EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

Single-pass Scalable Subsurface Rendering with Lightcuts

Adam Arbree, Bruce Walter & Kavita Bala

Cornell University†

Abstract

This paper presents a new, scalable, single pass algorithm for computing subsurface scattering using the diffusion approximation. Instead of pre-computing a globally conservative estimate of the surface irradiance like previous two pass methods, the algorithm simultaneously refines hierarchical and adaptive estimates of both the surface irradiance and the subsurface transport. By using an adaptive, top-down refinement method, the algorithm directs computational effort only to simulating those eye-surface-light paths that make significant contributions to the final image. Because the algorithm is driven by image importance, it scales more efficiently than previous methods that have a linear dependence on translucent surface area. We demonstrate that in scenes with many translucent objects and in complex lighting environments, our new algorithm has a significant performance advantage.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The appearance of many translucent materials, including marble, leaves and human skin, depends on light scattered both through the material as well as reflected from its surface. Because of the long, complex paths light takes through the material, Monte Carlo simulation of subsurface scattering is computationally expensive. However, for homogeneous, highly scattering materials, the problem can be simplified and more efficient solutions exist. In these cases, the subsurface scattering can be approximated using an analytic dipole expansion [JMLH01]. The simulation is often separated into two passes [JB02]. In the first pass the light incident on the surfaces of all translucent objects is calculated. The second pass uses this pre-computed irradiance to calculate the subsurface transport at each shading point for all pixels. While sharing irradiance estimates in the second pass significantly improves performance in some scenes, we show that in large scenes with complex illumination, like Figure 1, the first pass with its linear, brute force irradiance evaluation becomes a significant expense.

Our work addresses this scalability problem with a novel, single-pass algorithm based on Multidimensional Lightcuts

Figure 1: Image of the Mezquita de Cordoba model lit by a sun/sky model, direct point lights and global illumination. The capital of each column and half of the bricks in the arches are a translucent marble. Such scenes, with large amounts of translucent material, are expensive to render using traditional methods.

† email: {arbree,bjw,kb}@graphics.cornell.edu

[WABG06] that simultaneously estimates both the subsurface transport and the surface irradiance. Our new algorithm adapts the accuracy of both estimates relative to the contributions of complete eye-subsurface-light paths. Additionally, like the two pass method, our algorithm avoids recomputation of the surface irradiance. This adaptability allows our algorithm to perform more efficiently in scenes that contain many translucent objects especially under complex illumination.

The rest of the paper proceeds as follows. The next section presents background and previous work. In section 3, we discuss the motivation for our algorithm. In the next two sections, we present our algorithm and describe some important implementation details. Section 6 compares the performance of the new algorithm with that of [JB02] and finally we conclude in section 7.

2. Background and Related Work

2.1. Diffusion Approximation

Subsurface scattering is often represented by a Bidirectional Subsurface Scattering Reflectance Distribution Function (BSSRDF) [NRH*77]. The radiance scattered in the subsurface $L^{ss}(x,\omega)$ is computed by integrating the product of the BSSRDF and the incoming radiance over the surface of each translucent object:

$$L^{ss}(x,\omega) = \int_A \int_\Omega S(x,\omega,x_i,\omega_i) \cdot L^i(x_i,\omega_i)(n\cdot\omega_i)d\omega_i\, dA(x_i) \quad (1)$$

where n is the surface normal at x_i. Jensen et al. [JMLH01] solved Equation 1 by splitting the BSSRDF into two terms—a single scattering term S^1 and a multiple scattering term S^d—and computing each integral separately.

$$S(x,\omega,x_i,\omega_i) = S^1(x,\omega,x_i,\omega_i) + S^d(x,\omega,x_i,\omega_i) \quad (2)$$

The single scattering term S^1 was previously shown by Hanrahan and Kruger [HK93] to have an efficient Monte Carlo solution. Computing the multiple scattering term is more difficult. Jensen et al. [JMLH01] simplify the calculation by assuming a homogeneous, highly scattering material where the subsurface transport is nearly diffuse and is well represented by its 1^{st} order expansion in spherical harmonics. The resulting diffusion approximation can be solved analytically for the one dimensional diffuse BSSRDF $R(\| x-x_i \|)$. The multiple scattering term can be computed by solving

$$L^d(x,\omega) = \frac{1}{\pi}T(\eta,\omega)\int_A R(\| x-x_i \|) \cdot \int_\Omega T(\eta,\omega_i)L^i(x_i,\omega_i)(n\cdot\omega_i)d\omega_i\, dA(x_i) \quad (3)$$

where $T(\eta,\omega)$ is the Frensel transmittance and η is the material's relative index of refraction. Recently [DJ05] provided a similar solution for thin slabs using a multi-pole expansion.

2.2. Subsurface Rendering

Many recent works have discussed different aspects of subsurface rendering but few works have specifically considered scalably rendering large scenes under complex illumination. Monte Carlo (MC) methods can accurately solve general, volume scattering problems and can be directly applied to subsurface scattering. Dorsey et al. [DEJ*99] used photon mapping to simulate the appearance of weathered stone and Pharr and Hanrahan [PH00] formalized a general, MC scattering operator. However, even for simple lighting, pure MC algorithms remain expensive. Jensen et al. [JMLH01] presented a modified MC algorithm to solve the simplified diffuse BSSRDF (Equation 3) but its significant cost motivated a follow-up, two pass algorithm [JB02]. This two pass algorithm most closely relates to our work and is discussed in detail in Section 4.2.2. Recently a paper by Donner and Jensen [DJ07] presented another two pass algorithm based on photon mapping [Jen96]. By tracing photons within the subsurface material, their algorithm addresses the problems of approximating volumetric shadows and capturing difficult light paths that pass through translucent materials multiple times. Unfortunately capturing these effects is expensive and the new algorithm had to forgo the performance advantages of [JB02]. Since our work focuses on performance, we will limit our later discussion in Section 4.2.2 to the earlier two pass method.

Several hybrid algorithms have combined both MC methods and the dipole solution [LPT05, CTW*04, TWL*05]. These works all use a similar approach; they split the subsurface volume into an inner core and an outer shell. Light transport in the inner core is estimated using the dipole, diffusion approximation while MC methods are used in the outer shell. The hybrid method of Li et al. [LPT05] detects and avoids the errors that arise from using the dipole approximation in optically thin regions. The two other works [CTW*04, TWL*05] use hybrid methods to render heterogeneous materials. None of these works considers scalably extending subsurface rendering to large scenes or complex illumination.

Haber et al. [HMBR05] use a multi-grid, finite element solver to efficiently estimate subsurface transport using the diffusion approximation, but like [JB02] (see Section 4.2.2) use a potentially expensive, brute force surface irradiance calculation. Several works [HBV03, MKB*03b, HV04, MKB*03a, DS03] describe GPU algorithms for interactive, subsurface scattering, but the limitations of graphics hardware fundamentally restrict lighting to a few point sources. Precomputed radiance transfer (PRT) [SKS02] techniques can be used to relight translucent objects under complex, global illumination at interactive rates [WTL05, SLS05]; however, they rely on expensive pre-computation using other methods discussed above.

2.3. Multidimensional Lightcuts

Multidimensional Lightcuts (MDLC) [WABG06] is a scalable, hierarchical, point-based algorithm for evaluating the integrated radiance through a pixel.

$$L^{pixel} = \int_P \int_\Omega L(x,\omega) d\omega dP$$

The MDLC algorithm first discretizes the problem. It represents the radiance function with a set of light samples $\mathbb{L}$ and the spatiotemporal integration domain P with a set of gather samples $\mathbb{G}$. This reduces the integral to a summation over all pairs of light and gather samples.

$$L = \sum_{(j,i)\in\mathbb{G}\times\mathbb{L}} G_j F_{ji} I_i \tag{4}$$

where G_j is the relative importance of gather sample j, I_i is the intensity of light sample i and F_{ji} is the point-to-point form factor between j and i (for details, see [WABG06]).

To efficiently compute Equation 4, the MDLC algorithm clusters similar gather-light pairs and replaces each cluster's sum with an error bounded approximation. The clustering is chosen by exploring a hierarchy of potential clusters. Since this cluster hierarchy can be very large, it is implicitly constructed as needed by incrementally computing the Cartesian product graph of two explicit, hierarchical, binary clusterings of the gather and light samples. Each node in the implicit product graph represents a gather cluster/light cluster pair $(\mathbb{C}_\mathbb{G}, \mathbb{C}_\mathbb{L})$. The MDLC algorithm selects a clustering by computing a cut through the product graph. A cut is a set of nodes such that, for any leaf node, the set of all paths from the root to that leaf node will contain exactly one node from the cut. The algorithm computes a cut by starting with the node containing the root cluster, a trivial cut, and iteratively replacing the highest error node. The new nodes are formed by replacing either the gather or the light cluster with the children from its respective hierarchy. The iteration stops when the error of the highest error node falls below a perceptual threshold.

After selecting a cut, the algorithm computes Equation 4 by approximating the contribution of each cut node using the constant form factor for a single representative sample pair $(g,l) \in \mathbb{C}_\mathbb{G} \times \mathbb{C}_\mathbb{L}$

$$L_\mathbb{C} = F_{gl}\Big[\sum_{j\in\mathbb{C}_\mathbb{G}} G_j\Big]\Big[\sum_{i\in\mathbb{C}_\mathbb{L}} I_i\Big] \tag{5}$$

Equation 5 can be computed efficiently by pre-computing the two final sums during the creation of the gather and light sample hierarchies. During cut selection, a cut node's error is estimated by its upper bound and is computed by replacing F_{gl} in Equation 5 with an upper bound on its maximum possible value

$$\| L_\mathbb{C}^{true} - L_\mathbb{C}^{est} \| \leq F_{gl}^{(ub)}\Big[\sum_{j\in\mathbb{C}_\mathbb{G}} G_j\Big]\Big[\sum_{i\in\mathbb{C}_\mathbb{L}} I_i\Big] \tag{6}$$

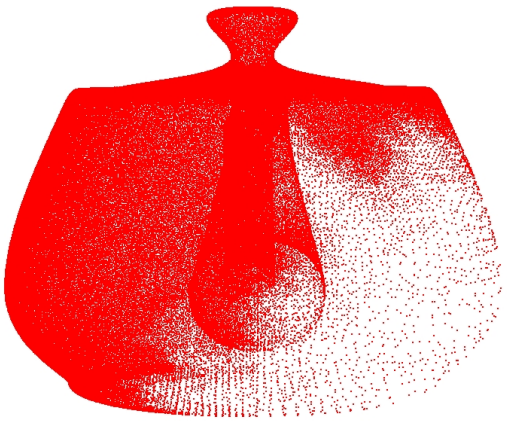


Figure 2: Visualization of the surface points where BSSRDF links are evaluated in the second pass of the [JB02] algorithm. The camera is located to the left and above. Notice that many fewer points are present on the side of the teapot opposite the camera.

3. Problem Motivation

Our algorithm computes the multiply scattered subsurface radiance L^d at a single point by solving Equation 3. The solution requires determining two quantities, the light incident on an object's surface and the transport through the material. Both quantities depend on arbitrary scene parameters: the former on the lighting environment and the latter on the object's geometry. Because of this dependence on arbitrary, potentially high-variance parameters, algorithms for solving Equation 3 must densely sample the integrand to ensure an accurate final result. This dense computation traditionally makes evaluating the multiple scattering term the most expensive component of subsurface rendering.

Previous work by Jensen and Buhler [JB02] described a two-pass algorithm that separately estimates these two quantities. The first pass computes the irradiance at a dense set of surface samples and the second pass gathers the contributions of these samples to compute the subsurface transport. The two-pass algorithm has two advantages. First, the surface irradiance can be pre-computed only once and reused to compute the radiance exitant at many surface points. Second, when computing the subsurface transport in the second pass, the irradiance samples can be clustered and the total subsurface transport estimated with a single BSSRDF evaluation. This is particularly effective because the diffuse, dipole BSSRDF decreases exponentially with distance and large clusters can be used where the BSSRDF term is small.

Despite these advantages, the algorithm scales poorly to large scenes with complex lighting because the cost of the first pass grows linearly with translucent surface area and lighting complexity. The second pass is efficient because it

exploits the exponential differences in the relative importance of the irradiance samples, but during the first pass, these differences are unknown. The algorithm must spend equal effort pre-computing the irradiance in both important and unimportant regions. Figure 2 shows, as a red dot, every surface location where the BSSRDF was evaluated during the second pass when rendering the teapot image in Figure 5. Illustrated by the sparse distribution of dots, there are many regions, such as backfacing areas or surfaces occluded in the camera (in upper left), where the BSSRDF is never densely sampled. In these regions, no irradiance sample makes a significant individual contribution to the image. Pre-computing the irradiance in these regions as accurately as important regions near the camera view, wastes considerable effort. Making the individual irradiance computations more efficient by using an efficient renderer, such as Lightcuts, can improve the performance of the first pass but still fails to efficiently exploit the relative importance of the samples.

Our scalable, single pass algorithm unifies the evaluation of both the surface irradiance and the subsurface transport. Instead of clustering just the BSSRDF evaluations, our algorithm clusters the evaluations of complete eye-subsurface-light paths. Even though our algorithm does not pre-compute the irradiance, we are still able to reuse intermediate irradiance calculations by introducing a new form factor cache. Thus, our algorithm preserves the two important computational advantages of the two pass algorithm, clustering and reuse. However, our algorithm is fundamentally more scalable because it avoids the brute force irradiance computation that prevents the two pass approach form scaling well to large scenes with complex illumination.

4. Algorithm

Our algorithm for solving Equation 3 for the multiply scattered diffuse radiance L^d builds on Multidimensional Lightcuts. First, we discretize the integration domain using point triples. Next, we use the point samples to convert the integral into a summation. Third, we cluster the triples and estimate the sum efficiently using error bounded approximations of the cluster sums. The first three parts of this section discuss these three parts of our algorithm: point sampling, summation and clustering. The last part of the section describes the final piece of our algorithm, the irradiance form factor cache that makes our algorithm efficient by storing expensive irradiance evaluations between calculations at different exitant surface points.

4.1. Point Sampling

We discretize the domain of Equation 3 using three types of samples: *light samples* $l_i \in \mathbb{L}$, *irradiance samples* $b_j \in \mathbb{B}$ and a single *eye sample* e. The eye sample is the intersection of a camera ray with the translucent object. It represents the location where L^d must be calculated. Light samples discretize

$$
\begin{aligned}
e &= \text{the eye sample} \\
b_j &= j\text{th sample in the irradiance set } \mathbb{B} \\
l_i &= i\text{th sample in the light set } \mathbb{L} \\
E &= T(\eta, \omega_e) \\
B_j &= T_d(\eta)\Delta A(b_j) \\
I_i &= \text{intensity of sample } l_i \\
R_j &= \text{the dipole term, } R(\| e - b_j \|) \\
F_{ji}^{(nv)} &= \tfrac{1}{\pi}(n_j \cdot \omega_{b_j \to l_i}) \\
F_{ji} &= F_{ji}^{(nv)} V(b_j \to l_i) \\
\omega_e &= \text{direction from } e \text{ to camera} \\
\Delta A(b_j) &= \text{surface area represented by } b_j \\
n_j &= \text{surface normal at } b_j \\
\omega_{b_j \to l_i} &= \text{direction from } b_j \text{ to } l_i \\
V(b_j \to l_i) &= \text{visibility of } l_i \text{ at } b_j \\
T_d(\eta) &= -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta
\end{aligned}
$$

Figure 3: Notation used to Section 4. Terms refer to the sample triple (e, b_j, l_i).

the lighting domain and are computed using the pre-process described in [WFA*05]. Irradiance samples are generated uniformly on all translucent surfaces (see Section 5.3). A triple of samples (e, b_j, l_i) represents the radiance leaving light l_i entering the surface at b_j and leaving the surface at e traveling towards the camera, a complete eye-subsurface-light path. Evaluating a triple requires computing the radiance traveling along the two links between the samples. The irradiance link connects the irradiance sample and the light sample, and the BSSRDF link connects the eye sample and the irradiance sample.

4.2. Summation

Throughout this section we will refer to Figure 4. The four different methods of computing L^d discussed in Section 3 are illustrated in each of the 4 subfigures. From left to right, they are: (a) the naïve summation; (b) the two pass algorithm from [JB02]; (c) the two pass algorithm using Lightcuts; and (d) our new unified algorithm. Each subfigure shows the single eye sample (center, blue circle), six irradiance samples (red circles) and four light samples (yellow stars). Irradiance links are drawn as straight black lines and BSSRDF links are curved lines.

To make the discussion of the different summation techniques more concise, we first introduce some shorthand notation (see Figure 3) to simplify their presentation. To evaluate the contribution of a single sample triple (e, b_j, l_i), we in-

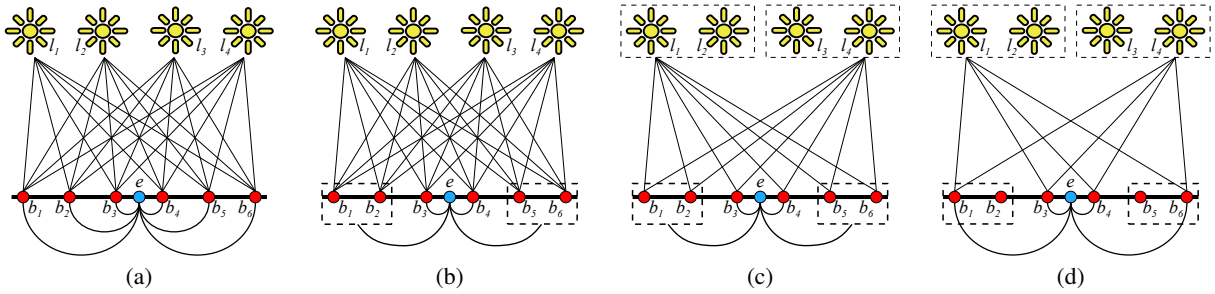


Figure 4: Four different methods of computing L^d from 4 lights points l_i, 6 irradiance points b_j and a single camera point e. (a) Naïve method: all links computed (b) Two pass method: all irradiance links computed, BSSRDF links clustered (c) Two pass method plus Lightcuts: both irradiance and BSSRDF links clustered, all irradiance samples evaluated (d) New method: clustered irradiance and BSSRDF link pairs.

troduce five new terms. With each of the three sample types, we associate a strength that represents the sample's relative importance and, with each of the two link types, we associate a link factor that represents the fraction of light following the link.

The eye sample strength E is the incoming Frensel term $T(\eta,\omega_e)$. The irradiance sample strength B_j is the product of the outgoing Fresnel term at j and the surface area allocated to b_j. Like [JB02] we remove the dependence on the outgoing direction by using the integrated, diffuse Frensel transmittance $T_d(\eta)$ instead of the exact Frensel term. The light sample strength I_i is the intensity of sample l_i. The irradiance link factor F_{ji} is the point to point form factor between the samples b_j and l_i. The BSSRDF link factor R_j is the dipole BSSRDF between e and b_j.

4.2.1. Naïve Summation

Using this notation, the naïve summation (see Figure 4(a)) can be expressed concisely as an inner summation of the contributions of all irradiance links and an outer summation of the contributions of all the BSSRDF links.

$$L^d = E \sum_{j\in\mathbb{B}} R_j \sum_{i\in\mathbb{L}} B_j F_{ji} I_i \tag{7}$$

It is important to emphasize the size of the sum in Equation 7. For a single object, the irradiance sample set $\mathbb{B}$ can contain hundreds of thousands of samples. The larger scenes presented in our results contain millions of irradiance sample points. For complex lighting effects including global illumination, a large light sample set $\mathbb{L}$ is required. In our results we use over 50,000 samples. Since the summation in Equation 7 scales by $O(|\mathbb{B}\times\mathbb{L}|)$, the sum can often contain billions of terms.

4.2.2. Two Pass Summation

The two pass algorithm (see Figure 4(b)) splits the inner and outer sums in the naïve solution. It first computes the integrated irradiance Ψ_j at each irradiance sample. Then it selects a set $\mathbb{S}$ of irradiance sample clusters $\mathbb{C}_\mathbb{B}$ using an octree. It computes the outer summation quickly by using only a single BSSRDF evaluation to the centroid of each cluster in $\mathbb{S}$.

$$1^{st}\text{ pass:}\quad \forall j\in\mathbb{B},\ \Psi_j = \sum_{i\in\mathbb{L}} B_j F_{ji} I_i$$

$$2^{nd}\text{ pass:}\qquad L^d = E R_{centroid} \sum_{\mathbb{C}_\mathbb{B}\in\mathbb{S}} \sum_{j\in\mathbb{C}_\mathbb{B}} \Psi_j$$

Because it reuses the inner sums, the two pass algorithm is significantly less expensive than the naïve summation. Additionally, the clustering used in the second pass reduces the cost of the outer sum from $O(|\mathbb{B}|)$ to $O(|\mathbb{S}|)$. However, the $O(|\mathbb{B}|)$ cost of the first pass is a fundamental limitation.

4.2.3. Two Pass Summation with Lightcuts

Using scalable algorithms to evaluate the irradiance computations required in the two pass algorithm can reduce the cost of the first pass by making each computation more efficient. Lightcuts clusters the light points and reduces each calculation from a sum over all light points to a sum over the representative lights of the set $\mathbb{C}_\mathbb{L}$ of light clusters.

$$\forall j\in\mathbb{B}, \Psi_j = \sum_{i\in\mathbb{C}_\mathbb{L}} B_j F_{ji} I_i^{rep}$$

However, the number of irradiance evaluations required in the first pass does not decrease. For large scenes, the number of calculations outstrips the efficiency that Lightcuts provides leaving the first pass as the bottleneck for the entire algorithm.

4.2.4. New Unified Summation

Our algorithm (see Figure 4(d))—instead of clustering either BSSRDF or irradiance links individually—clusters complete eye-subsurface-light paths. By unifying the two different clustering operations, we can consider the contributions of both the BSSRDF and the irradiance links when partitioning the paths into clusters. By having knowledge about the

complete light path, we are able to choose clusters that more efficiently reflect the total image contribution.

We represent a cluster of eye-subsurface-light paths with a triple $(e, \mathbb{C}_\mathbb{B}, \mathbb{C}_\mathbb{L})$. Here $\mathbb{C}_\mathbb{B}$ and $\mathbb{C}_\mathbb{L}$ are clusters of irradiance and light samples respectively. Given cluster triple, we can estimate its contribution using the BSSRDF and form factor terms for a representative triple of samples (e, b_b, l_l)

$$L^d_\mathbb{C} = ER_bF_{bl}\Big[\sum_{j\in\mathbb{C}_\mathbb{B}} B_j\Big]\Big[\sum_{i\in\mathbb{C}_\mathbb{L}} I_i\Big] \tag{8}$$

By pre-computing the sums of the irradiance and light strengths, this sum costs as much as a single triple evaluation. We evaluate the entire integral by adapting the MDLC cut selection algorithm to cluster the set of sample triples. The final summation reduces to the sum of the contributions of all the triple clusters in the cut.

$$L^d = \sum_{\mathbb{C}\in\text{cut}} L^d_\mathbb{C} \tag{9}$$

4.3. Cut Selection

Like MDLC, we compute a cut by traversing an implicit hierarchy of potential clusters. We build separate binary clusterings of the irradiance and the light samples. To chose a cut we walk the implicit Cartesian product graph of these two binary cluster trees. We start with the triple containing both root clusters and iteratively refine the highest error triple. Refining replaces the triple with two smaller triples formed by replacing either the irradiance cluster or the light cluster by its children from its respective hierarchy. We use a heuristic to chose which triple we refine at each stage (see Section 5.2). During cut selection, we keep a running total of the contributions of all triples currently in the cut. We stop refinement when the error of the highest error triple falls below a fixed fraction (2% for our tests) of the current estimate.

The cut selection algorithm requires an upper bound on the error of Equation 8 compared to the true contribution. Since E and the two sums are constants, only the F_{bl} and the R_b terms require bounding. The form factor term can be bounded using the techniques from [WABG06]. Since the dipole BSSRDF is monotonically decreasing with distance, we can bound the BSSRDF term R_b with the BSSRDF term from the camera point to the closest point in the irradiance cluster's bounding volume.

4.4. Form Factor Cache

The most significant cost of our algorithm is the evaluation of the irradiance link form factors. Each evaluation requires an expensive visibility check between the irradiance sample and light sample. Since these form factors and visibility do not depend on the eye sample, we introduce a form factor cache to store them between evaluations of L^d.

Our first implementation of a cache stored previously computed form factors in a hash table. The key to the table was a hash of the ID's for the two samples the form factor linked. However, the cache is accessed with every triple evaluation and the cost of hashing became a significant expense. We needed the implementation to be as efficient as possible. As an alternative, we built the form factor cache using a tree. A node in the cache represents a cluster triple and has four children. The children correspond to the two pairs of triples that could result from refinement. Each node stores the form factor for the node's representative sample triple. By building the cache in this way, the cache structure mirrors the order that nodes are refined in the triple cut. This avoids the expense of hashing because the cut selection algorithm can simultaneously traverse the cache and the cut explicitly maintaining references from triples in the cut to cache nodes. The algorithm builds the cache lazily. Anytime a cache node is requested, but does not exist, the algorithm creates a new cache node and stores the appropriate form factor within it.

Without an eviction heuristic, the form factor cache would quickly grow prohibitively large. Fortunately, the cache accesses tend to be local. At any time, most of the nodes in the cache store form factors for small cluster triples corresponding to surface regions near the eye sample. Nearby eye samples frequently reuse these form factors, but as the rendering progresses to more distant regions, the form factors for these small cluster triples are no longer needed. At these more distant points, their corresponding surface regions are represented by larger cluster triples in the cache. We experimented with several cache eviction algorithms to remove these stale nodes, but empirically found that none performed better than simply deleting the cache every so often. Our algorithm fixes the total number of nodes allowed in the cache (1,000,000 is a reasonable choice) and simply discards them when the cache is full.

4.5. Summary

In summary our algorithm computes the multiply scattered diffuse subsurface radiance L^d by

1. Generating large fixed sets of irradiance samples $\mathbb{B}$ and light samples $\mathbb{L}$
2. Building two binary trees that hierarchically represent $\mathbb{B}$ and $\mathbb{L}$
3. Clustering eye-subsurface-light paths, represented as cluster triples $(e, \mathbb{C}_\mathbb{B}, \mathbb{C}_\mathbb{L})$, by refining a cut through an implicit hierarchy of potential cluster triples
4. Efficiently computing an bounded-error estimate of the integrated subsurface scattering by using Equation 8 to approximate the contribution of each cluster
5. Caching the form factors for irradiance links between multiple L^d evaluations

5. Implementation Details

In this section, we discuss some implementation details essential to our algorithm.

5.1. Representative Selection

When computing a cluster triple's contribution, it is important to chose the representative sample triple carefully. The MDLC algorithm stores multiple representatives per cluster. Each time a cluster's contribution is evaluated, a new representative is chosen randomly. Random representative selection makes it unlikely that nearby integral evaluations will choose the same representatives. Since our refinement heuristics (see the next section) depend on the representative, fixed representatives can cause sets of nearby pixels to refine the cut in a similar manner. When the refinement pattern changes, the difference can cause aliasing in the final image. However, in our algorithm, by caching the irradiance form factor, we implicitly cache the choice of the representative samples used to compute it. Caching multiple form factors per cluster would be prohibitively expensive and significantly reduce the cache's utility. However, within a representative triple, it is not necessary to use the same irradiance sample to evaluate both the BSSRDF and irradiance links. By using different representatives, we are able to avoid aliasing artifacts by selecting from multiple representatives during BSSRDF link evaluations (our results use 64) while still caching a single representative term in the form factor cache.

5.2. Refinement Heuristic

During cut selection, a choice must be made to either refine the irradiance cluster or light cluster. The ideal choice is the cluster that would ultimately produce the smallest, and thus cheapest, final cut. However, since this cannot easily be determined, a heuristic choice must be made. Our choice is motivated by three factors. First, since the BSSRDF term decreases exponentially with distance, splitting the irradiance cluster tends to most rapidly isolate the small cluster triples near the eye sample with significant contribution. Second, light cluster refinement helps identify the visible light sources. Since the refinement's stopping criteria depends on the current cut estimate, locating contributing triples early helps avoid unnecessary refinement. Third, long chains of the same refinement are poor choices; alternation ensures that a larger space of cluster triples are explored.

Based on these factors, our heuristic applies the following four tests in order. First, if the irradiance cluster contains the camera point, split the irradiance cluster. Second, if the last m consecutive refinements have made the same choice, split the opposite cluster. Third, if

$$F_{ji}^{(nv)} I_i > \alpha * W \tag{10}$$

split the light cluster. Here W is the image white point (the radiance value mapped to white in the output image) and $F_{ji}^{(nv)}$ is the irradiance link form factor without the visibility term. Equation 10 ensures that bright light clusters are subdivided early. Finally, if

$$R_j > \beta * F_{ji}^{(nv)} \tag{11}$$

split gather cluster, otherwise split the light cluster. Equation 11 tries to estimate which link contributes most to the current error and splits it. The values of m, α and β are user defined parameters that adjust the relative importance of the various heuristics. We have found that the values 8, 10 and 1 respectively work well for all of our results.

5.3. Irradiance Sample Generation

Our algorithm assumes that a smooth, uniformly distributed set of surface samples is available along with the model. In [JB02] these samples were computed using an energy based point repulsion algorithm developed by Turk [Tur92]. However, our implementation of that algorithm required several hours to converge for large meshes. Instead we use a simpler algorithm—based on Poisson sampling by dart throwing [Mit87, Mit91]—that produces equal quality images and requires only a few minutes of computation.

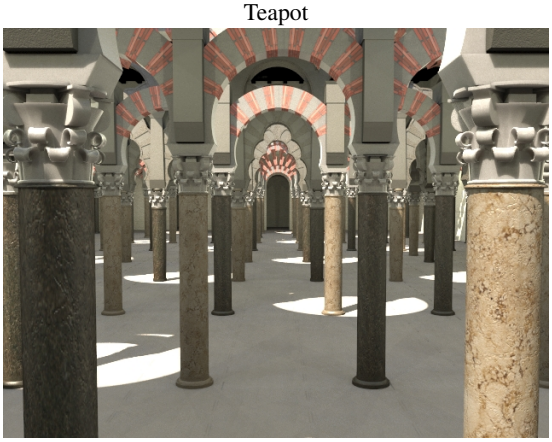
A 2D Poisson sampling is a random sampling satisfying the Poisson condition: no sample can lie within a fixed radius of any other sample. For this application, we use the mean free path of the material as the Poisson radius. Like traditional dart throwing, we generate samples one at a time and discard any sample that violates the Poisson condition. We use 3D Euclidean distance to approximate 2D distance on the mesh surface. However near the end of sample generation, the algorithm can stagnate since previous sample choices might make it impossible to correctly place a new sample. To avoid stagnation, we instead limit the number of candidate choices that can be considered before adding a sample to the set. If the candidate limit is reached, the candidate sample farthest from all previous samples is used even though it might violate the Poisson condition. In practice, the algorithm generates only 20 candidates per sample on average. We allowed up to 10,000 candidates per sample and found that the Poisson condition was never violated during sample generation for our scenes. For our meshes, using this method reduced sample generation cost per mesh to less than a minute.

6. Results

Figure 5 compares the results of our new method and a reference implementation of the two pass method from [JB02]. To make the reference two pass renderer as efficient as possible, we use the Lightcuts algorithm to pre-compute the irradiance during the first pass. From the results in [WFA*05], this reduces the cost of the irradiance pre-computation by at least a factor of 10 over other methods. In both algorithms

Teapot

Kitchen



Cordoba (New)

Cordoba (Reference)

Model	Shadow Rays		Time				SS Cut Size	
	Ref	New	Surface	Ref [Pre]	Ref [Total]	New [Total]	Total	S. Ray
Teapot†	53M	12M	320s	321s	641s	618s	3,589	243
Kitchen†	1,414M	62M	1,719s	5,179s	6,727s	2,679s	3,360	405
Cordoba‡	91,100M	304M	1,092s	50,276s	52,717s	1,258s	6,306	6,220

† Two 3GHz processors ‡ Sixteen 1.7GHz processors

Figure 5: Results for our three test scenes. Table columns from left to right: name of the model; total number shadow rays traced by each method; time separated into the surface reflectance computation, reference pre-computation time, reference total time and new algorithm total time; the average final subsurface triple cut size; and the average number of shadow rays per cut.

the reflected surface component is computed using Multidimensional Lightcuts [WABG06].

All results are listed in seconds for a 640x480 resolution image. To compute the Teapot and Kitchen images we used a dualcore 3GHz Pentium 4 with 2GB RAM. The comparison two-pass image of the Cordoba scene would have taken prohibitively long to render on a single machine. Instead we present the times for both methods as rendered on a cluster of sixteen 1.7GHz Pentium 3s with 1GB of RAM. For all images, each pixel is 32x super-sampled with samples drawn from a quasi-random sequence; however the multiple scattering term is sampled just once per object per pixel. The single scattering term was approximated by a BRDF [HK93] in both algorithms. An initial culling operation ensures that for both algorithms, no pre-computation is performed for any translucent object not visible in the image. In both implementations the evaluation of the diffuse BSSRDF term $R(\| x - x_i \|)$ had a significant cost. However, we found that interpolating $R(\| x - x_i \|)$ from a table of pre-computed values introduced no visible change to the final images and we use this look-up table for all results. We present results for three scenes. Figure 7 gives the number of polygons, irradiance samples and light samples used in each scene and Figure 5 has results and images. In all scenes, the results of our algorithm and the two pass algorithm produce nearly identi-

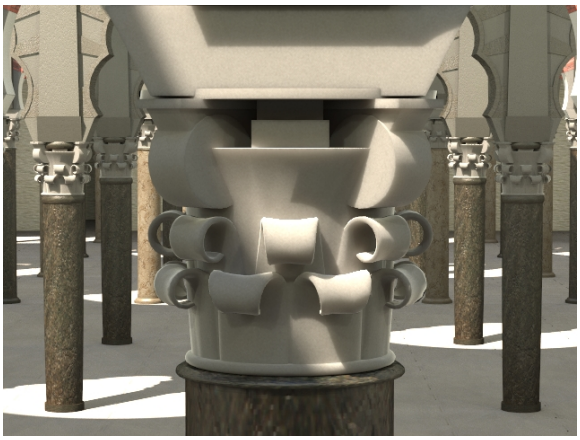


Figure 6: Closeup of the marble capitals in the Mezquita de Cordoba model.

Model	Polygons	Irradiance Samples	Light Samples
Teapot	16,422	173,671	53,128
Kitchen	1,238,126	3,540,428	53,896
Cordoba	2,070,732	64,483,644	53,128

Figure 7: The number of polygons, irradiance samples and light samples for the three scenes in our results.

cal results. The Teapot scene contains a solid teapot made of smooth marble (scattering parameters from [JMLH01]) on a small table lit by an area key light and the Grace Cathedral environment map [Deb02]. The Kitchen scene includes several white and black marble objects on a table lit by several small recessed area lights and a sun/sky model shining through several large windows out of frame. The final scene is a model of the Mezquita de Cordoba. The capitals of each column and every other brick in the upper archways has been rendered using a translucent marble (see Figure 6). The Mezquita model has several large sky lights that pass in sun/sky illumination. The model also includes a grid of 90 point lights positioned near the ceiling. The subsurface marble textures in the Kitchen scenes are applied using the methods described in [JB02]. All scenes contain indirect illumination discretized into 50,000 point lights using the methods from [WABG06]. We note that more indirect lights could have been used at virtually no extra cost. However, we found that with 50,000 lights our algorithm almost never refined the cut to include point triples containing an individual indirect light and adding more indirect lights produced no visible changes to the results.

The Teapot scene is the worst case for our algorithm. It is a small scene with high visibility. In this case, the reference algorithm's brute force irradiance calculation is relatively inexpensive while our algorithm pays a non-trivial overhead for tree construction and cut refinement. However, despite these disadvantages, our algorithm performs comparably with the reference solution. In the larger scenes, the cost of the two pass algorithm quickly becomes dominated by the irradiance pre-computation taking almost 15 hours to render the Cordoba scene even using a sixteen node cluster. This cost is dominated by the large number of shadow ray evaluations required. However, in the larger scenes, the strongly sub-linear performance of our new algorithm becomes a significant advantage. The subsurface cut size, and likewise the cost of computing the multiple scattering term, grows slowly with the scene complexity. Additionally, the form-factor cache further reduces cost by sharing shadow ray evaluations between pixels. Overall, our algorithm reduces the required number of shadow rays, the largest cost of the two pass algorithm, by almost a factor of 300. This savings results in over an order of magnitude reduction in the total rendering cost for the Cordoba model.

7. Conclusion

This paper presents a scalable, unified, single pass algorithm for computing subsurface scattering using the diffusion approximation. By simultaneously estimating the surface irradiance and the subsurface transport, the new algorithm can accurately determine the set of complete eye-subsurface-light paths that contribute to the final image and, unlike previous methods, avoid computation in regions where either the subsurface or surface irradiance is small. This advantage leads to a significant performance improvement in scenes with many translucent objects under complex illumination.

Acknowledgments

We would like to thank our funding agencies and corporations. This work was funded by NSF Career grant 0644175 and the Intel Corporation.

References

[CTW*04] CHEN Y., TONG X., WANG J., LIN S., GUO B., SHUM H.-Y.: Shell texture functions. In *SIGGRAPH '04* (2004), pp. 343–353.

[Deb02] DEBEVEC P.: Image-based lighting. *IEEE Comput. Graph. Appl. 22*, 2 (2002), 26–34.

[DEJ*99] DORSEY J., EDELMAN A., JENSEN H. W., LEGAKIS J., PEDERSEN H. K.: Modeling and rendering of weathered stone. In *SIGGRAPH '99* (1999), pp. 225–234.

[DJ05] DONNER C., JENSEN H. W.: Light diffusion in multi-layered translucent materials. In *SIGGRAPH '05* (2005), pp. 1032–1039.

[DJ07] DONNER C., JENSEN H. W.: Rendering translucent materials using photon diffusion. In *EGRW '07* (2007), pp. 243–251.

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *EGRW '03* (2003), pp. 197–201.

[HBV03] HAO X., BABY T., VARSHNEY A.: Interactive subsurface scattering for translucent meshes. In *SI3D '03* (2003), pp. 75–82.

[HK93] HANRAHAN P., KRUEGER W.: Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93* (1993), pp. 165–174.

[HMBR05] HABER T., MERTENS T., BEKAERT P., REETH F. V.: A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects. In *GI '05* (2005), pp. 79–86.

[HV04] HAO X., VARSHNEY A.: Real-time rendering of translucent meshes. *ACM TOG 23*, 2 (2004), 120–142.

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. In *SIGGRAPH '02* (2002), pp. 576–581.

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *EGRW '96* (1996), pp. 21–30.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *SIGGRAPH '01* (2001), pp. 511–518.

[LPT05] LI H., PELLACINI F., TORRANCE K. E.: A hybrid monte carlo method for accurate and efficient subsurface scattering. In *EGSR '05* (2005), pp. 283–290.

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87* (1987), pp. 65–72.

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distribution ray tracing. In *SIGGRAPH '91* (1991), pp. 157–164.

[MKB*03a] MERTENS T., KAUTZ J., BEKAERT P., REETH F. V., SEIDEL H.-P.: Efficient rendering of local subsurface scattering. In *Pacific Conference on Computer Graphics and Applications* (2003), pp. 51–58.

[MKB*03b] MERTENS T., KAUTZ J., BEKAERT P., SEIDELZ H.-P., REETH F. V.: Interactive rendering of translucent deformable objects. In *EGRW '03* (2003), pp. 130–140.

[NRH*77] NICODEMUS F. E., RICHMOND J. C., HSIA J. J., GINSBERG I. W., LIMPERIS T.: *Geometrical considerations and nomenclature for reflectance*. National Bureau of Standards (US), Oct. 1977.

[PH00] PHARR M., HANRAHAN P.: Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *SIGGRAPH '00* (2000), pp. 75–84.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM TOG 21*, 3 (2002), 527–536.

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. In *SIGGRAPH '05* (2005), pp. 1216–1224.

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *SIGGRAPH '92* (1992), pp. 55–64.

[TWL*05] TONG X., WANG J., LIN S., GUO B., SHUM H.-Y.: Modeling and rendering of quasi-homogeneous materials. In *SIGGRAPH '05* (2005), pp. 1054–1061.

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. In *SIGGRAPH '06* (2006), pp. 1081–1088.

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. In *SIGGRAPH '05* (2005), pp. 1098–1107.

[WTL05] WANG R., TRAN J., LUEBKE D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. In *SIGGRAPH '05* (2005), pp. 1202–1207.