Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

Reduced Aggregate Scattering Operators for Path Tracing

Adrian Blumer[1] Jan Novák[1] Ralf Habel[1] Derek Nowrouzezahrai[2] Wojciech Jarosz[1,3]

[1]Disney Research [2]University of Montreal [3]Dartmouth College

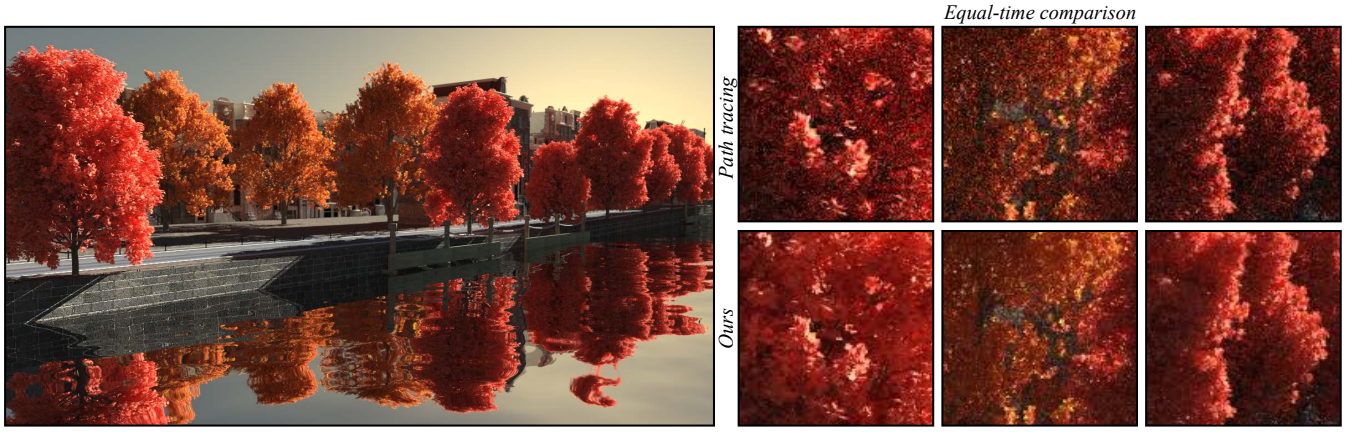


Figure 1: *Our reduced aggregate scattering operators (ASOs) compactly represent per-asset direct and indirect illumination, and the relation between them, in a reduced-dimensional subspace represented by a chain of dense matrices. ASOs can be used in path tracing to reduce the noise due to asset-internal scattering at the cost of a small amount of temporally stable bias.*

Abstract

Aggregate scattering operators (ASOs) describe the overall scattering behavior of an asset (i.e., an object or volume, or collection thereof) accounting for all orders of its internal scattering. We propose a practical way to precompute and compactly store ASOs and demonstrate their ability to accelerate path tracing. Our approach is modular avoiding costly and inflexible scene-dependent precomputation. This is achieved by decoupling light transport within and outside of each asset, and precomputing on a per-asset level. We store the internal transport in a reduced-dimensional subspace tailored to the structure of the asset geometry, its scattering behavior, and typical illumination conditions, allowing the ASOs to maintain good accuracy with modest memory requirements. The precomputed ASO can be reused across all instances of the asset and across multiple scenes. We augment ASOs with functionality enabling multi-bounce importance sampling, fast short-circuiting of complex light paths, and compact caching, while retaining rapid progressive preview rendering. We demonstrate the benefits of our ASOs by efficiently path tracing scenes containing many instances of objects with complex inter-reflections or multiple scattering.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Robust image synthesis in large scenes with complex geometry, realistic reflectance models, and volumetric scattering remains a challenging open problem. The generality and robustness of path-based algorithms [Kaj86] has led to their recent widespread adoption in the animation and special effects industries, where complex scenes are commonplace [KKG*14]. Unfortunately, path-sampling approaches are slow to converge, especially in the presence of glossy transport, complex visibility, and high albedo. These effects contribute significantly to the realism of synthetic images, but they often require simulating *long paths* with many interactions.

Several hybrid techniques isolate individual indirect effects and tailor specialized solutions, leveraging e.g., repetitive structure [ZHRB13], simplified transport configurations [LAM*11, ZW06, JMLH01], or statistical representations [MWM07]. Motivated by these approaches, we also segment costly effects; however, instead of devising solutions specialized to individual effects, we propose a representation that encodes *all internal transport* within an *asset*; here an "asset" refers to a collection of surfaces and/or participating media. Our *aggregate scattering operators* (ASOs) precompute all internal transport of at least two bounces, such as multi-bounce surface reflection from arbitrary spatially varying BRDFs, or volumetric scattering due to any participating medium.

Given some radiometric input representing the incident illumination—defined as the light arriving on the asset from surrounding emitters and scene elements—ASOs map this input to the corresponding internally-scattered radiometric quantity; henceforth referred to as the output. The inputs and outputs can take on different forms, such as scalar or vector irradiance, or incident radiance represented in some basis. While we use directionless scalar quantities (irradiance and fluence) in our implementation, our formulation imposes no constraints on the form of the input and output, so long as we can convert the input to radiance prior to transport, and convert the internally transported radiance to the output format.

In contrast to single-bounce operators [SKS02, HPB06], where multi-bounce transport is synthesized by applying the operator repeatedly, ASOs capture the entirety of the *internal* transport. Even if we ignore directionality at the input and output locations, we still fully capture the arbitrary-directional transport between them. This is more accurate than using traditional one-bounce operators that impose directionless quantities at *every* bounce, not just the first and last.

We devise reduced, low-dimensional bases to represent the input and output quantities; hence the name *reduced* aggregate scattering operator (reduced ASO). In doing so, we tailor the bases to the nature of the multi-bounce transport inside the asset, allowing us to represent internal scattering of light concisely. Since all precomputation is per asset, ASOs naturally support instancing, rigid-transformation, and uniform scale; this scenario is typical in visual effects scene dressing. ASO assets can be combined with "regular" assets in the same scene; we demonstrate the integration on the path tracing algorithm. The ASO permits the path tracer to efficiently evaluate many high-order scattering events without having to trace them explicitly. This enables fast generation of high-quality images of complex scenes at the cost of a few hundreds of MBs of memory per asset, and a few hundreds of KB per instance. In the rest of the paper, we describe the following contributions:

- we derive a spectral, per-asset basis tailored to the internal, multi-bounce transport *and* the set of expected illumination scenarios;
- we impose no constraints on the asset's internal transport (e.g., surface, participating media, and/or subsurface scattering), geometry, nor any of its internal scattering properties;
- we design a novel importance sampling scheme to stochastically evaluate the reduced-dimensional-transport representation; and,
- we analyze the performance and accuracy, and demonstrate the benefits of using ASOs on a complex scene with multiple assets.

2. Related Work

Aggregating scattering solutions. Methods to aggregate complex scattering and avoid expensive render-time simulation have a long history in graphics. Prior work has derived analytic models to express the macroscopic behavior of scattering at finer scales for a variety of phenomena, including rough reflection off surfaces using microfacet theory [TS67], scattering off of hair or cloth fibers [MJC*03, ZW06], or multiple scattering in translucent objects using the diffusion dipole [JMLH01]. Data-driven approaches have also been proposed to tabulate aggregate surface [MPBM03] or volume [PvBM*06] scattering from physical measurements or from offline simulation [WAT92, APS00, MWM07, SML*12, VADWG15]. Pharr and Hanrahan [PH00] formalized the notion of aggregate scattering functions and demonstrated their utility in layer-oriented transport calculations. Jakob et al. [JdJM14] recently proposed a practical approach for coupling these aggregate scattering operators for layered materials. Our data-driven ASOs also encode aggregate scattering, but we can express arbitrary asset-internal transport in a compact representation tailored to the asset. The per-asset precomputation cost is amortized across instances, scenes, and render passes. In effect, our approach provides "canned assets" that know how to sample and transport light within themselves, similarly to the idea of "canned light sources" [HKSS98].

Precomputed light transport. Precomputed radiance transfer (PRT) techniques [SKS02] map incident basis illumination to outgoing radiance (see Lehtinen [Leh07] and Ramamoorthi [Ram09] for an overview). To improve the quality/storage trade-off, Sloan et al. [SHHS03] observed that many transport functions exhibit low-dimensional behavior in local spatial neighborhoods, allowing them to spatially cluster and compress transfer operators. We also benefit from clustering in our approach, but we locally expand the signal into a modified *spectral mesh basis* [WZH07]—a data-driven Fourier basis defined over the geometry—instead of using a predetermined analytical basis (e.g. wavelets [NRH03] or radial basis functions [TS06, WRG*09]). Additionally, we train our basis using a prior distribution of expected direct and indirect lighting signals, as proposed by Loos et al. [LAM*11].

Instead of precomputing functions that capture the entire light transport, "direct-to-indirect" approaches (e.g. Hašan et al. [HPB06]) focus on operators that only map direct to indirect illumination. These techniques can often leverage the smoothness of indirect effects to more efficiently compress the mapping operator. Our ASOs also considers secondary lighting effects; however, we specifically target complex multi-bounce scattering and use low-dimensional operators as sampling distributions in a path tracing context.

Modular approaches. Several works propose a *modular* approach to PRT (e.g. Zhou et al. [ZHL*05] or Sun and Mukherjee [SM06]), where light transport representations are precomputed on some (potentially simplified) subset of the scene. We are inspired by Loos et al.'s "modular radiance transfer" [LAM*11, LNJS12] which precomputes compact transport operators for simple geometric shapes and then dynamically warps them to the actual scene, recomposing the *total* light transport on-the-fly. Zhao et al. [ZHRB13] extended these ideas to volumetric transport in cloth expressed using exemplar voxel blocks. We also decompose the problem of computing full-scene light transport; however, we precompute the transport *directly* (and separately) on scene assets instead of on simplified building blocks. This yields a more appropriate, specialized basis representation for each asset, all without imposing any constraints on the *class* of light transport interactions within (and across) assets.

Many-light and matrix sampling approaches. Matrix interpretations [HPB07, OP11] of many-light methods [DKH*14] formulate the transport between all lights (columns) and shade points (rows) using a transport matrix. These approaches effectively exploit the

reduced rank nature of the matrix by clustering the lights and/or shade points. We also leverage clustering to sparsify the transport matrix, which, in addition to accelerating the evaluation, enables efficient, hierarchical importance sampling. Our technique can be interpreted as approximating matrix vector products using Monte Carlo, which is an active area of research outside of graphics (see Drineas et al. [DKM06] for an introduction).

Importance sampling. Importance sampling (IS) is critical for efficient, sampling-based rendering techniques, such as Monte Carlo path tracing [Kaj86]; Pharr and Humphreys [PH10] provide a comprehensive survey of IS techniques. Most relevant to ours is the work by Lawrence et al. [LRR04] who propose an IS routine for BRDFs, which uses a factored matrix representation, and the so-called "product sampling" approaches [CJAMJ05, JCJ09, CAM08] that directly sample the *product* of (basis approximated) BRDFs and lighting signals. We similarly compress our ASOs using a tailored matrix factorization, but we use this to *both* evaluate and importance sample high-dimensional *multi-bounce* transport. Multi-bounce IS has been recently explored for light transport in volumes [NNDJ12, GKH*13], but only for two bounces at most. Our approach effectively importance samples arbitrary-bounce paths inside an asset at the expense of an asset-dependent precomputation.

Caching light transport. Many Monte Carlo rendering techniques leverage caching [WRC88, KG09, VKv*14] to amortize expensive lighting calculations across several shade points. We also facilitate caching by propagating multi-bounce evaluation results of one pass to all the shade points of the next pass. Our scheme is guaranteed to converge to the reduced-dimensional transport with more samples.

Vegetation rendering. We demonstrate ASOs on scenes with tree assets in several of our results, due to their geometric complexity and common use in scenes with instancing. The method of Geist and Steele [GS08] for rendering large vegetation ecosystems is perhaps most related to our approach. This approach precomputes volumetric lattice-Boltzmann simulations to encode fine-scale tree-internal propagation which can be coupled [SG09] with a similar scene-dependent coarse-scale precomputation for propagation across trees. We similarly decouple asset-internal transport from the remaining transport in the scene, but we do not impose an approximate volumetric transport model and we express ASOs in compact asset-tailored bases. We additionally avoid scene-dependent precomputation by coupling the transport between assets (and regular objects) using Monte Carlo path tracing.

Subsurface scattering. Our work can also be applied to rendering translucent materials, see Gutierrez et al. [GJJD09] for an overview. Wang et al. [WTL05] applied PRT to rendering of translucent objects to achieve fast all-frequency relighting. Most related to ours is the work by Sheng et al. [SSWN14], who represent subsurface scattering using a transport matrix and use a radiosity-style evaluation: the vector of incident illumination is iteratively multiplied by the subsurface-scattering matrix and the diffuse-interreflection matrix. Our approach differs from these in the construction of the matrices as well as in the evaluation: we employ a stochastic evaluation with importance sampling and caching.

3. Reduced Aggregate Scattering Operators

We start with a high level overview of what ASOs represent and how they can be integrated into a path tracer. ASOs aggregate the scattered transport of light inside and between all parts of an object. For example, Figure 2 illustrates a scene with three objects (outlined in blue) whose internal light transport—illustrated by dashed paths—can be represented using ASOs.

We can concisely define the *internally-scattered* transport represented by an ASO using Arvo et al.'s operator notation [ATS94]. ASOs transform some incident quantity (e.g. irradiance) to some output, internally-scattered quantity; this can be expressed as a Neumann series of the product of the scattering $\mathbf{K}$ and propagation $\mathbf{G}$ operators:

$$\mathbf{T} = \sum_{k=0}^{\infty} (\mathbf{GK})^k. \tag{1}$$

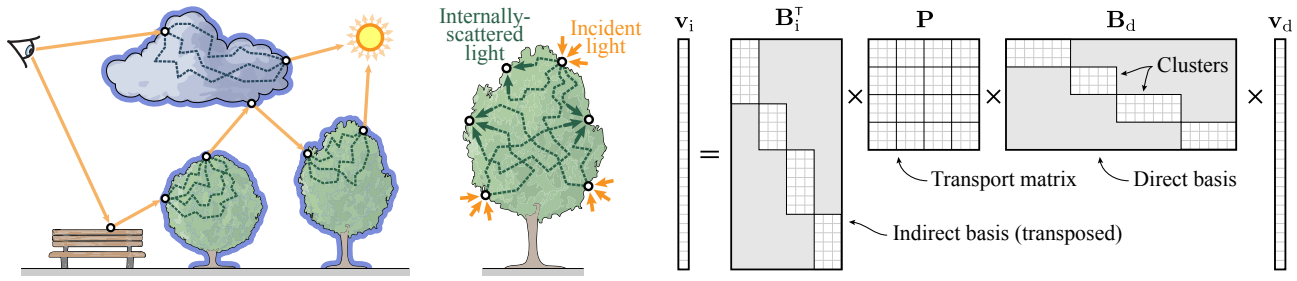
The two operators treat only the transport that involves the objects/volumes inside a single asset.

In theory, one could represent $\mathbf{T}$ with a large square matrix, where each cell $[i, j]$ encodes the transport between the i-th and j-th discretization element (while certain applications may benefit from specific discretizations, e.g. voxels or texels, we henceforth refer to discretization elements as vertices). Unfortunately, the storage requirements for such a matrix would quickly exceed available memory for even a medium-sized assets.

In order to keep memory footprints tractable, we devise two compact bases for expressing the input and output quantities in a low-dimensional space, and a transport operator that relates these input and output signals. We express both bases and the transport operator in matrix form, allowing us to approximate $\mathbf{T}$ as a product of three relatively small matrices.

The first basis $\mathbf{B}_\mathrm{d}$ is used to represent *direct* irradiance, which, in our context, we define as light arriving from luminaires and all surrounding objects in the scene. The second basis $\mathbf{B}_\mathrm{i}$ expresses the corresponding *internal*, multi-bounce irradiance. Both bases are adapted to the expected set of lighting scenarios. The internally scattered light transport—relating the two bases—can then be compactly expressed using a dense but small transport matrix $\mathbf{P}$. Concatenating the three matrices into a single product yields an accurate approximation of the full resolution transport matrix; we refer to this product as the the *reduced* aggregate scattering operator $\tilde{\mathbf{T}} = \mathbf{B}_\mathrm{i}^\top \mathbf{P} \mathbf{B}_\mathrm{d}$ (see Figure 2c for an illustration). Regardless of what quantities are chosen for representing the input and the output, matrix $\tilde{\mathbf{T}}$ always expresses the full internal transport between the input and output discretization elements, which can include e.g. arbitrary glossy BRDFs, mirror reflections, or participating media.

One can imagine using the reduced ASO in the spirit of PRT techniques, i.e. to compute "all-to-all" or "all-to-one" transport: given a vector $\mathbf{v}_\mathrm{d}$ representing the (direct) illumination at vertices, we would first project it into $\mathbf{B}_\mathrm{d}$, then transport it using $\mathbf{P}$, and finally reconstruct a vector of (indirect) illumination values $\mathbf{v}_\mathrm{i}$ from $\mathbf{B}_\mathrm{i}$: $\mathbf{v}_\mathrm{i} = \mathbf{B}_\mathrm{i}^\top \mathbf{P} \mathbf{B}_\mathrm{d} \mathbf{v}_\mathrm{d}$. One drawback of this approach is that it requires having the entire $\mathbf{v}_\mathrm{d}$, making it difficult to integrate into a path tracing algorithm that relies on averaging many relatively cheap samples.



(a) Example scene *(b) Single asset* *(c) Matrix representation*

Figure 2: *An example of a scene (a) where the light transport internal to some object (outlined blue) is represented using ASOs (dashed paths). A closeup of one such object is shown in (b). The ASO is stored as three matrices ($\mathbf{B}_i$, $\mathbf{P}$, and $\mathbf{B}_d$) that transform per-vertex direct illumination $\mathbf{v}_d$ values into a vector of internally scattered light $\mathbf{v}_i$ (c). Matrices $\mathbf{B}_i$ and $\mathbf{B}_d$ store basis functions and are sparse due to clustering.*

We demonstrate that our reduced ASOs can be effectively integrated into path tracing, improving its performance in several ways. In addition to evaluating the internally scattered light transport (described in Section 5), the operator can be used also for importance sampling (Section 5.1): when an incident path hits an asset with an ASO, we will importance sample a new location on the asset from which to continue the path while taking the internal, multi-scattered light transport into account. Furthermore, since the basis expansions are compact, we can progressively accumulate and store coefficient vectors representing irradiance samples computed for each path (Section 6). This allows each path constructed using this process to effectively contribute to *all* the shade points on an asset, and not just the original shade point that spawned the path. This importance sampling and caching yields a tremendous speedup at a moderate storage cost per asset.

4. Per-asset Precomputation

We precompute and associate an ASO with a single scene asset, to be later used during rendering. As such, precomputation is scene (and lighting) independent; however, we tailor the two aforementioned bases to expected lighting scenarios incident on the asset (Section 4.1). In order to efficiently handle large and complex meshes, we also devise a clustering scheme (Section 4.2) to spatially localize each individual basis function. Once the bases are constructed, we use path tracing to build a reduced-dimensional transport matrix that captures the internal multi-bounce transport between pairs of basis functions (Section 4.3).

In order to maintain low memory footprint, our current implementation resorts to direction-less scalar incident and internally-scattered *irradiance*. As such, we approximate light transport at the first and the last path vertex (with respect to the asset). Transport along the intermediate vertices is simulated without any constraint on the type of material (diffuse, glossy, or specular).

4.1. Learning Compact Bases

We first derive *direct* and *indirect* bases to accurately represent irradiance on the asset either coming "directly" from *scene illumination* or "indirectly" via multi-scattering inside the asset, respectively. Our bases are inspired by a spectral mesh basis [WZH07], but with three important differences:

- we employ the lighting prior proposed by Loos et al. [LAM*11] to adapt the bases to the expected lighting scenarios;
- we create separate bases for the direct and transported indirect light on/in an asset, each adapted to their expected signals; and,
- we compute a reduced transport matrix that directly computes all bounces of light—instead of relying on iterated multiplication of a one-bounce operator—allowing us to account for arbitrary, all-frequency internal scattering.

We begin by constructing a "training" set of physically realizable lighting scenarios. Ideally, this set should cover the expected illumination patterns of the asset. For example, given a tree, we use a set of uniformly distributed directional lights that span possible incident directions of sun and sky lighting, but also reflected illumination off the ground. For each lighting scenario i in the training set, we compute a vector $\mathbf{l}_i$ where the j-th entry corresponds to the direct irradiance at the j-th vertex. We organize these vectors as rows of an "observation" matrix $\mathbf{L}$ and perform the singular value decomposition (SVD): $\mathbf{L} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are high-dimensional rotation matrices and Σ is a diagonal matrix of singular values. Keeping only k rows of $\mathbf{V}$ yields an orthogonal, low-dimensional basis $\mathbf{B}$ that can compactly approximate any linear combination of lighting scenarios from our training set.

We can now create distinct bases for direct and indirect (i.e. multi-scattered) irradiance, yielding matrices $\mathbf{B}_d$ and $\mathbf{B}_i$, by retaining only k rows of the matrices $\mathbf{U}_d$ and $\mathbf{U}_i$ obtained from the SVD of direct and indirect irradiance observation matrices $\mathbf{L}_d$ and $\mathbf{L}_i$, respectively. We compute the elements of the observation matrices—the irradiance at individual vertices—using path tracing: we trace **radiance** along rays or paths by gathering energy from light sources directly or after scattering inside the asset, respectively. Figure 3 (top) visualizes a few basis functions (rows) of $\mathbf{B}_i$ for a complex tree asset.

4.2. Clustering

The bases described above have global support over the vertices of the asset, which makes it harder to represent local, high-frequency signals. Similarly to other PRT techniques (e.g. [WZH07]), we first partition each asset into a number of clusters consisting of 2k to 10k vertices using k-means. We then perform the basis construction

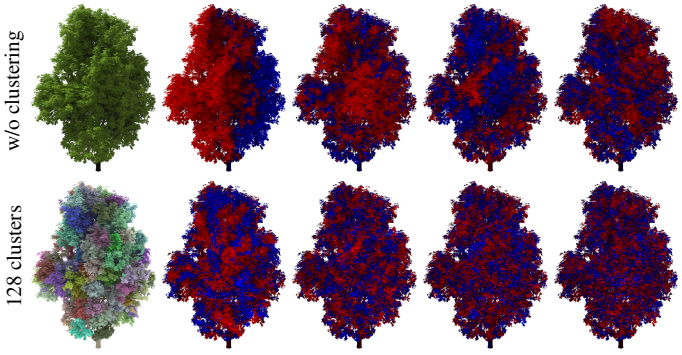


Figure 3: *Visualization of the 2nd, 4th, 8th, and 16th basis function(s) of the indirect basis on a tree without clustering (top row) and with 128 clusters (bottom) visualized in the first column. The clustering helps to capture high-frequency detail. Positive and negative values of each basis function are colored red and blue, respectively.*

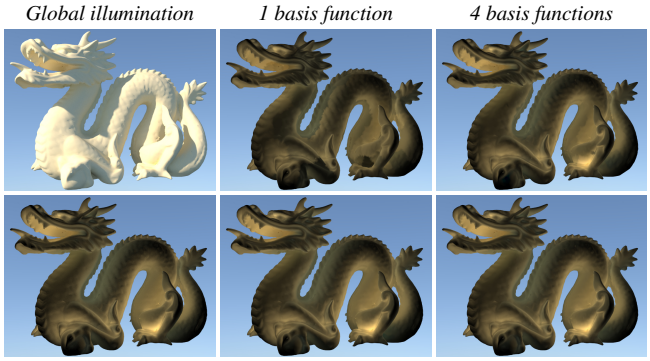


Figure 4: *Quality comparison of the reconstructed indirect illumination using 1, 4, 16, and all rows of* $\mathbf{V}$ *(per cluster) as the orthogonal basis* $\mathbf{B}$*. With low counts, borders between clusters become apparent. All further results use 16 basis functions; a good compromise between quality and storage requirements.*

process as described in the previous subsection but with an SVD computed independently within each cluster. The resulting bases have compact support inside a cluster, and are better suited to accurately representing local details (see bottom row of Figure 3). The basis functions of all clusters can be organized into a single matrix. In contrast to using an equal number of basis functions with global support, the resulting matrices $\mathbf{B}_\mathrm{d}$ and $\mathbf{B}_\mathrm{i}$ are sparse (and band diagonal, see Figure 2 (c)) and can be stored compactly. Figure 4 depicts the accuracy of representing the internally scattered illumination using only the first few basis functions, ordered according to their singular values. When using too few, artifacts at boundaries between individual clusters may arise.

4.3. Transport Matrix

In order to complete the ASO, we need to define the spectral transport operator that relates signals expressed in $\mathbf{B}_\mathrm{d}$ and $\mathbf{B}_\mathrm{i}$. This is often referred to as the direct to indirect transport [HPB06]. Given a coefficient vector $\mathbf{c}_\mathrm{d}$ representing the projection of direct irradiance values $\mathbf{v}_\mathrm{d}$ into the direct basis, i.e. $\mathbf{c}_\mathrm{d} = \mathbf{B}_\mathrm{d}\mathbf{v}_\mathrm{d}$, we seek a matrix $\mathbf{P}$ such that $\mathbf{P}\mathbf{c}_\mathrm{d}$ yields a coefficient vector $\mathbf{c}_\mathrm{i}$ of the corresponding indirect irradiance projected into $\mathbf{B}_\mathrm{i}$.

In order to compute this relationship, we consider the i-th basis function of $\mathbf{B}_\mathrm{d}$ and treat its (potentially signed; see Figure 3 for a visualization) values as irradiance at the corresponding vertices. We then compute full multi-bounce transport inside the asset using path tracing (similarly to Section 4.1) and record the converged, internally scattered (indirect) irradiance at each output vertex; this yields a vector $\mathbf{q}_i$ for each of the M basis functions of $\mathbf{B}_\mathrm{d}$. Next, we organize our observations as rows of a matrix $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ...\mathbf{q}_M\}$ and obtain the transport matrix $\mathbf{P}$ by simply projecting $\mathbf{Q}$ into $\mathbf{B}_\mathrm{i}$, i.e. $\mathbf{P} = \mathbf{B}_\mathrm{i}\,\mathbf{Q}$.

Combining all the terms above, the reduced ASO is represented with the following triple-matrix product:

$$\tilde{\mathbf{T}} = \mathbf{B}_\mathrm{i}^{-1}\mathbf{P}\mathbf{B}_\mathrm{d} \approx \mathbf{T}, \tag{2}$$

where the orthogonality of $\mathbf{B}_\mathrm{i}$ allows expressing $\mathbf{B}_\mathrm{i}^{-1} = \mathbf{B}_\mathrm{i}^\mathsf{T}$.

As mentioned in Section 3, $\tilde{\mathbf{T}}$ can be used to evaluate "all-to-all" and "all-to-one" light transport in the spirit of traditional PRT techniques; Figure 5ab illustrates these evaluation patterns and lists their asymptotic cost with N, M, C being the number of vertices, basis functions, and clusters, respectively. Note that the cost does not scale well with the size of the scene, and the fact that both access patterns require computing the incident illumination at all input vertices prevents efficient integration into a path tracer. Henceforth, our main goal will be to interpret the application of an ASO as a Monte Carlo estimator, and adapt its use to make it suitable for integration into standard path tracing algorithms.

5. Monte Carlo Evaluation of the ASO

In contrast to PRT, Monte Carlo integration relies on averaging many, relatively cheap samples. As such, it is amenable to trivial parallelization and enables fast previews and progressive rendering of large scenes with the error gradually reducing over time. To achieve the aforementioned with ASOs, we note that the matrix product $\tilde{\mathbf{T}}\mathbf{v}_\mathrm{d}$ can be expressed as a sum of the columns of $\tilde{\mathbf{T}}$ weighted by elements of $\mathbf{v}_\mathrm{d}$:

$$\mathbf{v}_\mathrm{i} = \sum_{k=1}^{N} \tilde{\mathbf{T}}[:,k]\,\mathbf{v}_\mathrm{d}[k]. \tag{3}$$

Such formulation, which is sometimes used in many-light rendering algorithms [HPB07], can also be evaluated stochastically: instead of iterating over all elements of $\mathbf{v}_\mathrm{d}$, we evaluate the contribution of only one vertex k chosen according to some probability $p(k)$. This produces the following unbiased Monte Carlo estimator:

$$\langle \mathbf{v}_\mathrm{i} \rangle = \frac{\tilde{\mathbf{T}}[:,k]\,\mathbf{v}_\mathrm{d}[k]}{p(k)}, \tag{4}$$

with the expected value being $\mathbf{v}_\mathrm{i}$. This constitutes a "**one-to-all**" evaluation of the transport matrix, which can be done in $\mathcal{O}(M^2/C + NM/C)$ time with our reduced representation (see Figure 5b for an illustration of the dual pattern).

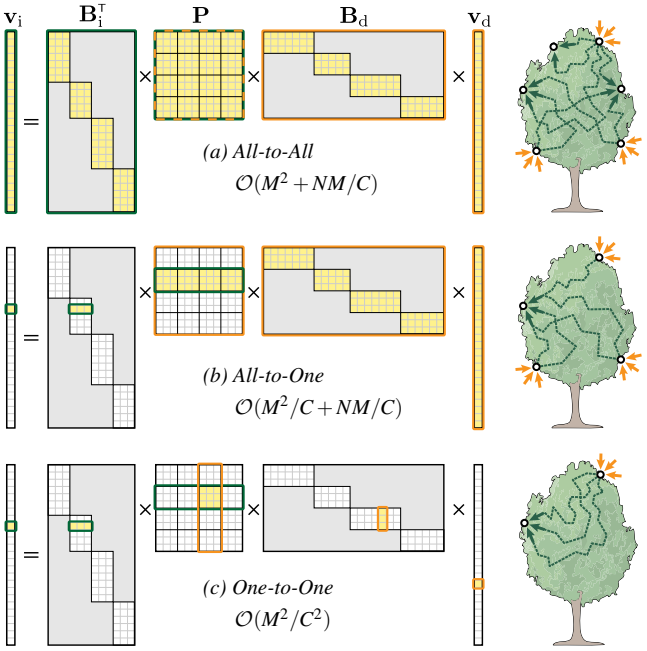


Figure 5: *Different transport evaluation approaches lead to different access patterns on the precomputed data. The matrix elements accessed during the evaluation are colored yellow. The regions of each matrix relevant to the selected elements of* $\mathbf{v}_\mathrm{d}$ *and* $\mathbf{v}_\mathrm{i}$ *are outlined orange and green, respectively.*

We can likewise create a Monte Carlo estimator for just a single element l of indirect light $\mathbf{v}_\mathrm{i}[l]$:

$$\langle \mathbf{v}_\mathrm{i}[l] \rangle = \frac{\tilde{\mathbf{T}}[l,k]\,\mathbf{v}_\mathrm{d}[k]}{p(k)}. \tag{5}$$

This constitutes a "**one-to-one**" evaluation of the transport matrix, which can be done in $\mathcal{O}(M^2/C^2)$ time with our reduced representation (see Figure 5c).

The estimators in Equations (4) and (5) provide us with the flexibility to customize the probability mass function (PMF) $p(k)$ through importance sampling. While we in general do not know $\mathbf{v}_\mathrm{d}$, we do have a stored representation of $\tilde{\mathbf{T}}$, which opens the possibility of performing transport-based importance sampling of this matrix product. In essence, such importance sampling could probabilistically choose incident vertices $\mathbf{v}_\mathrm{d}[k]$ that are more likely to contribute significantly to a particular shading vertex $\mathbf{v}_\mathrm{i}[l]$ based on the transport in $\tilde{\mathbf{T}}$; we discuss such importance sampling next.

5.1. Importance Sampling

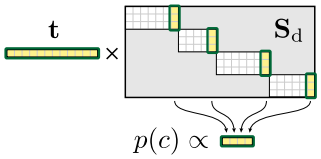
If $\tilde{\mathbf{T}}$ were represented using a *single* matrix, a perfect PMF could be easily derived from its l-th row $\mathbf{r}_l$, which describes the effective contribution of all other vertices. Since storing such matrices is impractical, we need to build $\mathbf{r}_l$ at runtime from the product $\mathbf{B}_\mathrm{i}^\mathsf{T}\mathbf{P}\mathbf{B}_\mathrm{d}$.

Perfect PMF $p_{\tilde{\mathbf{T}}}(k)$. Figure 5b highlights parts of the matrix product that represent the internal transport to a single "shading" vertex l (outlined green in $\mathbf{v}_\mathrm{i}$). To reconstruct $\mathbf{r}_l$, we multiply the l-th row of the transposed indirect basis with the transport matrix yielding an auxiliary vector $\mathbf{t} = \mathbf{B}_\mathrm{i}^\mathsf{T}[l,:]\mathbf{P}$, which is then right-multiplied by matrix $\mathbf{B}_\mathrm{d}$. Since $\tilde{\mathbf{T}}$ is a reduced version of $\mathbf{T}$, $\mathbf{r}_l$ may contain negative values due to ringing. As such, we can make the PMF proportional to the absolute value of $\mathbf{r}_l$, i.e. $p_{\tilde{\mathbf{T}}}(k) \propto |\mathbf{r}_l[k]|$.

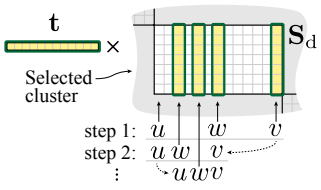
Since $\mathbf{B}_\mathrm{i}$ and $\mathbf{B}_\mathrm{d}$ are sparse, the vector-matrix products can be restricted to the non-zero interval of the l-th row in $\mathbf{B}_\mathrm{i}^\mathsf{T}$ and non-zero column-intervals in $\mathbf{B}_\mathrm{d}$, each containing M/C elements. The total cost of $\mathbf{B}_\mathrm{i}^\mathsf{T}[l,:]\mathbf{P}$ and $\mathbf{t}\mathbf{B}_\mathrm{d}$ thus amounts to $M^2/C+NM/C$ multiply-add operations. Unfortunately, this grows *linearly* with the number of vertices N and makes the sampling prohibitively expensive for practical applications.

Practical PMF $p^\star_{\tilde{\mathbf{T}}}(k)$. To remove the vertex-linear cost, we sidestep computing the product $\mathbf{t}\mathbf{B}_\mathrm{d}$ by first choosing cluster c according to a marginal probability function $p_\mathrm{c}(c)$ and then sampling a vertex within c proportional to a conditional probability function $p_v(k|c)$. In effect, this avoids computing the entire $\mathbf{r}_l$ as we only need its values within the chosen cluster c. We further reduce the sampling cost by precomputing an auxiliary matrix $\mathbf{S}_\mathrm{d}$, which has the same structure as $\mathbf{B}_\mathrm{d}$ but contains prefix sums of rows of $\mathbf{B}_\mathrm{d}$. The prefix sums can be used to efficiently build the PMFs.

The marginal PMF $p_\mathrm{c}(c)$ should be proportional to the *overall* contribution of cluster c to the shading vertex l. The contribution of c can be cheaply computed by dotting $\mathbf{t}$ with the *last* column h of c in $\mathbf{S}_\mathrm{d}$; here we leverage the sums (integrals) of the unweighted basis functions of c stored in $\mathbf{S}_\mathrm{d}[:,h]$. Setting $p(c) \propto |\mathbf{t}\cdot\mathbf{S}_\mathrm{d}[:,h]|$ ensures importance sampling w.r.t. the total contribution of c to l.



The conditional PMF $p_v(k|c)$ should be proportional to the contribution of vertex k to vertex l, i.e. $p_v(k|c) \propto \mathbf{t}\cdot\mathbf{B}_\mathrm{d}[:,k]$. To avoid evaluating these dot products for all vertices within the chosen cluster (this would be still prohibitively expensive), we draw each sample via a binary search in the cumulative distribution function (CDF) $P_v(k|c)$ and build the CDF lazily from $\mathbf{S}_\mathrm{d}$, evaluating dot products only where necessary. Given a random number ξ and an interval of vertices $\langle u,v\rangle$, where u and v are initialized to the indices of the first and last vertex in c, respectively, we split the interval at the middle element w. If $\mathbf{t}\cdot\mathbf{S}_\mathrm{d}[:,w] \le \xi\,\mathbf{t}\cdot\mathbf{S}_\mathrm{d}[:,v]$, then we repeat the splitting with the left half, i.e. $\langle u,w\rangle$, else with the right half, i.e. $\langle w,v\rangle$, until we have only one vertex left; this becomes the selected vertex k.



The lazy evaluation of $P_v(k|c)$ only works if the prefix sums in $\mathbf{S}_\mathrm{d}$ are monotonically increasing. Since basis functions can be negative, this is generally not true. As such, we ensure monotonicity by building the prefix sums over *absolute* values in $\mathbf{B}_\mathrm{d}$. The resulting PMF $p^\star_{\tilde{\mathbf{T}}}(k) = p_\mathrm{c}(c)p_v(k|c)$ is thus not exactly proportional to $\mathbf{r}_l$. However, the benefits of reducing the runtime cost of importance sampling outweigh the resulting imperfection of $p^\star_{\tilde{\mathbf{T}}}$ (see Figure 6).

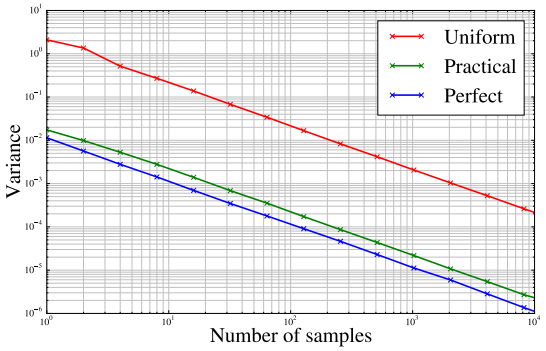
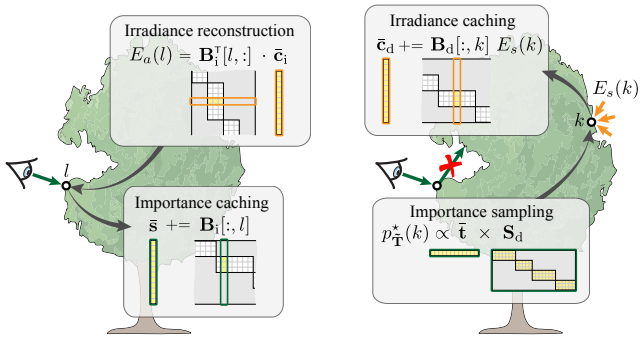


Figure 6: *Average variance of evaluating Equation (5) with perfect, practical, or uniform $p(k)$ on 5 different vertices of the Red Maple tree from Figure 10. While the practical PMF performs slightly worse than the perfect one, it generates samples 302 × faster yielding two orders of magnitude lower "effective" variance.*



(a) Computation upon a path hit *(b) Irradiance cache update*

Figure 7: *Integration of ASOs into a path tracer. When a path hits an ASO (a), we reconstruct the internally-scattered irradiance and cache the importance carried by the path. If the next ray hits the same object (b), we importance sample a vertex using the cached importance, estimate direct irradiance thereof, and accumulate it in $\bar{\mathbf{c}}_d$.*

6. Caching Signals using Spectral Coefficients

The spectral mesh bases $\mathbf{B}_d$ and $\mathbf{B}_i$ provide means to *compactly* represent signals over the entire mesh: the signal can be expressed via a relatively small vector of spectral coefficients of its basis expansion. As such, we can easily cache e.g. irradiance estimates and re-use them across many shading locations.

6.1. Illumination Caching

Suppose we have importance sampled the one-to-one transport, which gives us a contribution to a particular vertex. In many applications, however, we would like the sample to contribute not only to this shading vertex, but also to other potential shading locations. A naive solution is to evaluate the one-to-all transport and store the contributions in the *vertex* basis. This is however fairly costly and requires significant storage per each instance of the asset. We can greatly reduce the memory requirements by caching the contributions expressed by coefficients of the *spectral* basis. Instead of "pushing" the contributions to the vertex basis, we will "pull" it from the spectral coefficients on demand.

Given a direct-irradiance estimate $\mathbf{v}_d[k]$ at a single vertex k, we first expand it into $\mathbf{B}_d$ and accumulate the spectral coefficients $\mathbf{c}_d = \mathbf{B}_d[k,:]\mathbf{v}_d[k]$ in a *mean* spectral-coefficient vector $\bar{\mathbf{c}}_d$. Once $\bar{\mathbf{c}}_d$ contains K such samples, we compute the corresponding internally-scattered illumination expressed in $\mathbf{B}_i$ as $\bar{\mathbf{c}}_i = \mathbf{P}\bar{\mathbf{c}}_d$. The illumination at vertices can be readily reconstructed by computing the sparse product $\mathbf{B}_i^T\bar{\mathbf{c}}_i$. The caching has two benefits: using the spectral basis requires less memory, and caching the samples first in $\bar{\mathbf{c}}_d$ instead of expressing them immediately in $\mathbf{B}_i$ amortizes the cost of multiplying by $\mathbf{P}$ over K samples.

6.2. Importance Caching

An accurate estimation of the internally-scattered lighting may be more important at some points (e.g. those seen by the camera) than elsewhere. In path tracing, this is accounted for implicitly by tracing paths that distribute importance: some parts of the asset are hit more often than others. While the mean distribution of samples is optimal, the PMF of each direct-irradiance sample, in the context of ASOs, is only proportional to its contribution to a *single* shading point—the one that triggered the importance sampling—resulting in high variance. Normally, this can addressed by multiple importance sampling (MIS) [VG95], i.e. by weighting the sample to account for all sampling strategies that could have produced it. The drawback of MIS is that it requires evaluating the sample's density w.r.t. all other strategies. In our case, this amounts to evaluating the sample's PMF w.r.t. all shading points, which is prohibitively expensive.

Instead, we can cache the importance of camera paths and draw samples directly from the overall distribution. The probability of each sample will thus be proportional to its contribution to *all* (importance-weighted) shading points. The caching of importance is analogous to the caching of illumination. Whenever a camera path hits a point on the asset, we find the nearest vertex l and project the path throughput τ—the importance—into $\mathbf{B}_i$ yielding a coefficient vector $\mathbf{s} = \tau\mathbf{B}_i[:,l]$. We accumulate the coefficient vectors in $\bar{\mathbf{s}}$ and once we have a sufficient number of samples (to amortize the cost of right-multiplying by $\mathbf{P}$) we compute a vector $\bar{\mathbf{t}} = \bar{\mathbf{s}}\mathbf{P}$, which is used for importance sampling in place of $\mathbf{t}$ described in Section 5.1.

7. Integration into a Path Tracer

In this section, we connect all the pieces from previous sections and describe how to efficiently integrate ASOs into a path tracer. The changes to the code are local; we only need to adjust how the algorithm estimates scattered radiance and add support for caching irradiance and importance.

Local scattering. The scattered radiance at point x in direction ω_o is defined as:

$$L_o(x,\omega_o) = \int_\Omega f_r(x,\omega_i,\omega_o) L_i(x,\omega_i) \mathrm{d}\omega_i^\perp, \qquad (6)$$

where f_r and L_i are the BRDF and the incident radiance, and we folded the cosine term into the projected solid angle measure. We can express L_i as the sum of radiance L_s arriving from other objects in the scene, and radiance L_a arriving from the asset itself. We approximate

L_a using the internally scattered irradiance E_a captured by the ASO: $L_a \approx E_a/\pi$. The outgoing radiance can then be written as:

$$L_o(x,\omega_o) \approx \alpha(x)E_a(x) + \int_\Omega f_r(x,\omega_i,\omega_o)L_s(x,\omega_i)\mathrm{d}\omega_i^\perp, \quad (7)$$

where α is the albedo of the material.

Figure 7 illustrates the changes to the algorithm. When a path hits an asset with an ASO, we calculate E_a at the hitpoint x as:

$$E_a(x) = \lambda_i \mathbf{B}_\mathrm{i}^\mathsf{T}[i,:]\cdot\bar{\mathbf{c}}_\mathrm{i} + \lambda_j \mathbf{B}_\mathrm{i}^\mathsf{T}[j,:]\cdot\bar{\mathbf{c}}_\mathrm{i} + \lambda_k \mathbf{B}_\mathrm{i}^\mathsf{T}[k,:]\cdot\bar{\mathbf{c}}_\mathrm{i}, \quad (8)$$

where i,j,k are vertices of the triangle containing x, and $\lambda_i,\lambda_j,\lambda_k$ are barycentric coordinates of x within the triangle. The spectral coefficients in $\bar{\mathbf{c}}_\mathrm{i}$ represent the (cached) internally scattered irradiance.

We also need to estimate the reflected light due to other objects in the scene (i.e. the integral in Equation (7)). This part is nearly identical to the original path tracer. We trace a shadow ray to estimate illumination from luminaires and sample the BRDF to continue the path; however, if the path happens to immediately hit the same instance of the asset (see Figure 7b), we terminate it to prevent double counting transport represented by the ASO, and generate a direct-irradiance sample instead to improve the irradiance cache.

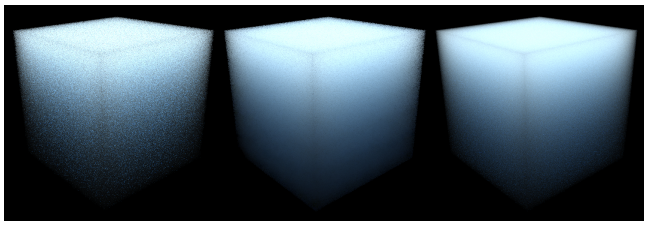
Caching. Importance and irradiance samples are first cached in $\bar{\mathbf{s}}$ and $\bar{\mathbf{c}}_\mathrm{d}$ and propagated to $\bar{\mathbf{t}}$ and $\bar{\mathbf{c}}_\mathrm{i}$ only once per frame—we render the image progressively, one sample per pixel in each frame—to amortize the cost of multiplying by $\mathbf{P}$. The importance is cached whenever a path hits the asset: we compute the corresponding $\mathbf{s}$ and add it to $\bar{\mathbf{s}}$ (see Figure 7a). The irradiance cache, in contrast, is populated lazily. Whenever we terminate a path to prevent double counting, we create a new direct-irradiance sample that will be used to refine the cache. We first importance sample vertex k using $p^\star_{\bar{\mathbf{T}}}(k)$ (which uses the cached importance $\bar{\mathbf{t}}$) and estimate the direct irradiance E_s thereof by tracing a path. The irradiance estimate is then accumulated in the mean spectral coefficient vector $\bar{\mathbf{c}}_\mathrm{d}$.

Implementation details. Matrices $\mathbf{B}_\mathrm{d}$, $\mathbf{B}_\mathrm{i}$, $\mathbf{P}$, and $\mathbf{S}_\mathrm{d}$ are precomputed for each asset. We use Matlab to perform the SVDs and Mitsuba [Jak10] for all path tracing during precomputation and the actual rendering. The importance cache $\bar{\mathbf{t}}$ is initialized to 1 to provide a default, non-zero PDF; all other coefficient vectors are set to zero. We first render a warm-up frame to populate the caches with valid data, which are updated after each subsequent frame as $\bar{\mathbf{t}} \mathrel{+}= \mathbf{P}\bar{\mathbf{s}}$ and $\bar{\mathbf{c}}_\mathrm{i} = \mathbf{P}\bar{\mathbf{c}}_\mathrm{d}$, i.e. importance is accumulated, but irradiance is overwritten to avoid early entries having greater impact than later ones; this would hinder the convergence rate otherwise.

8. Results

In the following, we present the results of applying reduced ASOs to volumetric data sets and unstructured geometry; trees in our examples. In all cases, we construct the training set by illuminating the asset using a directional light source from 42 different directions; which are defined by the vertices of a once-subdivided icosahedron.

Unless stated otherwise, the assets are partitioned into 128 clusters, where each cluster represents the signal using 16 basis functions. We record the direct and indirect *fluence* at voxels, in the case of volumetric datasets, and direct and indirect *irradiance* at vertices, in the case of trees.



PT (10 sec.) *ASO (10 sec.)* *PT (2.5 min.)*

Figure 8: *Adding ASOs to path tracing (PT) can yield faster estimation of multiple scattering in volumes. A 64^3 voxel grid is used as the discretization basis.*

8.1. Volume Rendering

Figure 8 shows renderings of a homogeneous volumetric cube with albedo $\alpha = (0.8, 0.9, 0.95)$. Constructing paths in this extremely simple scene is very fast. Adding an ASO, which operates on a discretized version of 64^3 voxels, increases the cost of drawing samples by a factor of 4. However, this overhead is well justified since each sample approximates many internal light paths reducing the noise more effectively; see the two equal-time renders for comparison.

Figure 9 demonstrates the benefits of using ASOs for subsurface scattering. For simplicity, we assume an index-matched boundary to allow both PT and PT with ASOs perform next-event estimation. The interior volume is isotropic with $\alpha = (0.95, 0.9, 0.8)$. The ASO uses a 128^3 voxel grid; we provide an analysis of the impact of the grid resolution in Section 9. In (a) we show single scattering only, which provides most of the high-frequency detail, but fails to capture the color. The color comes from long, multi-scattered light paths (b) that can be effectively aggregated by the ASO (c). The residual noise in the ASO renderings (c, d) is due to random sampling of free paths along the camera ray; this becomes the main source of variance in this setup. In contrast, the main bottleneck of standard PT (b) is the estimation of light transport due to long paths. Even after 5.5× more time the path-traced reference contains more noise than the ASO image.

Columns (d) to (e) of Figure 9 show an equal-time comparison of path tracing with ASOs to the diffusion dipole [JMLH01]. We used Mitsuba's implementation with default settings, which ignores single scattering and performs hierarchical integration [JB02] over a precomputed set of irradiance cache points. The diffusion approximation produces images with significantly different colors. We manually adjusted the abledo to yield a closer appearance, but the difference (bias) to the reference solution is still much larger than with our ASOs. The main benefit of the dipole approximation over our ASOs is that results are noise-free.

8.2. Unstructured Geometry

We also tested ASOs on unstructured geometry, namely on a number of tree assets in isolation (Figures 10 and 11) and in a larger scene with objects that are not using ASOs (Figure 1). Figure 10 shows *indirect* illumination on three trees in four different lighting configurations. Odd columns present path-traced results utilizing our ASOs. Even columns show reference solutions (i.e. the same path tracer but without ASOs). The top row contains one of the illumination scenarios from the training set. The remaining three rows are

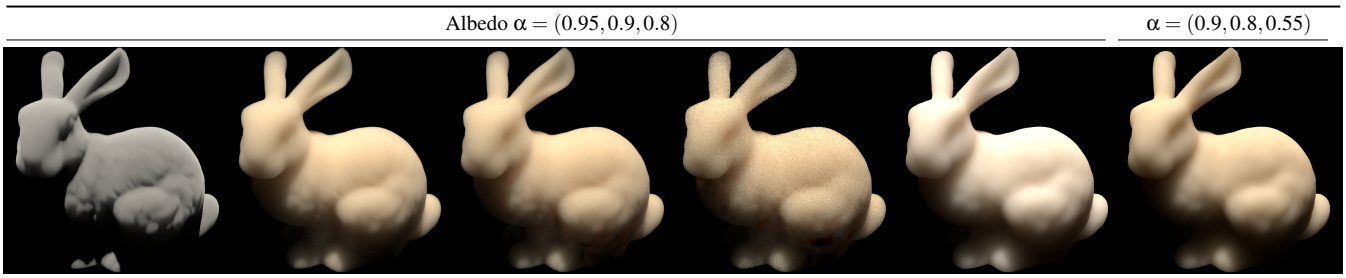


Figure 9: *Sub-surface transport split into single scattering (a) and multiple scattering (b-e). We compare multiple scattering computed using standard path tracing (b), using path tracing with ASOs (c, d), and using the diffusion dipole with the same albedo (f) and with albedo adjusted to produce closer appearance to the reference. Columns (d) to (e) provide an equal-time comparison.*

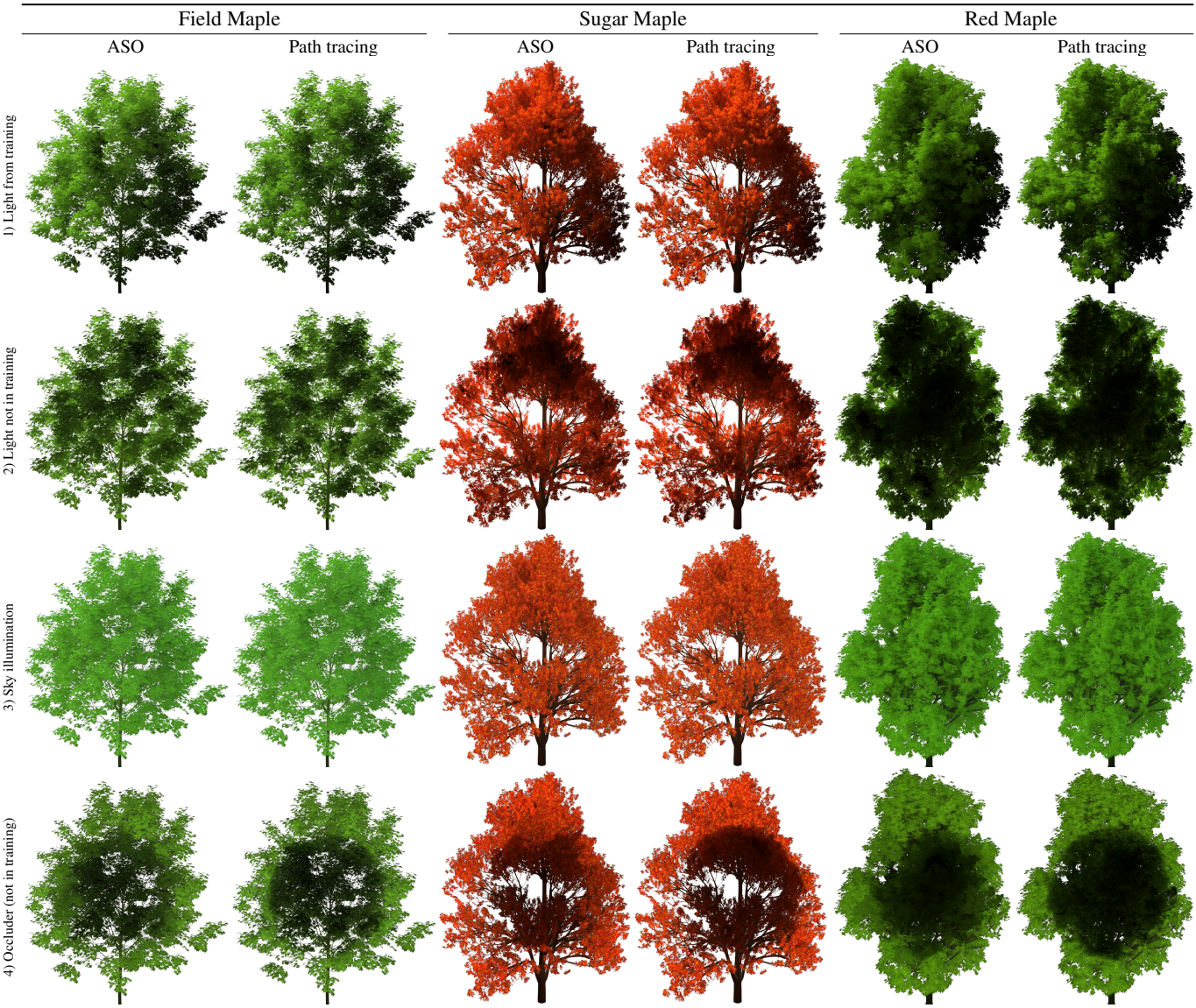


Figure 10: *We compare indirect illumination, computed using reduced ASOs with caching and 16 spp, to converged path tracing solutions on three different tree assets and four different lighting configurations: 1) a directional light from the training set, 2) a directional back light not in the training set, 3) sky illumination, and 4) a directional light with an occluder not in the training set. While there is some blurring, ASOs match the overall appearance of path tracing well. Refer to Figure 11 for insets.*

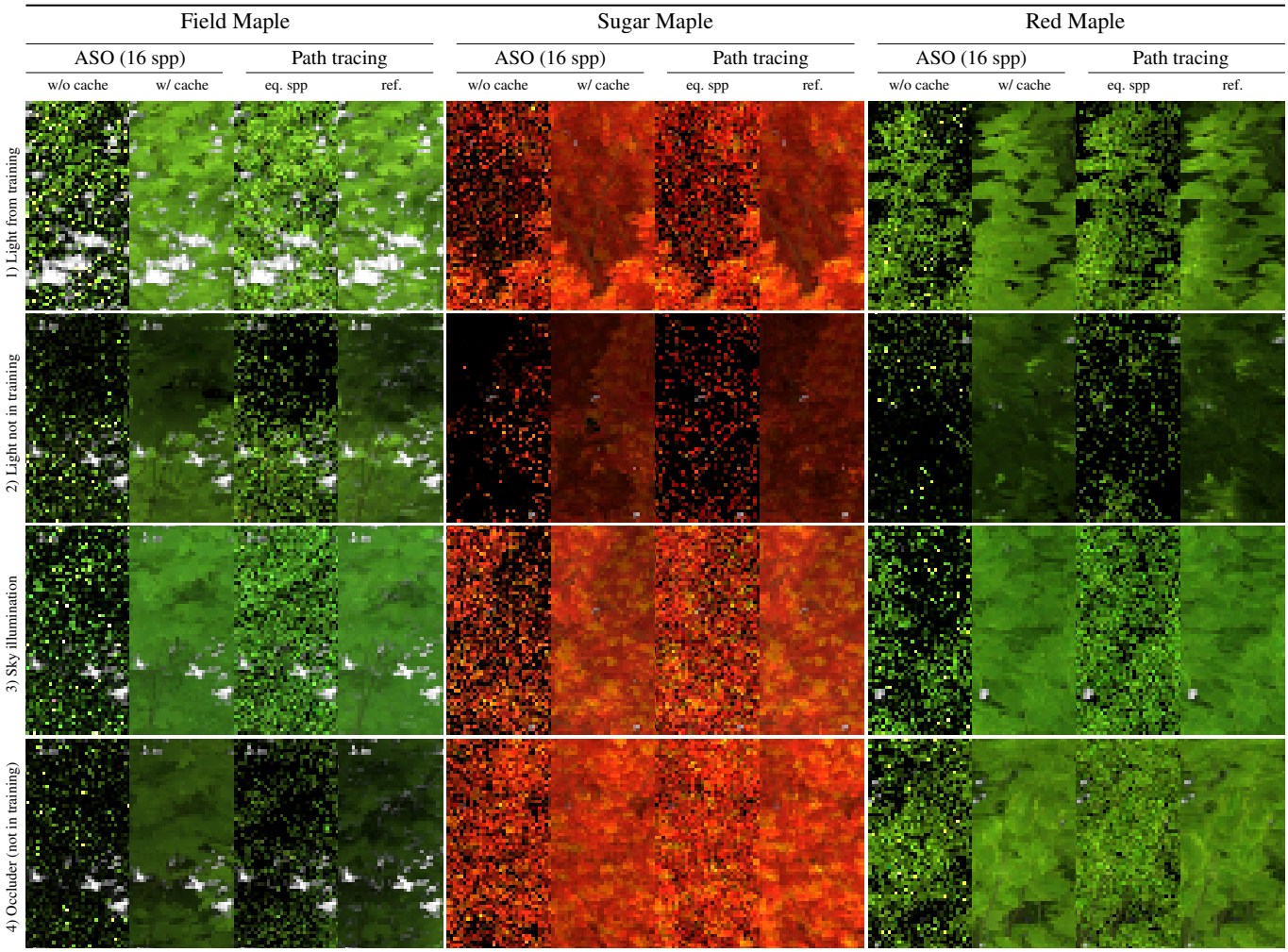


Figure 11: *We compare insets for the tree assets and lighting conditions from Figure 10 computed using ASOs at 16 spp without and with caching to path tracing using 16 spp and at reference quality. ASOs without caching do not improve upon standard path tracing, but with caching enabled, the results using just 16spp are visually close to the path-tracing reference; please refer to Table 1 for error statistics.*

		Field Maple		Sugar Maple		Red Maple	
		ASO	PT	ASO	PT	ASO	PT
1)	Var.	3.18e−5	8.92e−4	2.82e−5	6.05e−4	2.04e−5	6.59e−4
	Bias^2	1.65e−4	-	9.30e−5	-	1.28e−4	-
	Err.	6.67e−4	1.87e−3	7.51e−4	2.71e−3	7.06e−4	2.41e−3
2)	Var.	7.89e−6	2.84e−4	7.79e−6	1.85e−4	3.60e−6	1.16e−4
	Bias^2	8.42e−5	-	4.22e−5	-	1.77e−5	-
	Err.	3.33e−4	6.76e−4	3.10e−4	8.33e−4	1.01e−4	4.41e−4
3)	Var.	7.68e−6	4.27e−4	5.71e−6	1.68e−4	3.88e−6	1.72e−4
	Bias^2	1.53e−5	-	1.20e−5	-	9.05e−6	-
	Err.	8.55e−5	1.10e−3	1.12e−4	8.14e−4	6.37e−5	7.11e−4
4)	Var.	1.31e−5	4.05e−4	2.17e−5	4.27e−4	1.84e−5	5.49e−4
	Bias^2	1.53e−4	-	1.85e−4	-	2.03e−4	-
	Err.	6.06e−4	9.32e−4	1.32e−3	1.92e−3	1.06e−3	1.95e−3

Table 1: *Variance, bias^2, and effective squared error, i.e. render time × (variance + bias^2) for the trees and lighting configurations in Figure 10.*

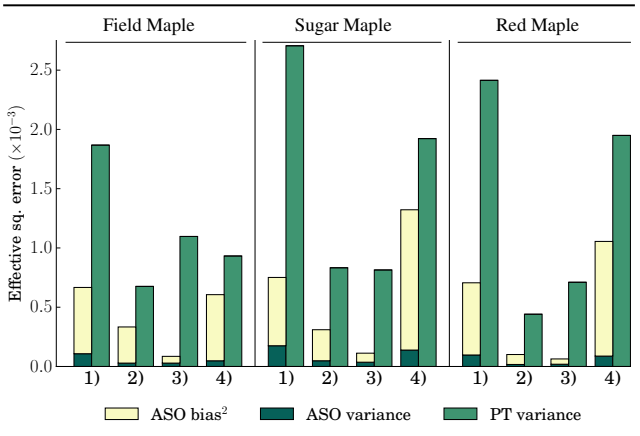


Figure 12: *Effective squared error for the trees and lighting configurations in Figure 10. Left and right bars of each pair correspond to path tracing with and without ASOs, respectively.*

arbitrary setups: back lighting, sky illumination, and front lighting with a spherical occluder (hidden from the camera). The last one is particularly tricky as none of the training scenarios simulate such a configuration. As such, the indirect illumination is synthesized only approximately, but the sparse basis functions that arise from clustering still allow for a reasonable reconstruction of the occlusion effect. The result could be improved by including a similar lighting configuration in the training set.

Figure 11 shows insets for images obtained with 16 samples per pixel, and the ground-truth reference. The layout is the same as in Figure 10, but we additionally evaluate our technique with and without caching. ASOs without caching do not provide much benefit, but with caching enabled, our results at 16 spp are both visually and objectively (in terms of MSE) closer to the ground truth than equal-sample-count path tracing. This comes at the expense of introducing bias and a slightly higher per-sample cost induced by updating and querying the irradiance and importance caches. The caches are very compact, requiring in all cases 128 KB (per instance).

Table 1 shows various statistics for 16 spp versions of images in Figure 10. The table has an analogous layout with lighting scenarios in rows and individual tree models represented by columns. When using ASOs, the rendering time increases by 20% to 30%. We take this into account by reporting the *effective squared error*, i.e. MSE scaled by the rendering time. For images rendered using ASOs, the MSE consists of not only the variance, but also bias. Figure 12 provides a visual comparison of all the metrics. Note how the error of ASOs is dominated by bias, which represents 82% of the MSE on average. Only 18% of the error is due to variance. In fact, the cost-weighted variance of path tracing with ASOs is on average $21\times$ lower than the cost-weighted variance of standard path tracing. This makes rendering the tree assets with ASOs nearly "noise-free" at the cost of a small amount of temporally stable bias.

Figure 1 shows a number of trees illuminated by a setting sun. The converged image on the left was rendered using our ASOs, the insets compare results obtained without (top) and with (bottom) our ASOs in 2.3 minutes. Despite some remaining noise in the ASO render, e.g. due to sampling illumination from the sky or indirect illumination from other objects in the scene, the results look significantly better than those obtained with standard path tracing.

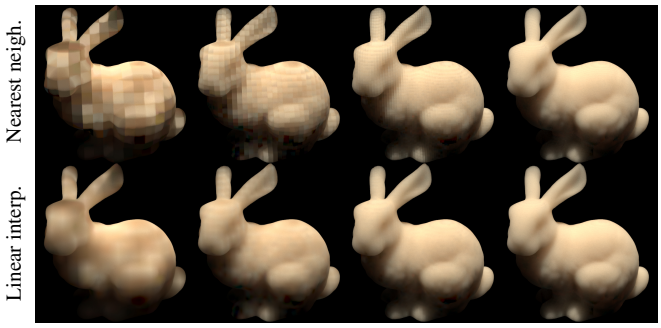
9. Analysis and Discussion

In the following, we review some of the key properties of ASOs and discuss the limitations of our current implementation.

9.1. Systematic Errors

As shown in Section 8, ASOs represent the internal light transport only approximately. The error will be somewhat proportional to how well the current illumination can be represented by a linear combination of the lighting scenarios from the training set. Furthermore, there are the following three sources of systematic errors.

Discretization. Error is introduced by parameterizing the direct and indirect quantities using a finite set of discretization elements (i.e. vertices or voxels). Figure 13 demonstrates the impact of the voxel grid resolution on visual fidelity: coarse grids with large voxels do not provide sufficient resolution and the interpolation around the boundary occasionally yields darker results. The problem diminishes as we increase the resolution, which, however, increases the precomputation and storage cost. Adaptive grids would be an efficient way of mitigating the problem.



(a) $16\times16\times16$ (b) $32\times32\times32$ (c) $64\times64\times64$ (d) $128\times128\times128$

Figure 13: *The resolution of the discretization basis impacts the visual quality. We include also results with nearest-neighbor interpolation (top row) for a better illustration of the resolution of the grid.*

Basis Truncation. Truncating the direct and indirect bases also introduces error as illustrated in Figure 4. In all our results, we keep 16 basis functions with the highest singular values in each cluster. Adapting the number to the signal being represented would be a trivial extension, and would likely strike a better trade-off between storage/computation and accuracy.

Stochastic Training. The direct and indirect quantities, as well as the relation between them, is estimated using Monte Carlo path tracing (in precomputation), which will always suffer from some residual variance. The noise is acceptable when "learning" the two bases: the bases will be slightly suboptimal in the worst case, but they can still represent the signal. However, the residual noise of estimating the internal transport is "baked" into the transport matrix, and can possibly impair the quality. We used 2k to 4k path samples per discretization element in all our examples that sufficiently reduced corresponding artifacts.

9.2. Precomputation and Storage Cost

The precomputation times in our current, unoptimized implementation were on the order of thousands of core hours (e.g. 1340 core hours for the Field Maple tree); we used a cluster to accelerate the precomputation. The majority of computation (97% for Field Maple) is spent on estimating indirect illumination at all tree vertices for all of the lighting scenarios. The memory footprint of the precomputed data, summarized in Table 2, depends on the discretization, but typically requires hundreds of MB per asset at 32-bit precision. Caching, however, necessitates only hundreds of KB per instance. The precomputation time and storage could be further reduced by adaptive sampling (e.g. using confidence intervals), reuse of computation, coarsening of the vertex/voxel bases, or reducing the number

			Memory per asset		per inst.
Asset	Discretization	#BF	$\mathbf{B}_d + \mathbf{B}_i$	**P**	Caches
Field Maple	400k vertices	16	2×145 MB	49 MB	128 KB
Sugar Maple	2500k vertices	16	2×991 MB	49 MB	128 KB
Red Maple	820k vertices	16	2×286 MB	49 MB	128 KB
Bunny	32^3 = 33k voxels	16	2×16 MB	49 MB	128 KB
Bunny	64^3 = 262k voxels	42	2×132 MB	333 MB	337 KB

Table 2: *Memory footprint of ASOs for different assets. Every asset is partitioned into 128 clusters. The third column shows the number of basis functions (#BF) in each cluster.*

of lighting scenarios; we used a relatively dense set of 42 directions to better isolate the errors due to discretization and basis truncation. Practical implementations might however still require several (tens of) core hours for the precomputation.

9.3. Temporal Stability

A key strength of the proposed approach is that the errors, described previously, are temporally stable. We tested this by rendering multiple independent frames of the same scene, and a turn-table animation; please see the accompanying video. It was sufficient to evaluate only 16 spp to resolve all temporal artifacts in both cases. While this may not generalize to all possible applications, given that PT generally requires many more samples, e.g. to capture distribution effects, using ASOs in animations is unlikely to produce flickering.

10. Future Work

All-frequency Transport. One of the main limitations of our current implementation is that it only considers the zero-th moments (irradiance or fluence) of the direct and indirect quantities. As such, directional information at the first and the last bounce is filtered out. One could expand the direct and indirect signals into e.g. spherical harmonics and represent the quantities using a set of coefficients. This would preserve directionality and better handle glossy surfaces and anisotropic phase functions. Keeping the memory footprint low in such case would however require additional research.

Clouds. One use case that would greatly benefit from ASOs is rendering of clouds. The appearance of a cloud is dominated by multiple scattering involving constructions of extremely long paths. Our initial investigations revealed a true need for adaptive discretization, otherwise the interpolation occasionally produces artificially dark cloud boundaries; we plan to investigate this further.

Parametric ASOs. Precomputing the *full* transport inside the asset prevents modifying any portion of it, such as the scattering functions on surfaces or in the volume, without necessitating another round of ASO precomputation. As such, our ASOs can be viewed as creating a "canned asset" that knows how to transport light within itself, but cannot be easily modified. Lifting this constraint by factoring out relevant components of the transport is an interesting research avenue.

11. Conclusion

We presented reduced aggregate scattering operators, a compact way to express, importance sample, and cache asset-internal scattering. We incorporated ASOs into a progressive path tracer as a means to short-circuit long light paths. The technique converges to the full *reduced* transport with a few samples and facilitates easy coupling with other scene assets. In a sense, our application of ASOs forms a bridge between traditional path tracing and precomputation based techniques like PRT, allowing us to harness the benefits of both, seemingly incompatible techniques. We hope our work will stimulate further research in the direction of importing advanced PRT techniques into path tracing frameworks.

Acknowledgements

We thank Skaldy for the model of the street in Figure 1; obtained via blendswap.com, and the Stanford Computer Graphics Laboratory for the Dragon and Bunny models in Figures 4, 9, and 13. The tree models were created using Plant Kit 1 by Laubwerk GmbH.

References

[APS00] ASHIKHMIN M., PREMOZE S., SHIRLEY P. S.: A microfacet-based BRDF generator. In *Annual Conference Series (Proc. SIGGRAPH)* (July 2000), pp. 65–74. 2

[ATS94] ARVO J., TORRANCE K., SMITS B.: A framework for the analysis of error in global illumination algorithms. In *Annual Conference Series (Proc. SIGGRAPH)* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 75–84. 3

[CAM08] CLARBERG P., AKENINE-MÖLLER T.: Practical Product Importance Sampling for Direct Illumination. *Comp. Graph. Forum (Proc. EG Symposium on Rendering) 27*, 2 (2008), 681–690. 3

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: Efficiently evaluating products of complex functions. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (Aug. 2005), 1166–1175. 3

[DKH*14] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable realistic rendering with many-light methods. 88–104. 2

[DKM06] DRINEAS P., KANNAN R., MAHONEY M. W.: Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing 36*, 1 (2006), 132–157. 3

[GJJD09] GUTIERREZ D., JENSEN H. W., JAROSZ W., DONNER C.: Scattering. In *ACM SIGGRAPH Asia Courses* (New York, NY, USA, 2009), ACM, pp. 15:1–15:620. 3

[GKH*13] GEORGIEV I., KŘIVÁNEK J., HACHISUKA T., NOWROUZEZAHRAI D., JAROSZ W.: Joint importance sampling of low-order volumetric scattering. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 6 (Nov. 2013), 1–14. 3

[GS08] GEIST R., STEELE J.: A lighting model for fast rendering of forest ecosystems. In *IEEE Symposium on Interactive Ray Tracing* (2008), pp. 99–106. 3

[HKSS98] HEIDRICH W., KAUTZ J., SLUSALLEK P., SEIDEL H.-P.: Canned lightsources. In *Rendering Techniques (Proc. EG Workshop on Rendering)* (1998), Drettakis G., Max N., (Eds.), pp. 293–300. 2

[HPB06] HAŠAN M., PELLACINI F., BALA K.: Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (July 2006), 1089–1097. 2, 5

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (July 2007). 2, 5

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 8

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 3 (July 2002), 576–581. 8

[JCJ09] JAROSZ W., CARR N. A., JENSEN H. W.: Importance sampling spherical harmonics. *Comp. Graph. Forum (Proc. EG Symposium on Rendering) 28*, 2 (Apr. 2009), 577–586. 3

[JdJM14] JAKOB W., D'EON E., JAKOB O., MARSCHNER S.: A comprehensive framework for rendering layered materials. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (2014). 2

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. *Annual Conference Series (Proc. SIGGRAPH) 35* (2001), 511–518. 1, 2, 8

[Kaj86] KAJIYA J. T.: The rendering equation. *Computer Graphics (Proc. SIGGRAPH) 20*, 4 (Aug. 1986), 143–150. 1, 3

[KG09] KŘIVÁNEK J., GAUTRON P.: Practical global illumination with irradiance caching. *Synthesis Lectures on Computer Graphics and Animation 4*, 1 (2009), 1–148. 3

[KKG*14] KŘIVÁNEK J., KELLER A., GEORGIEV I., KAPLANYAN A., FAJARDO M., MEYER M., NAHMIAS J.-D., KARLÍK O., CANADA J.: Recent advances in light transport simulation: Theory and practice. In *ACM SIGGRAPH Courses* (New York, NY, USA, 2014), ACM. 1

[LAM*11] LOOS B. J., ANTANI L., MITCHELL K., NOWROUZEZAHRAI D., JAROSZ W., SLOAN P.-P.: Modular radiance transfer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 30*, 6 (Dec. 2011). 1, 2, 4

[Leh07] LEHTINEN J.: A framework for precomputed and captured light transport. *ACM Trans. Graph. 26*, 4 (Oct. 2007). 2

[LNJS12] LOOS B. J., NOWROUZEZAHRAI D., JAROSZ W., SLOAN P.-P.: Delta radiance transfer. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)* (2012). 2

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient BRDF importance sampling using a factored representation. *ACM Trans. Graph. (Proc. SIGGRAPH)* (Aug. 2004). 3

[MJC*03] MARSCHNER S. R., JENSEN H. W., CAMMARANO M., WORLEY S., HANRAHAN P.: Light scattering from human hair fibers. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (July 2003), 780–791. 2

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (July 2003), 759–769. 2

[MWM07] MOON J. T., WALTER B., MARSCHNER S. R.: Rendering discrete random media using precomputed scattering solutions. In *Rendering Techniques (Proc. EG Symposium on Rendering)* (June 2007), pp. 231–242. 1, 2

[NNDJ12] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (July 2012). 3

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (July 2003), 376–381. 2

[OP11] OU J., PELLACINI F.: Lightslice: Matrix slice sampling for the many-lights problem. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 30*, 6 (Dec. 2011), 179:1–179:8. 2

[PH00] PHARR M., HANRAHAN P. M.: Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *Annual Conference Series (Proc. SIGGRAPH)* (July 2000), pp. 75–84. 2

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering, Second Edition: From Theory to Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., 2010. 3

[PvBM*06] PEERS P., VOM BERGE K., MATUSIK W., RAMAMOORTHI R., LAWRENCE J., RUSINKIEWICZ S., DUTRÉ P.: A compact factored representation of heterogeneous subsurface scattering. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (July 2006), 746–753. 2

[Ram09] RAMAMOORTHI R.: *Precomputation-Based Rendering*. NOW Publishers Inc, 2009. 2

[SG09] STEELE J. E., GEIST R.: *Relighting forest ecosystems*. Springer, 2009, pp. 55–66. 3

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (July 2003), 382–391. 2

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Trans. Graph. (Proc. SIGGRAPH)* (2002), pp. 527–536. 2

[SM06] SUN W., MUKHERJEE A.: Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (July 2006), 955–966. 2

[SML*12] SADEGHI I., MUÑOZ A., LAVEN P., JAROSZ W., SERON F., GUTIERREZ D., JENSEN H. W.: Physically-based simulation of rainbows. *ACM Trans. Graph. 31*, 1 (Feb. 2012), 3:1–3:12. 2

[SSWN14] SHENG Y., SHI Y., WANG L., NARASIMHAN S. G.: Translucent radiosity: Efficiently combining diffuse inter-reflection and subsurface scattering. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 20*, 7 (July 2014), 1009–1021. 3

[TS67] TORRANCE K. E., SPARROW E. M.: Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America 57*, 9 (Sept. 1967), 1105–1112. 2

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (July 2006), 967–976. 2

[VADWG15] VELÁZQUEZ-ARMENDÁRIZ E., DONG Z., WALTER B., GREENBERG D. P.: Complex luminaires: Illumination and appearance rendering. *ACM Trans. Graph. 34*, 3 (May 2015), 26:1–26:15. 2

[VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for monte carlo rendering. In *Annual Conference Series (Proc. SIGGRAPH)* (New York, NY, USA, 1995), ACM, pp. 419–428. 7

[VKv*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (July 2014), 101:1–101:11. 3

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. *Computer Graphics (Proc. SIGGRAPH) 26*, 2 (July 1992), 255–264. 2

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. *Computer Graphics (Proc. SIGGRAPH) 22*, 4 (1988), 85–92. 3

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 5 (Dec. 2009), 133:1–133:10. 2

[WTL05] WANG R., TRAN J., LUEBKE D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (July 2005), 1202–1207. 3

[WZH07] WANG R., ZHU J., HUMPHREYS G.: Precomputed radiance transfer for real-time indirect lighting using a spectral mesh basis. In *Rendering Techniques (Proc. EG Symposium on Rendering)* (2007), pp. 13–21. 2, 4

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (July 2005), 1196–1201. 2

[ZHRB13] ZHAO S., HAŠAN M., RAMAMOORTHI R., BALA K.: Modular flux transfer: Efficient rendering of high-resolution volumes with repeated structures. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4 (July 2013), 131:1–131:12. 1, 2

[ZW06] ZINKE A., WEBER A.: Global illumination for fiber based geometries. In *Proc. Ibero-American Symposium in Computer Graphics (SIACG)* (2006). 1, 2