative index of refraction $\eta > 1$, **P** is a valid solution to our refracted path finding problem if and only if:

$$\hat{\mathbf{H}} = -\hat{\mathbf{N}}_s \quad \text{and} \quad (\mathbf{V}\cdot\hat{\mathbf{N}}_g) \leq (\mathbf{P}\cdot\hat{\mathbf{N}}_g) \leq (\mathbf{L}\cdot\hat{\mathbf{N}}_g) \tag{4}$$

which tests that Snell's law is obeyed relative to the shading normal $\hat{\mathbf{N}}_s$ and that **V** and **L** lie on opposite sides of the local geometric tangent plane. We use the conventions that the normals point outwards from the refractive object while $\hat{\mathbf{H}}$ points into the denser medium.

In general there is no analytical solution for this set of equations. Prior work has focused on iterative solutions for geometric representations with nice analytic properties and derivatives such as implicit surfaces [Mitchell and Hanrahan 1992; Chen and Arvo 2000]. However, the most common geometric representations in computer graphics are triangle meshes, usually with varying shading normals. These meshes pose different challenges in terms of solving for the refracted paths and are the focus of this paper.

3.2 Case 1: Triangle without shading normals

For a triangle without shading normals, if **V** and **L** lie on the appropriate sides of the triangle's plane, then there always exists a single solution **P** in this plane and it must lie in the incidence plane defined by **V**, **L**, and $\hat{\mathbf{N}}_g$. Thus, we can solve for **P** using a 1D search and then check to see if it lies within the triangle.

Let us work in a coordinate system such that all the relevant points lie in the $z = 0$ plane, the triangle lies in the $y = 0$ plane and **V** is on the y-axis so we can express the problem in 2D using: $\mathbf{V} = (0, V_y)$, $\mathbf{P} = (x, 0)$, $\mathbf{L} = (L_x, L_y)$, and $\hat{\mathbf{N}}_g = \hat{\mathbf{N}}_s = (0, 1)$ (see Figure 3). Then we need only search for x such that $\hat{\mathbf{H}} = -\hat{\mathbf{N}}_g$. Since we already know $\hat{\mathbf{H}}$ and $\hat{\mathbf{N}}_g$ lie in the $z = 0$ plane, this is equivalent to letting $f(x) = (\hat{\mathbf{H}}\cdot(1,0))$, and we can use the Newton-Raphson iterative method for finding x such that $f(x) = 0$.

$$f(x) = \frac{-\eta x}{\sqrt{x^2 + V_y^2}} + \frac{L_x - x}{\sqrt{(L_x - x)^2 + L_y^2}} \tag{5}$$

$$x_0 = \frac{-V_y L_x}{\eta L_y - V_y}\frac{4}{3+\eta} \tag{6}$$

$$x_{i+1} = x_i - f(x_i)/f'(x_i) \tag{7}$$

where x_0 is an initial guess for x, f' is the derivative of f with respect to x, and the last equation is the definition of 1D Newton-Raphson iteration. We have intentionally omitted the normalization term of $\hat{\mathbf{H}}$ because overall scaling factors do not affect the locations of the zeros of a function. We also clamp all x_i to the range $[0, L_x]$ since the solution must lie in this range. Equation 6 is an empirical heuristic that required fewer average iterations to converge than other starting points we tried. With these settings, the Newton-Raphson always converges, and usually in just 2 to 4 iterations.

3.3 Case 2: Triangle with shading normals

When triangles have varying shading normals, the problem is much harder. Within the plane of the triangle there may be zero, one, or multiple solutions for **P** and they need not lie within a single incidence plane, thus requiring a 2D rather than a simpler 1D search. We use a two stage strategy; we progressively split the triangle into smaller sub-triangles until they meet some criteria while trying to prune out sub-triangles that cannot contain a solution. Then we use 2D Newton-Raphson iteration to locate potential solutions in the remaining sub-triangles. We first describe the 2D Newton-Raphson.

2D Newton-Raphson. One key to the successful use of Newton-Raphson and fast convergence is defining a good target function f. If we express the locations on the sub-triangle using barycentric coordinates (a, b) so that a point is inside the sub-triangle if $a \geq 0$, $b \geq 0$, and $a + b \leq 1$, then we can express the point, half-vector, and shading normals as: $\mathbf{P}(a,b)$, $\hat{\mathbf{H}}(a,b)$, and $\hat{\mathbf{N}}_s(a,b)$ and define the target function as:

$$\mathbf{f}(a,b) = \hat{\mathbf{H}}(a,b) + \hat{\mathbf{N}}_s(a,b) \tag{8}$$

which from Equation 4 must go to zero when $\mathbf{P}(a,b)$ is a valid refraction solution.

Standard 2D Newton-Raphson uses the inverse of the Jacobian matrix J of **f**. Since our **f** maps from 2D to 3D, its Jacobian is a 3×2 matrix and not invertible. We tried several approaches to this problem but by far the best was to use the pseudo-inverse $J^+ = (J^T J)^{-1} J^T$. The iteration using the pseudo-inverse is:

$$\begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} = \begin{bmatrix} a_i \\ b_i \end{bmatrix} - J^+(a_i, b_i)\,\mathbf{f}(a_i, b_i) \tag{9}$$

Computing the Jacobian of **f** involves repeatedly applying the following rule for the derivative of a normalized vector (derived via the quotient and chain rules from calculus):

$$\hat{\mathbf{n}} = \frac{\vec{\mathbf{u}}}{\|\vec{\mathbf{u}}\|} \quad \Rightarrow \quad \hat{\mathbf{n}}' = \frac{\|\vec{\mathbf{u}}\|^2\vec{\mathbf{u}}' - (\vec{\mathbf{u}}\cdot\vec{\mathbf{u}}')\vec{\mathbf{u}}}{\|\vec{\mathbf{u}}\|^3} \tag{10}$$

To improve stability, we limit the maximum step size per iteration ($\|\triangle a, \triangle b\| \leq 1/2$) and replace a step with a smaller step size if it fails to reduce $\|\mathbf{f}\|$. We use $a_0 = 1/3$, $b_0 = 1/3$ as our initial starting guess, but if $\|\mathbf{f}(a_0, b_0)\| > 1$ then we also try a few other starting points and start with the one with the least $\|\mathbf{f}\|$ instead. Specifically we also try the vertices of the sub-triangle and the projection of **V** onto the triangle's plane along the average shading normal. With these modifications, we have found the 2D Newton-Raphson works very well. It nearly always converges when there is a real solution in the vicinity of the sub-triangle and usually within 3 to 7 iterations.

3.3.1 Split-and-prune for sub-triangle search

Splitting a triangle into 4 smaller triangles is trivial by splitting each edge; the interesting part of our recursive split-and-prune approach for smooth normals is how to detect when a triangle cannot contain a solution and so can be safely pruned from our search. We accomplish this by computing bounding cones for directions $-\hat{\mathbf{H}}$ and $\hat{\mathbf{N}}_s$ and then pruning the triangle whenever these cones do not overlap.

For triangles with interpolated normals, all the shading normals within the triangle must lie within a spherical triangle defined by the shading normals at the vertices, and it is easy to compute the bounding cone of a spherical triangle. We do not yet support normal mapped triangles, but bounding cones could be precomputed and stored with the normal map in that case.

Computing a bounding cone on $\hat{\mathbf{H}}$ involves a few more steps. Since $(\mathbf{L}-\mathbf{P})/\|\mathbf{L}-\mathbf{P}\|$ and $(\mathbf{V}-\mathbf{P})/\|\mathbf{V}-\mathbf{P}\|$ define two spherical triangles, we compute 3D axis-aligned bounding boxes for each. Then

we scale the second box by η and take their Minkowski sum (which just adds the corresponding bounding values) to get a bounding box on the unnormalized version of $\hat{\mathbf{H}}$. Lastly we compute a bounding cone for this box which is also therefore a bounding cone on $\hat{\mathbf{H}}$. Note that all of our bounding cones have their apex at the origin. Finally, two cones overlap if the angle between their axes is less than the sum of their bounding semiangles.

We have tested two strategies for deciding when to stop subdividing a triangle and run the Newton iteration. The first is a quick heuristic based on the size of these bounding cones; stop if the sum of the $\hat{\mathbf{H}}$ and $\hat{\mathbf{N}}_s$ cone angles is less than thirty degrees. This heuristic has worked well in our experience, producing visually good results, but is not guaranteed to find all solutions. The second is a more expensive interval test with strong guarantees described below.

3.3.2 Guaranteed interval-based refinement test

While 2D Newton iteration converges very quickly (quadratically) once close enough, it may fail to converge even if a solution exists and only finds at most one solution even if there may be several nearby. This can be fixed by subdividing the triangle until we have starting points sufficiently near every solution, but how do we know when its safe to stop subdividing? The cone angles heuristic (see above) works well in practice, but if desired we can get stronger guarantees using interval tests at the cost of some extra subdivision.

Moore and others [Moore 1977; Rall 1981] introduced interval extensions to Newton Raphson iteration. The original formulation required inverting the interval Jacobian matrix, which can be non-trivial. Krawczyk [1969] introduced a formulation which does not require inverting an interval matrix. This formulation was also used by [Mitchell and Hanrahan 1992] for reflections. We modify this formulation to work with our 2D Newton iteration and the pseudoinverse. Both Krawczyk and Newton Raphson are iterative fixed points algorithms where the iteration maps a point to itself if and only if it is a solution to the equation (i.e., $x_{i+1} = x_i \Leftrightarrow f(x_i) = 0$). Krawczyk works with intervals allowing reasoning about regions of the search space, but in the limit of degenerate intervals (single points), it is identical to Newton Raphson.

Let us introduce some notation. $\mathbf{f}(a,b) = 0$ is the equation we are solving. $\mathbf{X} = \{\mathbf{X}_a, \mathbf{X}_b\}$ is a 2D interval representation of our solution space, where $\mathbf{X}_a$ and $\mathbf{X}_b$ are intervals on a and b respectively. For any triangle all valid values of a and b lie within the interval $[0,1]$. $\mathbf{F}(\mathbf{X})$ is the interval representation of $\mathbf{f}$ and includes all possible values that $\mathbf{f}$ can take over points in $\mathbf{X}$. $\mathbf{F}'(\mathbf{X})$ is the interval representation of the Jacobian matrix. $m(\mathbf{X})$ is an operator that replaces each interval in $\mathbf{X}$ by its midpoint. Let $m_0 = m(\mathbf{X}) = (m_a, m_b) = (0.5, 0.5)$, and $\mathrm{M} = m(\mathbf{F}'(\mathbf{X}))$. $\mathbf{F}'(\mathbf{X})$ is a 3×2 matrix of intervals, M is an ordinary (non-interval) matrix and M^+ is the pseudoinverse of M. We compute $\mathbf{K}(\mathbf{X})$ as:

$$\mathbf{K}(\mathbf{X}) \quad = \quad m_0 - \mathrm{M}^+ f(m_0) + [I - \mathrm{M}^+ \mathbf{F}'(\mathbf{X})](\mathbf{X} - m_0) \quad (11)$$

A solution, or fixed point, can not exist in the interval $\mathbf{X}$ if either of these conditions is true:

$$\emptyset \notin \mathbf{F}(\mathbf{X}) \quad (12)$$

$$\mathbf{K}(\mathbf{X}) \cap \mathbf{X} = \emptyset \quad (13)$$

Furthermore, if both of the following conditions are true, then there is exactly one unique solution in $\mathbf{X}$, and ordinary (non-interval) Newton iteration will converge to it.

$$\mathbf{K}(\mathbf{X}) \subseteq \mathbf{X} \quad (14)$$

$$||\mathbf{I} - \mathrm{M}^+ \mathbf{F}'(\mathbf{X})|| < 1 \quad (15)$$

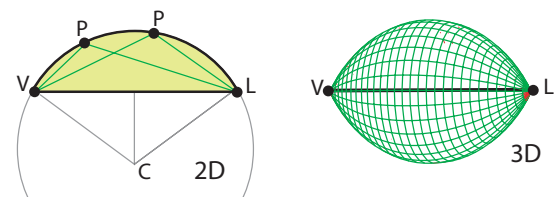

Figure 4: Spindle test. Valid solutions for $\mathbf{P}$ must be within the shaded region in 2D, and within the surface of revolution in 3D.

where the first condition requires that the interval maps back to a subset of itself, and combined with the second condition guarantees that the interval continues to shrink at each subsequent iteration.

When using the interval-based refinement test, we skip sub-triangles if Equations 12 or 13 are true. If both Equations 14 and 15 are true, then we switch to ordinary 2D Newton iteration which is guaranteed to converge (though the solution may lie outside the sub-triangle). Otherwise, we recursively subdivide our triangle and keep searching for solutions.

3.4 Scalability to triangle meshes

For meshes with small numbers of triangles, one can simply test all triangles for solutions, as described above, but to handle larger meshes we use a hierarchical pruning process to eliminate whole groups of triangles that cannot contain a solution. We first construct a hierarchy over the triangles called a position-normal tree [Bala et al. 2003; Snyder 1992]. This is a binary tree where the leaves are triangles and each node contains a 3D axis-aligned bounding box over positions and a bounding cone over shading normals that is valid for all its descendants. We construct this hierarchy using a bottom-up approach similar to that in [Walter et al. 2005].

To solve for all refractive connection paths through a triangle mesh, we perform a depth-first traversal of the position-normal tree. At each node we test to see if we can prune out the subtree by proving it has no solutions using the following three pruning tests: the spindle test, the sidedness test, and the cone overlap test. We now briefly describe each of these tests; for more details see [Walter et al. 2009].

The spindle test. A single refraction can only bend the light by a limited amount. Even for arbitrarily large values of η, the light can be bent by at most a right angle and for more reasonable η values, the bending is more restricted. Specifically the angle between $\mathbf{PL}$ and $\mathbf{PV}$ must form an angle between $\pi/2 + \arcsin(1/\eta)$ and π in radians. This is only possible if $\mathbf{P}$ lies within a region centered around the line segment $\mathbf{VL}$ as shown in Figure 4. This region, which we call the spindle, is a surface of revolution of a circular arc from $\mathbf{V}$ to $\mathbf{L}$. This shape is a direct consequence of the refractive bending angle limit and the inscribed-angle theorem for circles.

We have developed a simple test to see if a bounding box may overlap this region. Let r be the distance from a point $\mathbf{P}$ to the line containing $\mathbf{L}$ and $\mathbf{V}$. Then the point $\mathbf{P}$ lies within the spindle if:

$$\left\| \mathbf{P} - \frac{\mathbf{L}+\mathbf{V}}{2} \right\|^2 + \frac{\|\mathbf{L}-\mathbf{V}\|}{\eta} r \leq \frac{\|\mathbf{L}-\mathbf{V}\|^2}{4} \quad (16)$$

If η is arbitrarily large, this reduces to a point in sphere test, but becomes more stringent as η decreases. We use a slightly modified standard box-sphere overlap test to test if the box could overlap the spindle by also computing a lower bound on the distance from the box to the line containing $\mathbf{V}$ and $\mathbf{L}$. This test is cheap to compute and does not depend on the shading normals, allowing it to prune many subtrees even if they have large normal bounding cones (i.e., large normal variation).

The sidedness test. For a point to be a valid refraction point, $\mathbf{L}$ must lie on the positive side of its shading normal, and $\mathbf{V}$ must lie on the negative side of the shading normal and within a cone bounded by the critical angle, $\arcsin(1/\eta)$. Using techniques from [Walter et al. 2005], we can compute bounds on the angles between the axis of the shading normal bounding cone and $\mathbf{L}$ and $\mathbf{V}$ as seen from any point in the position bounding box. We then offset these by the semiangle of the normal bounding cone and see if it is possible for the respective angles to lie within the required ranges.

The cone overlap test. This test is a simple extension of the pruning test used for pruning sub-triangles in the smooth normal case. The one difference is that the positions are bounded by a 3D box rather than a 2D triangle, but the test is otherwise the same.

4 Contribution of Refracted Light Paths

Once we have computed a refracted path connecting $\mathbf{L}$ and $\mathbf{V}$ through some point $\mathbf{P}$ on the boundary, we need to compute how much contribution it makes to the illumination. Most of the terms are similar to the contribution from a regular shadow ray. For example, if $\mathbf{L}$ is a point light with intensity I_e, F is the fresnel factor at $\mathbf{P}$, A is the volume attenuation (i.e., integral of σ_s along the line segment inside the media), and $\mathbf{V}$ is a volume scattering event with phase function ρ, then the contribution for an unoccluded path is:

$$\text{contribution} = \frac{I_e\, F\, A\, \rho}{D} \tag{17}$$

where D is the distance correction factor discussed below. If the medium is clear then $A = 1$. If $\mathbf{V}$ is a surface point, then replace ρ with the BRDF times the cosine of angle with surface normal at $\mathbf{V}$. If simulating an area light then replace I_e with the emitted radiance times the cosine of angle with surface normal at $\mathbf{L}$, divided by the probability of sampling $\mathbf{L}$.

If there is no refraction ($\eta = 1$) then we have the usual distance squared factor, $D = \|\mathbf{L} - \mathbf{V}\|^2$. If there is refraction but we are at normal incidence without shading normals, $\hat{\omega}_L = \hat{\mathbf{N}}_g = \hat{\mathbf{N}}_s$, then we get $D = (\|\mathbf{P} - \mathbf{V}\| + \eta\,\|\mathbf{L} - \mathbf{P}\|)^2$. For the special case of no shading normals (i.e., $\hat{\mathbf{N}}_s = \hat{\mathbf{N}}_g$) the exact equation is:

$$D = (d_V + \eta d_L)\left(\frac{|\hat{\omega}_L \cdot \hat{\mathbf{N}}_g|}{|\hat{\omega}_V \cdot \hat{\mathbf{N}}_g|} d_V + \frac{|\hat{\omega}_V \cdot \hat{\mathbf{N}}_g|}{|\hat{\omega}_L \cdot \hat{\mathbf{N}}_g|} \eta d_L\right) \tag{18}$$

with d_V and d_L defined below.

In the general case of shading normals, D is more complex and does not seem to have a simple equation. Computing D correctly is necessary to get visually good results that match reference solutions. First, we compute derivatives for $\mathbf{L}$ in the plane perpendicular to $\hat{\omega}_L$ with respect to perturbations in $\hat{\omega}_V$, which is equivalent to computing the solid angle of a small area around $\mathbf{L}$ as seen from $\mathbf{V}$. This was inspired by and uses techniques from ray differentials [Igehy 1999]. We can compute this differential using the following equations:

$$d_V = \|\mathbf{V} - \mathbf{P}\| \tag{19}$$

$$d_L = \|\mathbf{L} - \mathbf{P}\| \tag{20}$$

$$\mathbf{P}' = d_V\left(\hat{\omega}_V' - \frac{\hat{\omega}_V' \cdot \hat{\mathbf{N}}_g}{\hat{\omega}_V \cdot \hat{\mathbf{N}}_g}\hat{\omega}_V\right) \tag{21}$$

$$\mu = \hat{\omega}_L \cdot \hat{\mathbf{N}}_s + \eta\, \hat{\omega}_V \cdot \hat{\mathbf{N}}_s \tag{22}$$

$$\mu' = \left(\eta^2 \frac{\hat{\omega}_V \cdot \hat{\mathbf{N}}_s}{\hat{\omega}_L \cdot \hat{\mathbf{N}}_s} + \eta\right)\left(\hat{\omega}_V' \cdot \hat{\mathbf{N}}_s + \hat{\omega}_V \cdot \hat{\mathbf{N}}_s'\right) \tag{23}$$

$$\hat{\omega}_L' = \eta\, \hat{\omega}_V' + \mu' \hat{\mathbf{N}}_s + \mu \hat{\mathbf{N}}_s' \tag{24}$$

$$\mathbf{L}' = \mathbf{P}' - (\mathbf{P}' \cdot \hat{\omega}_L)\hat{\omega}_L + d_L \hat{\omega}_L' \tag{25}$$

model	boundary	smooth	other	time
teapot	12	N	4096	15.3s
cuboctahedron	20	N	0	13.9s
amber	36	N	60556	19.2s
glass tile	798	Y	60	66.9s
glass mosaic	20813	Y	1450	87.8s
pool	2632	Y	4324	59.4s
bumpy sphere	9680	Y	0	304.3s

Figure 5: Results for CPU version of our method on seven models. Shows the number of triangles in refractive boundaries, whenever the boundary uses interpolated normals, the number of other triangles in the scene, and time to compute a 512x512 image using 64 samples per pixel (except bumpy sphere used 128 samples).

where we use the notation that a' is the derivative of a with respect to a small perturbation in $\hat{\omega}_V$ and perpendicular to it. We compute these derivatives for two different directions perpendicular to $\hat{\omega}_V$ and to each other to get $\mathbf{L}'_\perp$ and $\mathbf{L}'_\parallel$ and the correct distance factor from their vector cross product as:

$$D = \|\mathbf{L}'_\perp \times \mathbf{L}'_\parallel\| \tag{26}$$

We were surprised by the seeming complexity of D, but have not managed to find a simpler form for it. However, it is not difficult to code once the equations are known and the only other piece needed is the ability to compute the derivatives of the shading normal $\hat{\mathbf{N}}_s$ which can easily be computed for triangles with interpolated normals [Igehy 1999]. For more discussion see [Walter et al. 2009].

5 Results

We implemented our refractive path finding algorithms in both CPU and GPU versions. The CPU version is implemented in a Java-based ray tracer using Sun's 1.6 server JVM and running on an 8-core, 2.83 Ghz machine (dual Intel Xeon 5440 chips). All CPU results are for 512×512 images with 64 samples per pixel except the bumpy sphere that used 128. Each eye ray is traced through refractive or reflective bounces until it hits a diffuse or glossy surface or scatters in a volume (importance sampled according to the volume attenuation along eye ray). For points inside a refractive medium, we connect it to the light source using our techniques by finding all connecting refractive paths, checking them for occlusion, and computing their contribution. Points outside the medium are shaded using conventional direct illumination. Results and statistics for our scenes are shown in Figures 1 and 5. All the refractive media are homogeneous with $\eta = 1.5$. Each scene is lit by one point light except for the pool which has two small area lights.

The only additional storage required when adding our technique to a ray tracer is the position-normal tree, currently about 100 bytes per boundary triangle in our implementation. The tree build is very fast; for the glass tile model it takes 6ms and only needs to be rebuilt if the geometry is deformed.

The first three models demonstrate faceted refraction without interpolated normals. The teapot embedded in purple glass example shows interesting refractive effects even in a very simple configuration with three distinct refractive beams lighting the teapot. The cuboctahedron shows single scatter volume caustics including brightening due to beam overlap near edges and vertices. The scorpion in amber combines both volume and surface effects.

The glass tile is a piece of rounded glass with a diffuse backing, modeled with only 798 triangles but with interpolated normals to simulate a smooth surface. You can see sharp caustics caused by the focusing of light by the simulated curved surface. The glass mosaic combines 25 such glass tiles into a more complex shape.

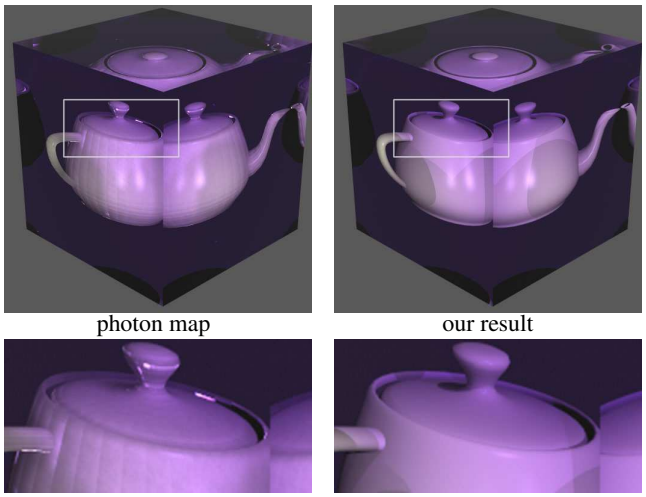

Figure 6: Glossy teapot embedded in purple glass cube model rendered using photon mapping with 20 000 photons and our result. The photon map solution shows blurring, light leaks, and other visual artifacts despite taking twice as long to compute as our result. Bottom row shows a zoom-in of a region in the result images.

The pool is modeled after a famous image from [Veach and Guibas 1997], but with two much smaller area lights that make the caustics sharper and much harder to find by conventional methods.

The bumpy sphere is our most challenging model because it focuses light into surprisingly small and detailed volume caustics in its interior. The radiance in such as caustic can grow arbitrarily large near focal points or cusps in the caustics leading to spike noise when sampling the volume along eye rays even if the region has little effect on the integral due to its small size. This noise can be greatly reduced by clamping D (Equation 26) to be larger than a minimum threshold at the cost of some loss of energy. For this model only, we used a manually chosen clamping threshold that greatly reduces noise with little energy loss, but more research is needed in automatic ways to handle such caustics.

We used our cone angles triangle subdivision heuristic for all our result images. The interval-based refinement test with convergence guarantees is typically 2 to 5 times slower depending on the scene, while producing visually equivalent images for these examples. On rare occasions, the heuristic misses a few paths near caustic cusps that the interval test finds, but in our results this affects at most a few pixels per image and the differences are quite hard to spot. The position-normal tree and hierarchical pruning are essential for getting good performance on boundary meshes with more than a few triangles. For example the pool scene took 59.4s to render with the refinement heuristic and hierarchy, 141.1s with the interval refinement test, and 1934.6s without hierarchical pruning (slowing down by 2.4x and 32x respectively).

A comparison to photon mapping for the teapot model is shown in Figure 6. Photon mapping is a versatile algorithm that can simulate a wider range of light paths than our technique. However the blurring inherent in its density estimation kernel can cause a variety of visual artifacts. In this example, even though the photon map image took twice as long as our result, you can see significant blurring of the shadow boundaries, light leaks that coupled with the glossy BRDF lead to bright speckles, and faceting artifacts caused by discontinuities in the photon distribution between facets coupled with the shading normal modified BRDF. Typically a final gather pass would used to greatly ameliorate such artifacts, but for single scattering, this is not feasible as it would degenerate to path tracing.

Figure 7 shows the bumpy sphere rendered with four algorithms.

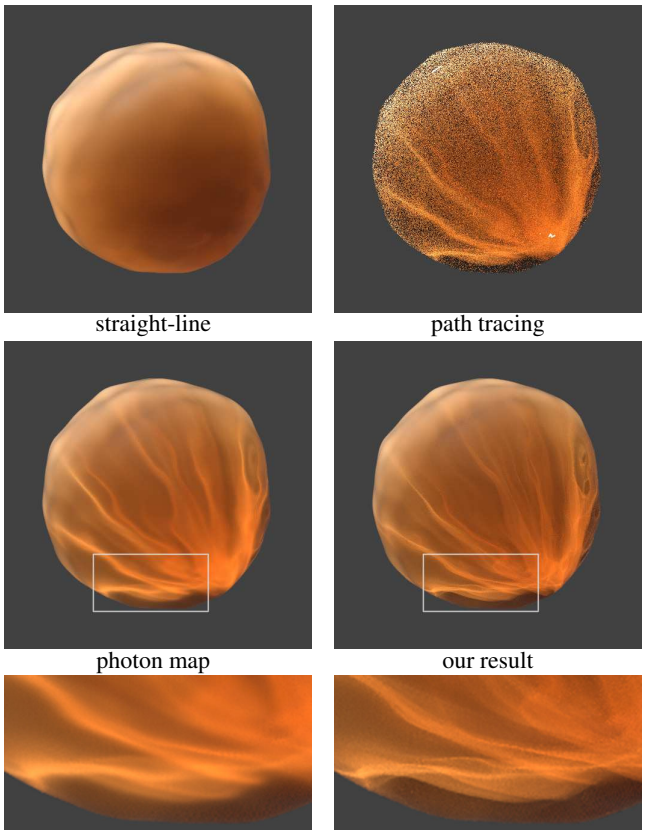

Figure 7: Back lit bumpy sphere with four rendering algorithms. The straight-line approximation cannot capture the volume caustics. Path tracing required replacing the point light with a small area source and even with 32,768 samples per pixel (compute time: 1.4 hours) produces a very noisy result (white region is a reflection of the area source). The photon map is much better and takes roughly equal time as our result, but even with ten million photons, it still blurs out the finer details of the caustics as shown in the bottom row zoom-in. Our algorithm is able to capture these fine details without the high memory or time requirements of the other methods.

The common approximation of ignoring refraction on shadow rays is cheap, but eliminates all the visually rich detail in the image. To get path tracing to work at all on this scene we had to replace the point light with a small area light and then convergence is very slow. Even after 1.4 hours, the path tracing result is extremely noisy. Bidirectional path tracing would be even less efficient for the types of paths we are simulating. Photon mapping is much better, but requires very high photon densities to reduce its inherent blurring. In this example, even storing a ten million photons in the sphere is not enough to resolve the fine detail that you can see in our result. Photon mapping and our result took roughly equal time, but our method uses far less memory (hundreds of kilobytes vs. hundreds of megabytes) and captures finer details.

GPU results. We have also implemented a slightly more limited GPU version of our algorithm that allows us to achieve interactive performance on several models. The GPU version is written in CUDA 2.0 and runs on an nVIDIA GTX 280 card with driver version 181.20. We used a GPU ray tracer based on [Popov et al. 2007] along with the methods described in this paper except that we do not perform dynamic triangle subdivision on the GPU. Each GPU thread handles all the tasks for a pixel including ray tracing, hierarchical pruning, and Newton iteration. Results used 2 samples per pixel for anti-aliasing, but with multiple volume samples per

model	time	fps
teapot	0.1s	10fps
cuboctahedron	0.14s	7fps
amber	0.3s	3fps
glass tile	6.9s	0.14fps

Figure 8: Results for GPU version of our method on four scenes.

eye ray (40 for cuboctahedron and 100 for amber). Timing statistics for our GPU version are in Figure 8. Result images from the GPU are shown in the accompanying video.

Limitations. Our method is designed to find refractive connections that only cross the refractive boundary once. Thus while it works well for rendering many caustic effects inside a medium, there are other caustic effects it cannot compute such as the caustic a glass object casts on a table. Our method is easily extended to simulate low-order scattering by continuing the eye rays, but is not a good way to compute higher-order scattering such as in the diffusion regime where other more appropriate approximations exist. Our method assumes the index of refraction is constant inside the medium though other volume properties such as the scattering coefficients could be inhomogeneous.

6 Conclusions

We have presented an accurate and efficient solution for solving for single scattering in a refractive medium. This is an important class of light paths which are often neglected or approximated simplistically, due to the difficulty of solving these paths accurately and due to the relative lack of general solution methods.

Our techniques explicitly solve for paths connecting two points: one inside a refractive medium and the other outside. We present a formulation of this problem that can handle triangle meshes with interpolated, or otherwise perturbed, shading normals. We introduce techniques to solve for paths through individual triangles and hierarchical culling techniques to handle larger meshes without needing to test every triangle. We have also presented an exact method to compute the contribution along each such refracted path. Our paper allows us to compute single scattering in refractive media efficiently without sacrificing accuracy.

We believe our work can be extended to enhance Monte Carlo algorithms such as bidirectional path tracing by allowing them to connect subpaths in different media thus allowing them to more efficiently find these types of problematic paths. Our work could also be used to compute reflective caustics with a small change in definition of the half-vector and pruning tests. It is also possible to extend the method to find paths that cross more than one refractive boundary. However this would greatly expand the search space and better culling methods would be needed to make this practical.

Acknowledgements This work was supported by NSF CAREER 0644175, NSF CPA 0811680, NSF CNS 0615240, NSF CNS 0403340, and grants from Intel Corporation, NVidia Corporation, and Microsoft Corporation. This work was started while N. Holzschuch was on a sabbatical at Cornell, funded by INRIA, and he'd like to acknowledge the INRIA CIPRUS associate team. LJK is UMR 5224, a joint research laboratory of CNRS, INRIA, INPG, U. Grenoble I and U. Grenoble II.

References

BALA, K., WALTER, B., AND GREENBERG, D. P. 2003. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics 22*, 3, 631–640.

CHEN, M., AND ARVO, J. 2000. Theory and application of specular path perturbation. *ACM Transactions on Graphics 19*, 4, 246–278.

ERNST, M., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Interactive rendering of caustics using interpolated warped volumes. In *Graphics Interface 2005*, 87–96.

ESTALELLA, P., MARTIN, I., DRETTAKIS, G., AND TOST, D. 2006. A gpu-driven algorithm for accurate interactive reflections on curved objects. In *Rendering Techniques 2006 (Proc. EG Symp. on Rendering)*.

HANRAHAN, P., AND KRUEGER, W. 1993. Reflection from layered surfaces due to subsurface scattering. In *Computer Graphics* Proceedings, ACM SIGGRAPH, Annual Conference Series, 165–174.

IGEHY, H. 1999. Tracing ray differentials. In *Computer Graphics* Proceedings, ACM SIGGRAPH, Annual Conference Series, 179–186.

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2003. A fast rendering method for refractive and reflective caustics due to water surfaces. *Computer Graphics Forum 22*, 3 (Sept.), 601–610.

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. Eurographics '08) 27*, 2.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Computer Graphics* Proc., ACM SIGGRAPH, Annual Conf. Series, 511–518.

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters.

KRAWCZYK, R. 1969. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing 4*, 3, 187–201.

MITCHELL, D., AND HANRAHAN, P. 1992. Illumination from curved reflectors. *Computer Graphics (Proc. of Siggraph) 26*, 3, 283–291.

MOORE, R. E. 1977. A test for existence of solutions for non-linear systems. *SIAM Journal on Numerical Analysis 14*, 4, 611–615.

NISHITA, T., AND NAKAMAE, E. 1994. Method of displaying optical effects within water using accumulation buffer. In *Computer Graphics* Proceedings, ACM SIGGRAPH, Annual Conference Series, 373–379.

NISHITA, T., MIYAWAKI, Y., AND NAKAMAE, E. 1987. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *Computer Graphics (Proc. of Siggraph) 21*, 4, 303–310.

OFEK, E., AND RAPPOPORT, A. 1998. Interactive reflections on curved objects. In *Computer Graphics* Proceedings, ACM SIGGRAPH, Annual Conference Series, 333 – 342.

POPOV, S., GÜNTHER, J., SEIDEL, H.-P., AND SLUSALLEK, P. 2007. Stackless kd-tree traversal for high performance gpu ray tracing. *Computer Graphics Forum 26*, 3 (Sept.), 415–424.

RALL, L. B. 1981. *Automatic Differentiation: Techniques and Applications*, vol. 120 of *Lecture Notes in Computer Science*. Springer.

ROGER, D., AND HOLZSCHUCH, N. 2006. Accurate specular reflections in real-time. *Computer Graphics Forum (Proc. of EG 2006) 25*, 3.

SNYDER, J. M. 1992. Interval analysis for computer graphics. *Computer Graphics 26*, 4 (July), 121–130.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S. G., AND NAYAR, S. K. 2005. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics 24*, 3, 1040–1049.

SUN, X., ZHOU, K., STOLLNITZ, E., SHI, J., AND GUO, B. 2008. Interactive relighting of dynamic refractive objects. *ACM Transactions on Graphics 27*, 3 (Aug.), 35:1–35:9.

SZÉCSI, L. 2006. The hierarchical ray engine. In *WSCG (Winter School of Computer Graphics)*.

SZIRMAY-KALOS, L., ASZÓDI, B., LAZÁNYI, I., AND PREMECZ, M. 2005. Approximate ray-tracing on the GPU with distance impostors. *Computer Graphics Forum (Proc. Eurographics '05) 24*, 3.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Computer Graphics* Proceedings, ACM SIGGRAPH, Annual Conference Series, 65–76.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics 24*, 3 (Aug.), 1098–1107.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet Models for Refraction through Rough Surfaces . In *Rendering Techniques (Proc. EG Symposium on Rendering)*, 195–206.

WALTER, B., ZHOU, S., HOLZSCHUCH, N., AND BALA, K. 2009. Supplemental to single scattering in refractive media with triangle mesh boundaries. Technical Report PCG-??-??, Cornell, Aug.