Eurographics Symposium on Rendering 2022
A. Ghosh and L.-Y. Wei
(Guest Editors)

Volume 41 (2022), Number 4

A bidirectional formulation for Walk on Spheres

Yang Qi[1] Dario Seyb[1] Benedikt Bitterli[1,2] Wojciech Jarosz[1]

[1]Dartmouth College [2]NVIDIA

Abstract

Numerically solving partial differential equations (PDEs) is central to many applications in computer graphics and scientific modeling. Conventional methods for solving PDEs often need to discretize the space first, making them less efficient for complex geometry. Unlike conventional methods, the walk on spheres (WoS) algorithm recently introduced to graphics is a grid-free Monte Carlo method that can provide numerical solutions of Poisson equations without discretizing space. We draw analogies between WoS and classical rendering algorithms, and find that the WoS algorithm is conceptually equivalent to forward path tracing. Inspired by similar approaches in light transport, we propose a novel WoS reformulation that operates in the reverse direction, starting at source points and estimating the Green's function at "sensor" points. Implementations of this algorithm show improvement over classical WoS in solving Poisson equation with sparse sources. Our approach opens exciting avenues for future algorithms for PDE estimation which, analogous to light transport, connect WoS walks starting from sensors and sources and combine different strategies for robust solution algorithms in all cases.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing;*** *Modeling and simulation;* • ***Mathematics of computing*** → *Stochastic processes;*

1. Introduction

Monte Carlo methods have been very successful in rendering. They provide accurate solution estimates to the rendering equation [Kaj86] in very complex scenes with an often simple implementation compared to mesh based methods such as radiosity [CW93; GTGB84]. While initially Monte Carlo methods were comparatively slow and of mostly academic interest, they now form the predominant rendering methodology in movie production [CJ16; FHF*17] and increasingly in interactive applications such as games.

Catalyzed by the recent introduction of the walk-on-spheres (WoS) algorithm [Mul56] to graphics [SC20; SSJC22], a similar development is now happening for numerical solvers of partial differential equations (PDEs). Analogous to path tracing in rendering, WoS uses random walks to compute point estimates of the solution of harmonic PDEs [Eva10]. These equations are of great importance in many areas of science since they can model natural phenomena such as heat dissipation, diffusion of electrostatic charges, and distribution of water in soil. Additionally, diffusion equations are often used in rendering and related fields to approximate the behaviour of light in highly scattering media [JMLH01; Sta95].

Analogous to path tracing, existing WoS algorithms start "paths" (i.e. walks) at "sensor points" and end them at "lights" (i.e. boundary points). This works well when the probability of finding source points is high, but produces high variance for sparse sources.

Because of the apparent parallels between the Monte Carlo algorithms used for solving rendering problems and those solving PDEs, we hope to leverage the decades of rendering research and apply them to PDEs to develop new robust and efficient Monte Carlo PDE solvers. A major step in rendering was the transition from unidirectional "forward" methods like path tracing to "backward" methods such as light tracing, photon mapping [Jen96] and virtual point lights (VPLs) [Kel97]. This precipitated the comprehensive framework of bidirectional rendering methods [Vea97] and, ultimately, the wealth of transport methods available today [PJH16]. In this paper, we mirror this development and propose extensions of WoS in both forward and backward directions.

The standard "forward" WoS algorithm leverages the mean value theorem to form recursive estimators for the solution. We instead formulate a mean value theorem for the Green's function. This allows us to derive a new class of WoS algorithm that start paths "backwards" from source and boundary points, distributing energy more evenly throughout the domain, before connecting either to sensor points directly, or to short sensor subpath, mimicking a form of "final gather" [Rei92] to reduce structured sampling artifacts.

We demonstrate the effectiveness and correctness of our method both by solving the diffusion equation in highly scattering media, and by rendering diffusion curve images [OBW*08]. Much like in light transport, we hope that the development of these algorithms opens the path to comprehensive bidirectional methods in the future for fully robust Monte Carlo solution of PDEs.

For simplicity, we limit ourselves to a particular class of elliptic PDEs called the Poisson equation with Dirichlet boundary condi-

tions, which we review in Sec. 3. We only handle homogeneous coefficients, but expect that our method can be extended to heterogeneous coefficients without much additional work by incorporating concurrent advances in WoS solvers [SSJC22]. Finally, Neumann boundary conditions are, to the best of our knowledge, not supported in any WoS algorithm to date and handling these is orthogonal to the contribution of our paper.

2. Related Work

Deterministic solvers vs. Monte Carlo. The trade-offs between traditional, finite element method (FEM)-based PDE solvers and Monte Carlo solvers are well explored, and we refer to Sawhney and Crane [SC20] and Sawhney et al. [SSJC22] for a thorough discussion. In this paper, we focus on problems with sparse sources. For these scenarios, traditional FEM-based solvers solvers do not work well out of the box, and their discretization must instead be adapted to faithfully incorporate high-frequency sources and boundaries. In the limit case of delta-sources, the source points must be explicitly incorporated into the mesh; failing to do so will result in missing their contribution entirely. Mirroring early finite-element radiosity methods in rendering, finite-element methods for PDEs have to continually grow in complexity to handle such challenging inputs. Much like in rendering, our belief is that the generality and simplicity of Monte Carlo solvers will make this complexity redundant for PDEs, and our new reverse formulations are a first step to making these methods robust enough for general use.

Bidirectional/two-pass rendering methods. Two-pass algorithms have a long tradition in rendering. In a first "backward" pass, these algorithms simulate transport starting at light sources and then cache illumination in a view-independent structure. A second "forward" pass then simulates transport from sensors to gather the cached illumination and generate the final image. Initially developed for radiosity [CGIB86; CW93; Rei92], this approach has found wide success in Monte Carlo rendering in the form of photon mapping [Jen01; Jen96] and VPL/many-light rendering [DKH*14; HPB07; Kel97; WABG06; WFA*05; WKB12]. These methods exploit that some transport is more easily simulated starting from lights and amortize much of the transport computation by caching and reusing it across sensor paths. We take inspiration from these methods, particularly variants developed for scenes with volumetric media [BJ17; DJBJ19; JC98; JNSJ11; JNT*11; JZJ08; NNDJ12a; NNDJ12b], and develop related WoS methods for solving PDEs. Much like the initial formulations of the aforementioned rendering methods, our current algorithms rely on ad hoc heuristics to choose when to connect sensor subpaths with light subpaths. A particularly exciting avenue for future work would be to develop a unified path space [GKDS12; HPJ12] that allows combining all these strategies automatically using multiple importance sampling (MIS) [VG95b]. We believe that these approaches applied to solving PDEs would lead to greatly improved robustness in the future.

Shell tracing. A method called *shell tracing* [LO07; MWM07] has been used in rendering to accelerate transport in a variety of (often dense) media. Shell tracing has been studied in particular in the context of granular media [MPG*16; MPH*15], but more recently there has been effort to use it for general volume rendering as well [LHW21]. There are very strong parallels between shell tracing and walk on spheres. In both methods, one expands the largest ball that fits inside the medium/domain and then picks a point on the boundary of the ball to recursively continue the path. This point is in both cases picked according to the exit distribution of the underlying random process. In walk on spheres, this distribution is uniform, but in shell tracing one has to sample from a more complicated *shell transfer function.* The techniques we develop here should to a large degree also apply to shell tracing and could lead to bidirectional shell tracing techniques.

Diffusion in graphics and rendering. The Poisson equation we solve is directly applicable to approximating multiple scattering in participating media [Sta95]. In rendering, this equation has traditionally been solved approximately with dipole or multi-pole diffusion methods [dEI11; DWdE*08; JB02; JMLH01], which make subsurface scattering practical at the cost of additional error on top of the diffusion approximation. Hybrid rendering methods [DJ07; HCJ13], which inject sources into a diffusion domain via a backward tracing pass, are most similar to our proposed WoS algorithm. However, our approach applied to subsurface scattering could solve the same diffusion equation without additional error or complex mirroring heuristics needed by prior methods. We show proof-of-concept results of our method applied to diffusion for multiple scattering, indicating fruitful future avenues for transferring our WoS work back into rendering. Diffusion problems in rendering have also been solved with FEM methods [AWB11; KPS*14], which solve the diffusion equations exactly (down to discretization error) and can handle heterogeneity. However, the FEM-based nature of these methods does not fit well with Monte Carlo rendering methods. We believe that in the future, these methods could be replaced by our proposed WoS algorithm extended to heterogeneous domains using the concurrent work of Sawhney et al. [SSJC22].

3. Background

In the following we will go over the fundamental mathematical concepts needed to understand our method and introduce the notation we use going forward. A reader familiar with PDE literature might want to skip to Sec. 3.2.

3.1. Partial Differential Equations

A partial differential equations (PDE) is an equation describing the property of the partial derivatives of a multi-variable function $u(x)$ where $x \in \mathbb{R}^n$. In this paper, we will use $U \subset \mathbb{R}^n$ to denote the domain of the function we are solving and ∂U to be the boundary of U. We use Δ to denote the Laplace operator, which is the sum of all the unmixed second partial derivatives. Our method is mainly focused on solving the Laplace and Poisson equations.

The Laplace equation. Given a Dirichlet boundary condition g, which is a function prescribing the value of the solution u on ∂U, the Laplace equation on U is:

$$\begin{aligned} \Delta u(x) &= 0 \qquad \text{if } x \in U, \\ u(x) &= g(x) \qquad \text{if } x \in \partial U. \end{aligned} \tag{1}$$

If we interpret u as a physical quantity such as heat, then the condition $\Delta u(x) = 0$ prescribes that the solution be at equilibrium: No energy is added or removed within the domain. The values of u within the domain are entirely defined by (and are convex combinations of) its values at the boundary ∂U.

The Poisson equation. A more general form of the Laplace equation allows for the addition of sources within the domain by modifying the equilibrium condition to include a source term f. This is the Poisson equation:

$$\begin{aligned} \Delta u(x) &= f(x) \quad \text{if } x \in U, \\ u(x) &= g(x) \quad \text{if } x \in \partial U. \end{aligned} \tag{2}$$

This is the equation we are interested in solving for this paper.

Green's Function. The method of Green's functions [Eva10] is a general approach for solving linear PDEs. In this approach, we first consider a reduced problem where we compute the solution $\mathcal{G}_y^U(x)$ to a Poisson problem where the source term is a delta impulse δ_y centered at y and the boundary value is 0:

$$\begin{aligned} \Delta u(x) &= \delta_y(x) \quad \text{if } x \in U, \\ u(x) &= 0 \quad \text{if } x \in \partial U. \end{aligned} \tag{3}$$

Notably, the solution of this linear PDE is symmetric ($\mathcal{G}_y^U(x) = \mathcal{G}_x^U(y)$), so we will write it as $\boldsymbol{\mathcal{G}}(x \leftrightarrow y)$ and omit the superscript when it is the entire domain U. We define the Green's function to be 0 if x or y are not in the interior of the considered domain.

Intuitively, we can interpret the Green's function as representing the amount of energy transported from point source y to x via all possible paths within the domain. Given the Green's function, we can therefore easily write down the solution for any Poisson problem with boundary values of 0 by integrating over *all* source points y in the domain and computing the energy transported to x:

$$u(x) = \int_U f(y)\, \boldsymbol{\mathcal{G}}(x \leftrightarrow y)\, \mathrm{d}y. \tag{4}$$

Poisson Kernel. Special care needs to be taken to incorporate boundary values into Eq. (4). This is accomplished using the *Poisson kernel*, which is the normal derivative of the Green's function:

$$\boldsymbol{\mathcal{P}}(x \to z) = \boldsymbol{\mathcal{P}}(z \leftarrow x) := \frac{\partial \boldsymbol{\mathcal{G}}(x \leftrightarrow z)}{\partial n(z)}, \tag{5}$$

where $x \in U$, $z \in \partial U$ and $n(z)$ is a vector normal to ∂U at z. Unlike the Green's function the Poisson kernel is not symmetric, and the directed arrows always point to the position on the boundary. For the Poisson and Laplace equations, the Poisson kernel integrates to one:

$$\int_{\partial U} \boldsymbol{\mathcal{P}}(x \to z)\, \mathrm{d}z = 1. \tag{6}$$

Representation Formula. Given the Green's function and its Poisson kernel, the solution for general Poisson equations can be expressed using the representation formula [Eva10]

$$u(x) = \int_U f(y)\, \boldsymbol{\mathcal{G}}(x \leftrightarrow y)\, \mathrm{d}y + \int_{\partial U} g(z)\, \boldsymbol{\mathcal{P}}(x \to z)\, \mathrm{d}z, \tag{7}$$

which is equivalent to Eq. (4) with an additional boundary term.

For the Laplace equation (1), there are no source terms in the domain ($f(x) = 0$), and the representation formula simplifies to:

$$u(x) = \int_{\partial U} g(z)\, \boldsymbol{\mathcal{P}}(x \to z)\, \mathrm{d}z. \tag{8}$$

The Mean Value Theorem. It is also possible to write an *integral equation* for the solution [Eva10]:

$$u(x) = \underbrace{\int_{B_x} f(y)\, \mathcal{G}(x \to y)\, \mathrm{d}y}_{\text{volume term}} + \underbrace{\int_{\partial B_x} \overbrace{u(z)}^{=g(z) \text{ when } z \in \partial U}\, \mathcal{P}(x \to z)\, \mathrm{d}z}_{\text{boundary term}}. \tag{9}$$

This equation looks superficially similar to Eq. (7), but has two important differences. Firstly, the integration, Green's function $\mathcal{G}(x \to y)$ and Poisson kernel $\mathcal{P}(x \to z)$ are now defined with respect to the largest ball B_x and sphere ∂B_x centered at x. This is attractive because these functions are known *analytically*. We use non-bold symbols for these functions now to indicate they are not over the entire domain, and, since the largest ball at x and y are generally different, we use a directed arrow that starts at the center of the ball. Secondly, the boundary term in Eq. (9) is now defined *recursively* in terms of u, resulting in a Fredholm integral equation much like the rendering equation [Kaj86].

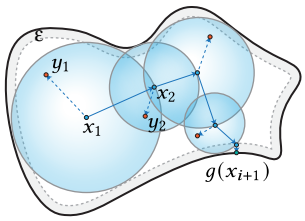
The Poisson kernel on the ball is a function of only radius, so, lacking any source terms, this equation states that the solution $u(x)$ is equal to the *weighted average* of u over the boundary of the ball.

3.2. The Walk on Spheres Algorithm

The classical walk on spheres algorithm can now be derived by applying Monte Carlo integration to Eq. (9). We sample a $y_i \sim p_{B_{x_i}}(y_i)$ inside ball B_{x_i} to estimate the volume term, sample $x_{i+1} \sim p_{\partial B_{x_i}}(x_{i+1})$ on the corresponding sphere to estimate the boundary term, and evaluate

$$\langle u(x_i) \rangle = \frac{f(y_i)\, \mathcal{G}(x_i \to y_i)}{p_{B_{x_i}}(y_i)} + \frac{\langle u(x_{i+1}) \rangle\, \mathcal{P}(x_i \to x_{i+1})}{p_{\partial B_{x_i}}(x_{i+1})}. \tag{10}$$

This is a single-sample estimator of solution u, which we denote $\langle u \rangle$. Notably, $\langle u(x_{i+1}) \rangle$ appears on the right-hand side, requiring a recursive evaluation of Eq. (10): For each sample x_{i+1}, we select a new ball $B_{x_{i+1}}$ centered on x_{i+1} and recurse, as illustrated in the inset figure. This process continues until we generate a sample x_{i+1} sufficiently close to the boundary ∂U of the domain (when the distance to the boundary at that point is less than ε), at which point we evaluate $g(x_{i+1})$ instead of recursing (see the inset figure).



4. Our Method

A major issue with the walk on spheres algorithm presented in the previous section is that we have very little control over where in the domain the walks end up. We can locally control the sample positions y and x_{i+1} in Eq. (10), but we are restricted to the interior and boundary of the ball B_x respectively, which might only cover a small part of U. There is no global way to steer walks towards areas

of high contribution (i.e. where f and g take on large values). This poses a problem in scenes with sparse (or even delta-) sources. As discussed in Sec. 1, a similar issue appears with forward path tracing in rendering which was addressed by "reverse" methods such as photon mapping. In the following we will derive a "reverse" version of the walk on spheres algorithm that is able to start walks at source points, sampled freely according to a global criterion, and distributes source contribution over the whole domain. This contribution can then be directly evaluated at sensor points or looked up during a forward walk. In particular, going back to Eq. (7) we can see that here we have two terms that integrate over the *whole* domain U and boundary ∂U respectively and thus cover the whole support of f and g. This gives us a starting point for a method that reasons about the global distribution of sources, but leaves us with the issue of computing $\boldsymbol{\mathcal{G}}(x \leftrightarrow y)$ and $\boldsymbol{\mathcal{P}}(x \to y)$, which are not known analytically for arbitrary domains.

Recursive integrals for Green's functions. Luckily it is easy to find an equivalence to the mean value theorem in Eq. (9) for the Green's function. In particular, recall from Sec. 3 that the Green's function is a solution to the Poisson equation (3). As such we can simply plug its definition (3) into Eq. (9) as follows

$$\begin{aligned}\boldsymbol{\mathcal{G}}(x \leftrightarrow y) &= \int_{B_x} \delta_y(y')\mathcal{G}(x \to y')\,\mathrm{d}y' + \int_{\partial B_x} \mathcal{P}(x \to x')\boldsymbol{\mathcal{G}}(x' \leftrightarrow y)\,\mathrm{d}x' \\ &= \mathcal{G}(x \to y) + \int_{\partial B_x} \mathcal{P}(x \to x')\boldsymbol{\mathcal{G}}(x' \leftrightarrow y)\,\mathrm{d}x'. \end{aligned} \tag{11}$$

Due to the symmetry of the Green's function in our problems, we can swap x and y and perform the same process on a ball B_y centered at y to obtain a dual mean value theorem on B_y:

$$\boldsymbol{\mathcal{G}}(x \leftrightarrow y) = \mathcal{G}(x \leftarrow y) + \int_{\partial B_y} \boldsymbol{\mathcal{G}}(x \leftrightarrow y')\mathcal{P}(y' \leftarrow y)\,\mathrm{d}y'. \tag{12}$$

Source and sensor expansions. This now gives us the freedom to make progress along a path both from sensor points (using Eq. (11)) and source points (using Eq. (12)), similar to the expansion of the 3-point form of light transport.

For simplicity we will illustrate this using the Poisson equation with a zero-boundary term (4) and describe how to incorporate a non-zero boundary term in Sec. 5.3.

We can perform a "forward" unidirectional expansion by inserting Eq. (11) into Eq. (4). Doing this twice gives

$$\begin{aligned} u(x_0) = &\underbrace{\int_U \mathcal{G}(x_0 \to y_0) f(y_0)\,\mathrm{d}y_0}_{\text{1 "bounce" transport}} \\ &+ \underbrace{\iint_{U\times\partial B_{x_0}} \mathcal{P}(x_0 \to x_1)\mathcal{G}(x_1 \to y_0) f(y_0)\,\mathrm{d}x_1 y_0}_{\text{2 "bounce" transport}} \\ &+ \underbrace{\iiint_{U\times\partial B_{x_0}\times\partial B_{x_1}} \mathcal{P}(x_0 \to x_1)\mathcal{P}(x_1 \to x_2)\boldsymbol{\mathcal{G}}(x_2 \leftrightarrow y_0) f(y_0)\,\mathrm{d}x_2 x_1 y_0}_{\text{3+ "bounce" transport}}. \end{aligned} \tag{13}$$

Alternatively, we could perform a unidirectional expansion in the "reverse" direction "starting at y_0", by using Eq. (12). Doing this twice gives a structurally similar, but distinct set of integrals:

$$\begin{aligned} u(x_0) = &\underbrace{\int_U \mathcal{G}(x_0 \leftarrow y_0) f(y_0)\,\mathrm{d}y_0}_{\text{1 "bounce" transport}} \\ &+ \underbrace{\iint_{U\times\partial B_{y_0}} \mathcal{G}(x_0 \leftarrow y_1)\mathcal{P}(y_1 \leftarrow y_0) f(y_0)\,\mathrm{d}y_1 y_0}_{\text{2 "bounce" transport}} \\ &+ \underbrace{\iiint_{U\times\partial B_{y_0}\times\partial B_{y_1}} \boldsymbol{\mathcal{G}}(x_0 \leftrightarrow y_2)\mathcal{P}(y_2 \leftarrow y_1)\mathcal{P}(y_1 \leftarrow y_0) f(y_0)\,\mathrm{d}y_2 y_1 y_0}_{\text{3+ "bounce" transport}}. \end{aligned} \tag{14}$$

Finally, since we can freely choose between Eqs. (11) and (12) at each step, we can perform "bidirectional" expansions by, for instance, expanding first with Eq. (11) followed by Eq. (12):

$$\begin{aligned} u(x_0) = &\underbrace{\int_U \mathcal{G}(x_0 \to y_0) f(y_0)\,\mathrm{d}y_0}_{\text{1 "bounce" transport}} \\ &+ \underbrace{\iint_{U\times\partial B_{x_0}} \mathcal{P}(x_0 \to x_1)\mathcal{G}(x_1 \leftarrow y_0) f(y_0)\,\mathrm{d}x_1 y_0}_{\text{2 "bounce" transport}} \\ &+ \underbrace{\iiint_{U\times\partial B_{x_0}\times\partial B_{y_0}} \mathcal{P}(x_0 \to x_1)\boldsymbol{\mathcal{G}}(x_1 \leftrightarrow y_1)\mathcal{P}(y_1 \leftarrow y_0) f(y_0)\,\mathrm{d}y_1 x_1 y_0}_{\text{3+ "bounce" transport}}. \end{aligned} \tag{15}$$

This now suggests a multitude of new estimators that use a different number of source and sensor segments.

Analogies to light transport, path spaces, and MIS. We illustrate the three example expansions from Eqs. (13)–(15) in Fig. 1 (bottom). Since both the Green's function and the rendering equation are recursive Fredholm integrals, there is a natural analogy between these equations and different strategies of bidirectional light transport Fig. 1 (top). In fact, repeatedly expanding the recursion in the three-point form of the rendering equation is the typical process to obtain Veach's path integral formulation [Vea97], which provides methods like bidirectional path tracing [LW93; VG95a] with a powerful way to combine all these strategies into one algorithm using MIS [VG95b].

Unfortunately, the analogies depicted in Fig. 1 are imperfect. While the Green's function itself is symmetric, Eqs. (11) and (12) use different domains of integration (a sphere ∂B_x or ∂B_y). This means that each distinct sequence of expansion directions results in a different path space! This is easy to confirm by observing that the domains of integration for 3+ "bounce" transport in Eqs. (13)–(15) are all distinct. This is in contrast to the rendering equation, where using surface area measure ensures that expansion from either direction produces the exact same path space.

This means that while MIS can still be performed *within* a single path space choice, it is not immediately clear how to MIS *across* these different path space choices. Doing so would be akin to using

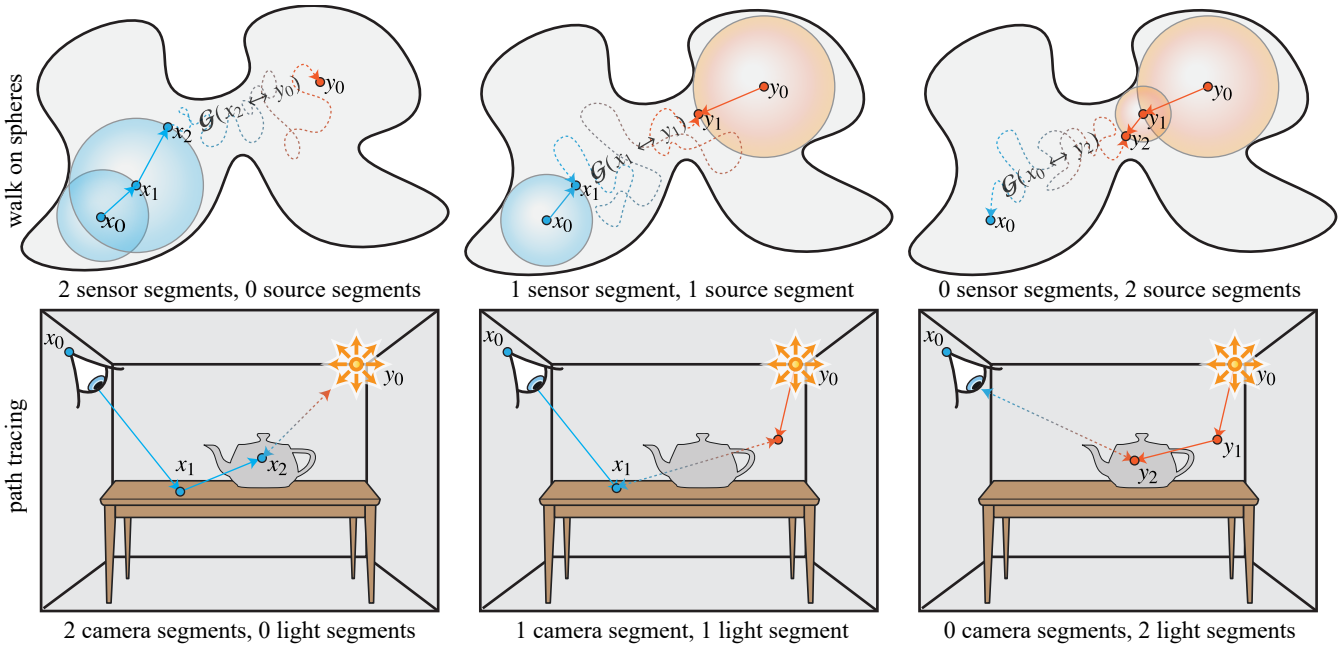


Figure 1: *Top row: We show the different ways our new formalism lets us construct WoS paths, corresponding to Eq. (13), Eq. (15) and Eq. (14) (left to right respectively). Bottom row: These methods correspond closely to forward path tracing, bidirectional path tracing and light tracing (left to right respectively).*

MIS to combine VPLs, photon mapping, and bidirectional path tracing within a unified path space [GKDS12; HPJ12]. Nevertheless, even before being cast into a unified path space, photon mapping and VPL methods proved highly successful using hand-crafted criteria for determining how many steps to take along a camera subpath before connecting to a light subpath. In the next sections we explore several such possible bidirectional combinations, and leave the exciting prospect of a unified path space for future work.

5. Algorithms

In the previous section, we showed a formalism for estimating the Green's function by performing step-wise recursive expansion in either the forward or backward direction. In this section, we will use this formulation to derive practical algorithms for solving the Poisson equation.

We begin by separating the full Poisson problem as $u(x) = v(x) + w(x)$, where v and w are source-only and boundary-only sub-problem respectively:

$$\begin{aligned} \Delta v(x) &= f(x) \quad & \Delta w(x) &= 0 \quad & &\text{if } x \in U, \\ v(x) &= 0 \quad & w(x) &= g(x) \quad & &\text{if } x \in \partial U. \end{aligned} \tag{16}$$

We then solve for v and w separately using different algorithms. From the linearity of the Laplace operator, we can see that the solution $u(x) = v(x) + w(x)$ satisfies the original Poisson problem (Eq. (2)), with $\Delta(v+w) = \Delta v + \Delta w = f$ and $v + w = g$. Solutions to the Poisson equation are unique [Eva10], allowing us to retrieve the original solution u exactly via the subproblems v and w.

In the remaining subsections, we will first show how to estimate the Green's function using forward and backward walks (Sec. 5.1), which we then use to estimate the partial solutions v and w with backward walk on spheres (Sec. 5.2, Sec. 5.3). We then show how to use backward walks efficiently through reuse (Sec. 5.4), before showing how to combat bias (Sec. 5.5) and how to selectively combine forward and backward walks with a practical heuristic (Sec. 5.6).

5.1. Estimating the Green's Function

We can easily obtain both forward and backward estimators of the Green's function by taking a one-sample Monte Carlo estimate of Eqs. (11) and (12):

$$\begin{aligned} \langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y) \rangle &= \mathcal{G}(x \to y) + \frac{\mathcal{P}(x \to x') \langle \boldsymbol{\mathcal{G}}(x' \leftrightarrow y) \rangle}{p^{\partial B_x}(x')} \\ \langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y) \rangle &= 0 \quad \text{if } x \in \partial U \text{ or } y \in \partial U \end{aligned} \tag{17}$$

$$\begin{aligned} \langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y) \rangle &= \mathcal{G}(x \leftarrow y) + \frac{\mathcal{P}(y \to y') \langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y') \rangle}{p^{\partial B_y}(y')} \\ \langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y) \rangle &= 0 \quad \text{if } x \in \partial U \text{ or } y \in \partial U \end{aligned} \tag{18}$$

where x' and y' are points sampled on the boundary of the ball B_x and B_y, with densities $p^{\partial B_x}(x')$ and $p^{\partial B_y}(y')$.

Eqs. (17) and (18) recursively estimate the Green's function. At each recursion step, we are free to estimate the next expansion using Eq. (17) or Eq. (18), allowing us great flexibility in mixing forward and reverse steps. The recursion terminates when samples x' or y' land on the boundary. In general, the probability of this event is almost zero, and we follow prior WoS work [SC20; SSJC22] and

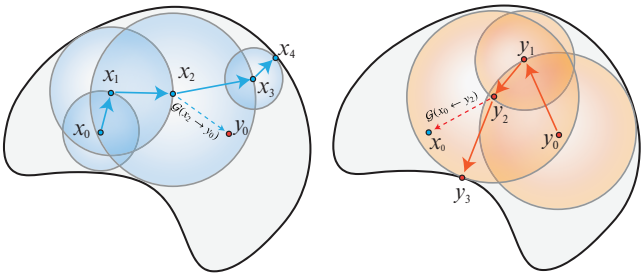


Figure 2: *We can estimate the Green's function by either (left) following a "forward" path $\{x_i\}$ and estimating $\langle \boldsymbol{\mathcal{G}}(x_0 \leftrightarrow y_0)\rangle = \mathcal{G}(x_2 \to y_0)$, or (right) following a "reverse" path $\{y_i\}$ and estimating $\langle \boldsymbol{\mathcal{G}}(x_0 \leftrightarrow y_0)\rangle = \mathcal{G}(x_0 \leftarrow y_2)$.*

instead terminate if the sample lands within a small distance ε of the boundary, at the cost of a small amount of bias. Fig. 2 shows an example of a purely forward and purely reverse walk.

Eqs. (17) and (18) form the building blocks of our algorithms, and we will now use them to estimate solutions of Poisson problems.

5.2. Estimating the Source Solution v

We begin by writing the representation formula for the Poisson equation of the sources-only term v (Eq. (16)). Because the boundary term is zero, we obtain

$$v(x) = \int_U f(y)\boldsymbol{\mathcal{G}}(x \leftrightarrow y)\,\mathrm{d}y. \tag{19}$$

We obtain a Monte Carlo estimator of this integral in two steps. First, we take a one-sample estimate of Eq. (19) by choosing point y with density $p^U(y)$ (e.g. proportional to source term $f(y)$). We then estimate $\boldsymbol{\mathcal{G}}(x \leftrightarrow y)$ with either Eq. (17) or Eq. (18) to obtain

$$\langle v(x)\rangle = \frac{f(y)\langle \boldsymbol{\mathcal{G}}(x \leftrightarrow y)\rangle}{p^U(y)}. \tag{20}$$

By reducing the Poisson problem to estimating the Green's function, we get great flexibility in estimating v in any combination of forward and backward steps. Notably, unlike the classical forward WoS algorithm, Eq. (20) allows us to sample y proportional to the source term $f(y)$ over the whole domain.

5.3. Estimating the Boundary Solution w

Writing the same representation formula for the boundary-only Poisson equation of w (Eq. (16)), we obtain

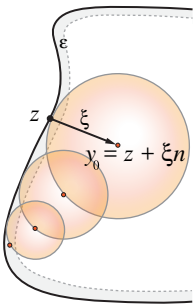
$$w(x) = \int_{\partial U} \boldsymbol{\mathcal{P}}(x \to z)g(z)\,\mathrm{d}z. \tag{21}$$

Estimating this requires knowing the Poisson kernel $\boldsymbol{\mathcal{P}}(x \to z)$ over the whole domain. However, we can reduce it to the Green's function by noting that the Poisson kernel is the normal derivative of the Green's function. Letting n denote the normal at z, we have:

$$\begin{aligned}\boldsymbol{\mathcal{P}}(x \to z) &= \frac{\partial \boldsymbol{\mathcal{G}}}{\partial n}(z) && (22)\\ &= \lim_{\xi \to 0} \frac{\boldsymbol{\mathcal{G}}(x \leftrightarrow z+\xi n) - \boldsymbol{\mathcal{G}}(x \leftrightarrow z)}{\xi} && (23)\\ &= \lim_{\xi \to 0} \frac{\boldsymbol{\mathcal{G}}(x \leftrightarrow z+\xi n)}{\xi} && (24)\end{aligned}$$

where the last step used the fact that $\boldsymbol{\mathcal{G}}(x \leftrightarrow z) = 0$ for $z \in \partial U$.

We can approximate Eq. (24) at the cost of bias by choosing a finite ξ instead of taking the limit, which is equivalent to taking the finite differences of the Green's function (see the figure on the right). This allows us to estimate the Poisson function with the Green's function estimators introduced in Sec. 5.1:



$$\langle \boldsymbol{\mathcal{P}}(x \to z)\rangle = \frac{\langle \boldsymbol{\mathcal{G}}(x \leftrightarrow z+\xi n)\rangle}{\xi} \tag{25}$$

By inserting Eq. (25) into Eq. (21) and applying Monte Carlo integration to z, we obtain the estimator

$$\langle w(x,z)\rangle = \frac{g(z)\langle \boldsymbol{\mathcal{P}}(x \to z)\rangle}{p^{\partial U}(z)} = \frac{g(z)\langle \boldsymbol{\mathcal{G}}(x \leftrightarrow z+\xi n)\rangle}{p^{\partial U}(z)\xi} \tag{26}$$

for w, where z is sampled from density $p^{\partial U(z)}$ on the boundary ∂U (e.g. proportional to boundary term $g(z)$). Much like the estimator for v, we have great flexibility in estimating the Green's function using any combination of forward- and backward steps. Unlike the classical forward WoS algorithm, Eq. (26) allows us to sample z proportional to the boundary term $g(y)$ over the entire boundary.

5.4. A Two-Pass Reverse Walk-on-Spheres Algorithm

In practice, we are usually interested in not only in estimating the solution $f(x)$ at a single point, but over a dense region. This allows for an efficient algorithm that *reuses* reverse walks.

If we recursively expand Eq. (20) or Eq. (26) using the reverse estimator Eq. (18), then at each step y_i the walk contributes to *all* points x within the ball B_{y_i}. This is analogous to VPLs, where at each bounce the VPL contributes its flux to all points in the scene, modulated by a geometry term. The equivalent of the geometry term for reverse WoS is the Green's function on the ball, $\mathcal{G}(x \leftarrow y_i)$.

We exploit this by first performing a large number of reverse walks from the boundary and sources in the domain, and store the ball and Green's function estimate of each step of each walk in a spatial data structure, as shown in Alg. 1. To estimate the solution at x, we then simply look up into the data structure to obtain all balls that overlap with x and accumulate each of their Green's function-weighted contributions. The pseudo-code for the look up pass is shown in Alg. 2. This is analogous to two-pass many-light algorithms common in rendering [DKH*14]. Suppose we know all the points we want to evaluate in advance, for example when solving a Poisson's equation in 2D where we want the solution for each pixel, we can rasterize all the disks directly when generating the walks without storing them and performing the look up. We

Algorithm 1 generate_reverseWoS(): Generate and store walks

```
/* Sample N reverse source walks and store into V. */
V ← vertex_storage()
for i ← 0 to N do
    y, p_y ← sample_source()      // Sample y according to pdf p_y.
    f_y ← f(y)                    // Evaluate the source term at y.
    r_y ← distance_to_boundary(y)

    /* Loop until the path hits the boundary. */
    while r_y > ε do
        V.store(y, p_y N, f_y, r_y)
        y ← sample_sphere_uniform(y, r_y)  // Continue the walk.
        r_y ← distance_to_boundary(y)
    end while
end for

/* Sample M reverse boundary walks and store into W. */
W ← vertex_storage()
for j ← 0 to M do
    z, p_z ← sample_boundary()  // Sample z according to pdf p_z.
    g_z ← g(z)                  // Evaluate the boundary value at z.
    n ← calculate_normal(z)     // Calculate the local normal n.
    y = z + ξn                  // Push the point away from the boundary.
    r_y ← distance_to_boundary(y)

    /* Loop until the path hits the boundary. */
    while r_y > ε do
        W.store(y, p_y M, g_z/ξ, r_y)
        y ← sample_sphere_uniform(y, r_y)  // Continue the walk.
        r_y ← distance_to_boundary(y)
    end while
end for
return V, W
```

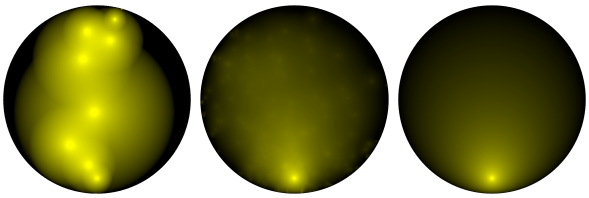
Algorithm 2 look_up_reverseWoS(x, V, W): Look up solution $u(x)$

```
Input: x, V, W
u ← 0
for (y, w_y, h_y, r_y) in V or W do      // h_y is f_y for V and g_y for W.
    if |x − y| < r_y then
        u ← u + h_y G(x ← y)/w_y
    end if
end for
return u
```

show an example of a solution estimate using an increasing number of reverse walks in Fig. 3. In contrast to two-pass light transport algorithms like photon mapping, our algorithm does not introduce extra bias for the source solution except the ε boundary.

5.5. Bias Compensation

Although the finite difference method in Sec. 5.3 makes it possible to estimate the Poisson kernel via the Green's function, the finite step ξ introduces additional bias. This is made worse by the fact that ξ must be larger than ε for practical reasons: If $\xi \leq \varepsilon$, then reverse walks starting at the boundary will immediately terminate; larger values of ξ are needed to "push off" walks away from the boundary.



1 sample *64 samples* *Converged result*

Figure 3: *Estimating the Poisson equation for a single point source in a disk domain with a black boundary using 1, 64, and 160K reverse walks.*

In practice, the bias from finite differences manifests as darkening of the solution due to reverse walks terminating early (Fig. 4). This means that, unlike Eq. (6), the finite difference Poisson kernel no longer integrates to 1.

We can compensate for this fact by *renormalizing* the Poisson kernel. While evaluating reverse walks, in addition to the solution w we also estimate the integral of the Poisson kernel at each point using a second Monte Carlo estimate,

$$\int_{\partial U} \langle \boldsymbol{\mathcal{P}}(x \to z) \rangle \, \mathrm{d}z \approx \frac{1}{M} \sum_{j=0}^{M} \frac{\langle \boldsymbol{\mathcal{P}}(x \to z_j) \rangle}{p^{\partial U}(z_j)} = \langle \mathcal{P}_{\mathrm{norm}}(x) \rangle, \quad (27)$$

where $z_1, \ldots, z_M$ are the boundary samples generated in the course of solving for w. Note that care should be taken that $p^{\partial U}(z) > 0$ over the entire boundary. Even if the boundary term $g(z) = 0$ for some of the boundary, the Poisson kernel is not.

Dividing Eq. (26) by the estimate of the normalization factor $\langle \mathcal{P}_{\mathrm{norm}}(x) \rangle$ then allows us to compensate for the systematic darkening caused by the finite difference Poisson kernel:

$$\langle w(x,z) \rangle = \frac{g(z) \langle \boldsymbol{\mathcal{P}}(x \to z) \rangle}{p^{\partial U}(z) \langle \mathcal{P}_{\mathrm{norm}}(x) \rangle} \quad (28)$$

Although this estimate is still biased, the apparent error is much reduced (Fig. 4, right).

5.6. Combining Forward and Reverse Walks

Forward- and reverse walk on spheres share some of the same tradeoffs as forward and reverse transport simulation in rendering.

no normalization *reference* *with normalization*

Figure 4: *Reverse WoS without normalization (left) is darker than the ground truth (mid). With normalization, the result (right) has the same brightness as the reference.*

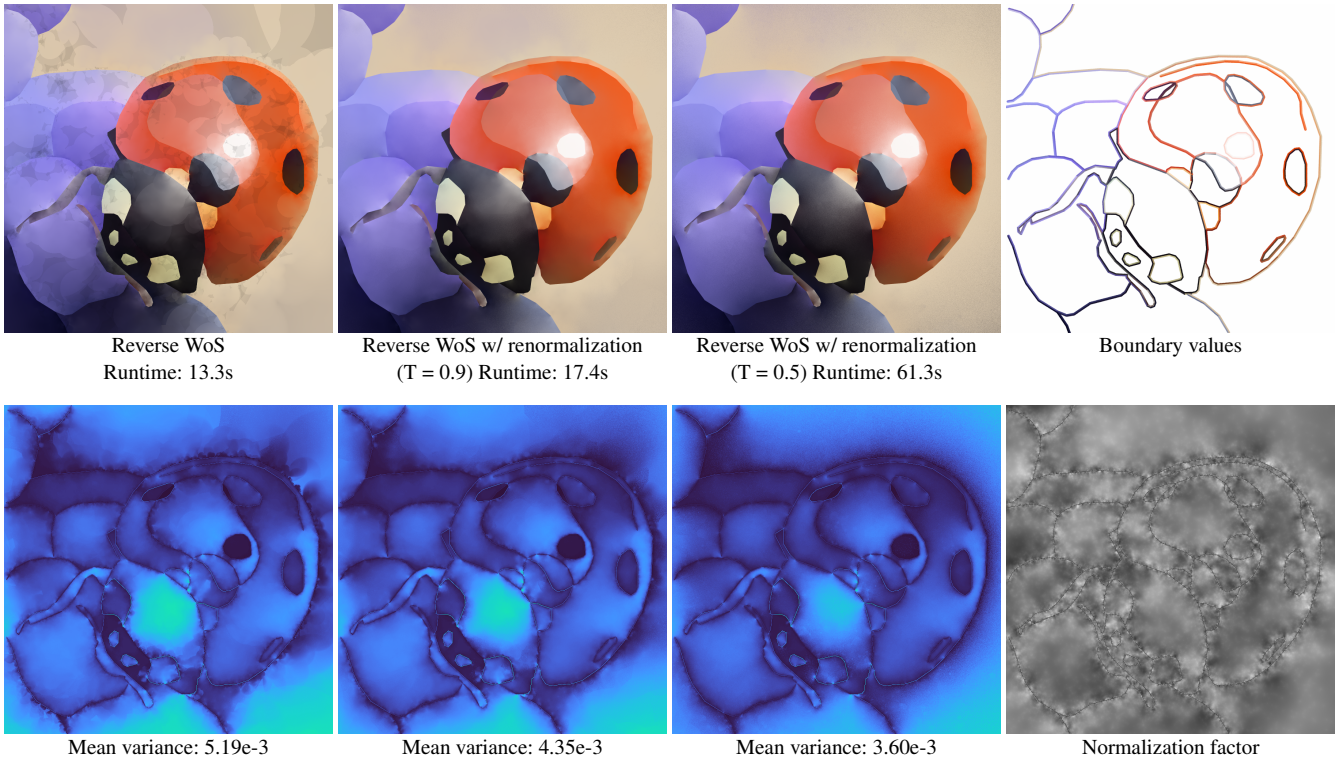


Figure 5: *Here we are solving a Laplace equation for the diffusion curve images [OBW*08], the boundary value is shown in the top right and the normalization factor is showed in the bottom right. If using purely reverse walks, insufficient samples of boundary values will lead to structured artifacts, especially in the region where reverse walks can hardly reach (left most). These artifacts can be fixed by doing a final gather with the Poisson's kernel heuristic (middle two).*

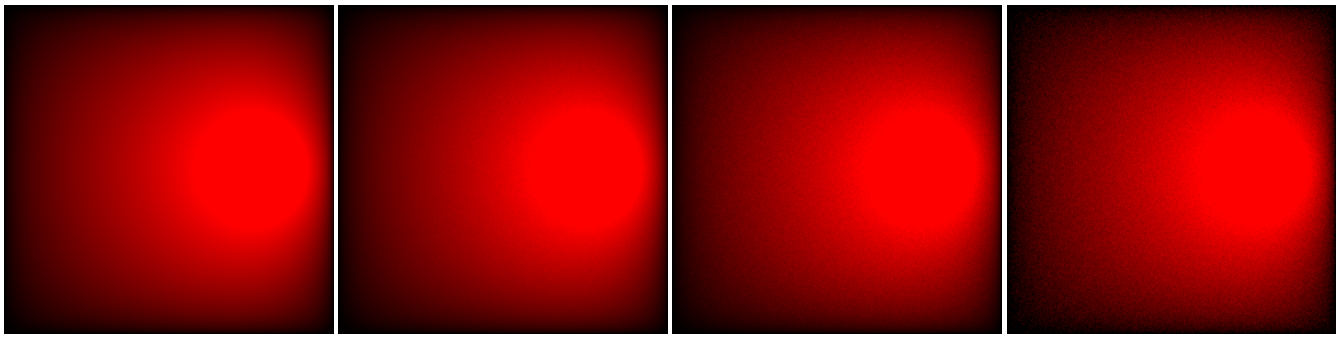


Figure 6: *Forward-and reverse walks can be combined when there is source term, these images are solutions to a source-only problem with one point source inside a black square boundary. Since we are reusing reverse walks and solving the equation for the entire domain, the reverse WoS algorithm will provide a smooth and noise-free image (left most). Combining walks from both directions with different choices of forward steps (middle two) or using only forward walks (right most) will introduce more noise.*

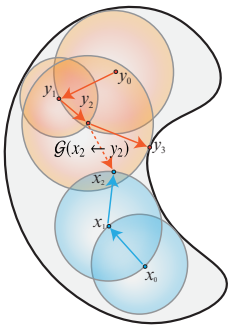
Sparse, high frequency sources are much more difficult to find for forward methods than reverse methods; simultaneously, it is much more difficult to get even coverage of the sensor points for reverse methods than forward methods.

For example, if we are interested in computing the solution in only a small subset of a scene, it may be more difficult for reverse walks to contribute to the solution. This is analogous to light tracing performing poorly when the camera only views a small part of a scene.

Although the different path spaces preclude robustly weighted combinations of all forward and reverse strategies using MIS

(Sec. 4), we take inspiration from early light transport work [Jen01] and choose between different combinations of forward and reverse strategies based on heuristics.

For the boundary solution w, we already have access to a metric of how well reverse WoS performs: The integral of the finite difference Poisson kernel, $\langle \mathcal{P}_{\text{norm}}(x) \rangle$. If this estimate deviates significantly from 1, reverse WoS is performing poorly. This leads to a simple but effective heuristic for combining forward and reverse walks: Instead of computing the solution $f(x)$ from the data structure directly as in Sec. 5.4, we first evaluate $|\langle \mathcal{P}_{\text{norm}}(x) \rangle - 1| < T$ to see if the Poisson kernel norm deviates from unity by more than a threshold T. If it does, the reverse WoS solution is unreliable, and we perform one forward WoS step and repeat the procedure. This continues until the heuristic succeeds, at which point we look up into the data structure to estimate the solution (see the inset figure). This is exactly analogous to final gather methods in graphics [Rei92], and helps greatly to reduce artifacts at little extra cost (Fig. 5). For the source solution v, we can still perform a "final gather" though we currently set the number of forward steps explicitly (Fig. 6). We show the pseudo-code for combining forward- and reverse WoS in Alg. 3.



Algorithm 3 bidirectional_WoS(x, V, W): Combine forward- and reverse walks

Input: x, V, W
$u \leftarrow 0$
$r_x \leftarrow$ distance_to_boundary(x)
while True **do**
  / If we hit the boundary when doing forward walk. */*
  if $r_x < \varepsilon$ **then**
    $u \leftarrow u + g(x)$
    break
  end if

  if terminate_forwardWoS(x) **then**
    / Evaluate the solution using Alg. 2. */*
    $u \leftarrow u +$ look_up_reverseWoS(x, V, W)
    break
  end if
  / Continue the forward walk */*
  / Sample a source point inside the ball $B(x, r_x)$. */*
  $y, p_y \leftarrow sample\_source(x, r_x)$
  $u \leftarrow u + \mathcal{G}(x \to y)\, f(y)/p_y$
  $x \leftarrow$ sample_sphere_uniform(x, r_x)
  $r_x \leftarrow$ distance_to_boundary(x)
end while
return u

6. Implementation and Results

6.1. Algorithm implementation

As with the forward WoS algorithm, generating a reverse walk only requires querying the closest point to the boundary in order to expand the largest sphere. We implemented a 2D version of reverse WoS entirely on the CPU and use a standard acceleration structure [Saw*21] to make closest point query efficient. The main bottleneck in this case is drawing the Green's disks, which we currently do naively by testing all pixels within each disk's bounding box. Performing the reverse walk on the CPU but then splatting the Green's disks using rasterization on the GPU would likely result in a dramatic speedup. To implement our method one needs to evaluate $\mathcal{G}(x \to y)$ and $\mathcal{P}(x \to y)$. The concrete values of these depend on the PDE one is solving and we refer to the appendix in Sawhney et al. [SSJC22] for a comprehensive listing.

6.2. Bias

Our algorithm uses two parameters to solve the boundary solution $w(x)$ and both introduce a small bias to the result which we analyze in Fig. 7. The first is the stopping tolerance ε, which is used to terminate the random walk in both forward and reverse WoS. In forward WoS, using a large ε results in a thick boundary, but it has little effect on the reverse WoS result. The dominant bias in reverse WoS is caused by ξ: the finite difference step size and also the distance we push the start of the reverse walk off the boundary. Large ξ make the boundary values look like they are inside the domain (Fig. 7, top right), while small ξ lead to results that are too dark (Fig. 7, 2nd column). If we choose $\xi = \varepsilon$, roughly half of the walks will be terminated after the first step.

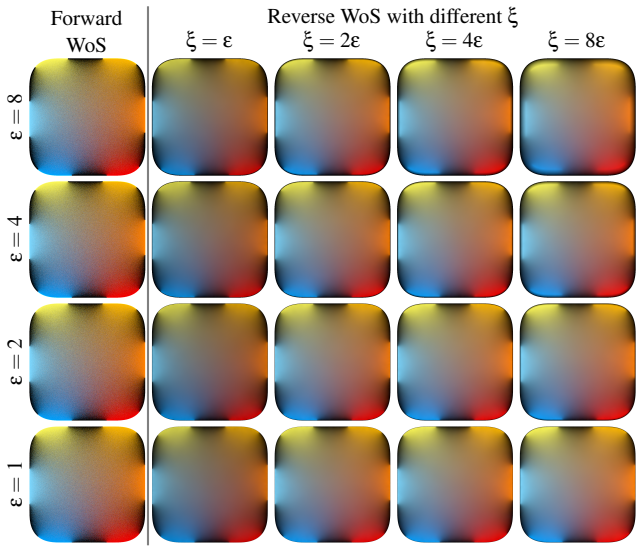


Figure 7: *An equal-time comparison of the forward WoS (1st column) and the reverse WoS with different ξ and ε. Using large ξ will "push" the boundary values into the domain, making them appear like sources inside the domain (top right corner). Setting ξ too close to ε will cause the algorithm to underestimate the Poisson's kernel (2nd column). Changing ε while keeping ξ fixed (moving diagonally down and right on the grid) has little effect on reverse WoS.*

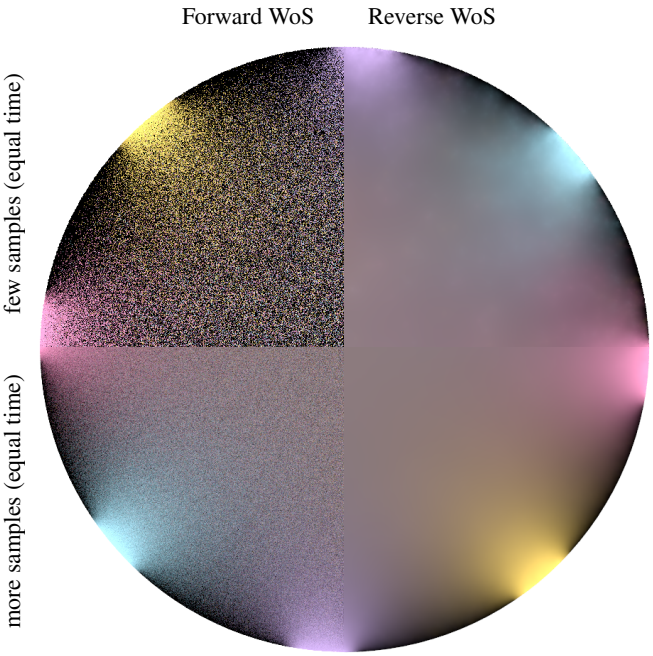


Figure 8: *We compare forward WoS (left) to reverse WoS (right) at equal time with few samples (top) and with* 16× *as many samples (bottom). Reverse WoS produces a smooth result even with few samples, converging more quickly than forward WoS.*

6.3. Comparison between forward and reverse WoS

We compared our method with forward WoS algorithm in multiple scenes to evaluate the efficiency, quality and robustness of our method. We found our method works better than the forward WoS in several cases because reverse WoS is able to importance sample the source term f and boundary value g in Eq. (7) globally, making the algorithm focus more on the sources with high impact to the entire scene.

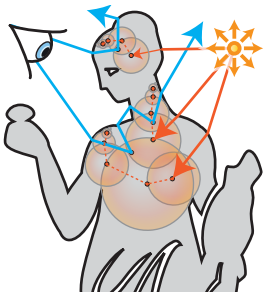
Sparse boundary values. Fig. 8 shows the estimated solution of a Laplace's equation of both algorithms using the same amount of time. In this example we set the boundary value to be 0 (black) at most boundary locations, leaving only a few small regions with colorful values along the circular boundary. A forward walk has no control of where the path will hit the boundary, so in a scene with sparse boundary values, most walks are unlikely to receive a large contribution, resulting in high variance. Just as in purely unidirectional path tracing, the variance for forward WoS would get arbitrarily worse if we were to make the "lights" (boundary values) even more concentrated. In contrast, reverse WoS can importance sample the boundary values, dramatically reducing variance.

Sparse sources. Fig. 9 shows the estimated solution by reverse and forward WoS on a Poisson equation with black boundary, but three differently sized disk-shaped sources inside the domain. Since the source terms are spatially sparse, it is difficult for a forward WoS path to hit those disks sources and evaluate their contribution along the forward path. In the reverse WoS, however, we know where the disks are located, so we can easily importance sample f in Eq. (7) when sampling the starting points of our paths. This importance sampling is essentially choosing global optimal sampled source points, while forward WoS can only sample local optimal choices at each step.

6.4. Reverse WoS for the diffusion approximation

Subsurface scattering is an effect that is expensive to reproduce accurately using forward path tracing. In dense, high albedo media, distances between scattering events are short, but light very rarely gets absorbed. This can lead to paths with hundreds of vertices, even when using techniques such as Russian roulette. Tracing all of these paths fully results in long render times, and in particular render times can strongly depend on the choice of medium parameters. To circumvent this, practical renderers often approximate subsurface scattering via a diffusion equation. While this introduces some error, especially when the medium is not very dense, it has the potential to greatly reduce render times. For an overview of related work on diffusion approximation in rendering, see Sec. 2. In the following we discuss a straightforward application of the algorithms discussed in the previous section to rendering diffusive media.

As in Sec. 5.4, we use a two-pass algorithm. In the first pass we generate photons on light sources (just as in photon mapping) and start tracing paths through the scene (shown in orange on the right). The first time we scatter inside a medium, we deposit a point source contribution there. That is, we start a reverse walk until we reach the medium boundary and store balls with the photons contribution attenuated by the Poisson kernel. In the



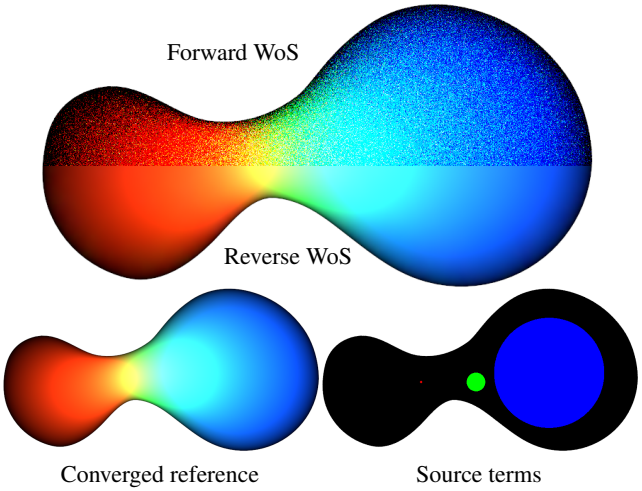


Figure 9: *Here we perform an equal-time comparison of forward (top of split) vs. our reverse (bottom of split) WoS on a scene with three sparse sources (bottom right) and zero boundary conditions. Compared to the converged reference (bottom left), our approach with only 4e4 sampled paths produces visually better results than forward WoS with 3.24e6 total paths.*

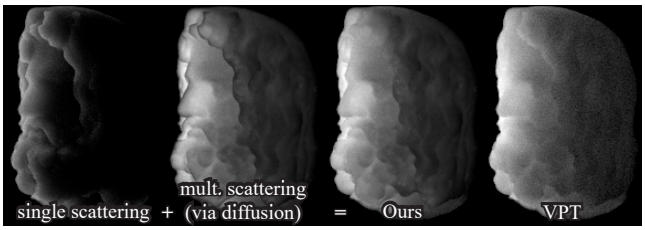


Figure 10: *We use path tracing to quickly compute the single scattering contribution (carefully excluding multiple-scattering paths, left) and our reverse WoS algorithm to compute the multiple-scattering part of the solution (middle-left). Adding these together gives us the full light transport (middle-right), albeit with error introduced by the diffusion approximation compared to VPT (right).*

forward pass we trace rays starting at the camera (shown in blue on the right). When we scatter inside the medium, we look up the multiple scattering contribution based on the stored diffusion solution. We also continue tracing the path to integrate any additional single scattering contribution. Our method is able to easily handle complex geometry and cases such as thin features which are often problematic with approaches based on further dipole approximations. Still, the results do not match the ground truth perfectly (see Fig. 10), but this is to be expected since we do still approximate light transport using a diffusion approximation. In scenes with small and directional light sources VPT will not be able to resolve an image, even with next event estimation. Our method produces a recognizable image even with few photons and quickly converges (see Fig. 11).

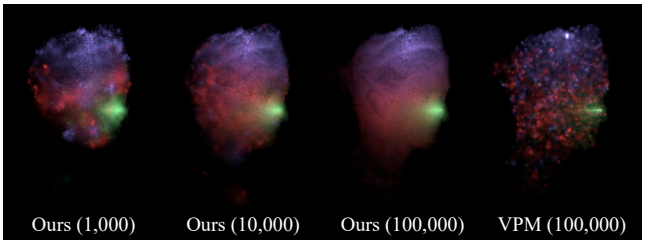


Figure 11: *Our method especially shines in scenes with sparse, strongly directional light sources such as the one shown above. In the left three images we show the impact of tracing more photons and the image quickly resolves. At the same number of photons, volumetric photon mapping (VPM) still shows clear artifacts. This is because each photon in our method can spread energy over much larger area without negatively affecting the quality of the solution.*

7. Conclusion, Limitations and Future Work

Conclusion. In this work we have presented a bidirectional formulation for the Walk on Spheres algorithm, taking the first steps towards a path integral formulation for Monte Carlo PDE solvers. The set of estimators and algorithms that we derive based on this outperform tradition walk on spheres on many scenes, as shown in Sec. 6. Going full circle, we can even use our method in a rendering context to speed up path tracing of volumetric media. We are particularly excited about the deep structural similarities to rendering and hope that these continue to inspire fruitful transfer of research between the two fields.

Limitations and Future Work. There are several topics that we do not address in this paper. Practically, the bias introduced by the technique we use to start walks on the boundary is unfortunate. While this does not affect source-term only problems, we would still prefer to find a more elegant way to estimate the Poisson kernel of the domain in a backwards fashion. In particular, it is desirable to find a method to step off the boundary without using the finite-difference approximation of the normal derivative. On the theoretical side, we do not extend the family of PDEs that WoS can solve. In particular, the restriction to Dirichlet boundary conditions restricts the problems we can solve and limits the practicality of our method. That said, the theory presented in Sec. 4 is very general and should easily be able to incorporate Neumann and Robin boundary conditions once they are integrated into the WoS framework. Finally, an obvious next step related to our contribution is to establish a single path integral formulation for all path construction strategies to allow for robust combination of strategies via MIS as discussed in Sec. 4. We expect that this will unlock a large step forward in Monte Carlo PDE estimation and our work provides a starting point to do so.

References

[AWB11] ARBREE, A., WALTER, B., and BALA, K. "Heterogeneous subsurface scattering using the finite element method". *IEEE TVCG* 17.7 (July 2011). DOI: 10/cmnn8f 2.

[BJ17] BITTERLI, B. and JAROSZ, W. "Beyond points and beams: higher-dimensional photon samples for volumetric light transport". *Proc. SIGGRAPH* 36.4 (July 2017). DOI: 10/gfznbr 2.

[CGIB86] COHEN, M. F., GREENBERG, D. P., IMMEL, D. S., and BRACK, P. J. "An efficient radiosity approach for realistic image synthesis". *IEEE CG&A* 6.2 (Mar. 1986). DOI: 10/c3mmt4 2.

[CJ16] CHRISTENSEN, P. H. and JAROSZ, W. "The path to path-traced movies". *Foundations and Trends® in Computer Graphics and Vision* 10.2 (Oct. 2016). DOI: 10/gfjwjc 1.

[CW93] COHEN, M. F. and WALLACE, J. R. *Radiosity and Realistic Image Synthesis*. NY: Academic Press, 1993. ISBN: 978-0-12-178270-2 1, 2.

[dEI11] D'EON, E. and IRVING, G. "A quantized-diffusion model for rendering translucent materials". *Proc. SIGGRAPH* 30.4 (July 1, 2011). DOI: 10/df9dtd 2.

[DJ07] DONNER, C. and JENSEN, H. W. "Rendering translucent materials using photon diffusion". *Proc. EGSR*. 2007. ISBN: 978-3-905673-52-4. DOI: 10/gfz9d3 2.

[DJBJ19] DENG, X., JIAO, S., BITTERLI, B., and JAROSZ, W. "Photon surfaces for robust, unbiased volumetric density estimation". *Proc. SIGGRAPH* 38.4 (July 2019). DOI: 10/gf6rx9 2.

[DKH*14] DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., and NOVÁK, J. "Scalable realistic rendering with many-light methods". *CGF* 33.1 (Feb. 1, 2014). DOI: 10/f5twgd 2, 6.

[DWdE*08] DONNER, C., WEYRICH, T., d'EON, E., RAMAMOORTHI, R., and RUSINKIEWICZ, S. "A layered, heterogeneous reflectance model for acquiring and rendering human skin". *Proc. SIGGRAPH Asia* 27.5 (2008). DOI: 10/gfz5ts 2.

[Eva10] EVANS, L. C. *Partial differential equations*. Providence, R.I.: American Mathematical Society, 2010. ISBN: 9780821849743 0821849743 1, 3, 5.

[FHF*17] FASCIONE, L., HANIKA, J., FAJARDO, M., CHRISTENSEN, P., BURLEY, B., GREEN, B., PIEKÉ, R., KULLA, C., HERY, C., VILLEMIN, R., HECKENBERG, D., and MAZZONE, A. "Path tracing in production (Parts 1 and 2)". *ACM SIGGRAPH Courses*. Aug. 2017. DOI: 10/gfz2ck 1.

[GKDS12] GEORGIEV, I., KŘIVÁNEK, J., DAVIDOVIČ, T., and SLUSALLEK, P. "Light transport simulation with vertex connection and merging". *Proc. SIGGRAPH Asia* 31.6 (Nov. 2012). DOI: 10/gbb6q7 2, 5.

[GTGB84] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., and BATTAILE, B. "Modeling the interaction of light between diffuse surfaces". *Proc. SIGGRAPH* 18.3 (July 1984). DOI: 10/fsjssr 1.

[HCJ13] HABEL, R., CHRISTENSEN, P. H., and JAROSZ, W. "Photon beam diffusion: a hybrid Monte Carlo method for subsurface scattering". *Proc. EGSR* 32.4 (June 2013). DOI: 10/f445m4 2.

[HPB07] HAŠAN, M., PELLACINI, F., and BALA, K. "Matrix row-column sampling for the many-light problem". *Proc. SIGGRAPH* 26.3 (July 29, 2007). DOI: 10/djv68s 2.

[HPJ12] HACHISUKA, T., PANTALEONI, J., and JENSEN, H. W. "A path space extension for robust light transport simulation". *Proc. SIGGRAPH Asia* 31.6 (Nov. 1, 2012). DOI: 10/gbb6n3 2, 5.

[JB02] JENSEN, H. W. and BUHLER, J. "A rapid hierarchical rendering technique for translucent materials". *Proc. SIGGRAPH* 21.3 (July 2002). DOI: 10/fdhw4b 2.

[JC98] JENSEN, H. W. and CHRISTENSEN, P. H. "Efficient simulation of light transport in scenes with participating media using photon maps". *Proc. SIGGRAPH*. ACM Press, July 1998. DOI: 10/b64p36 2.

[Jen01] JENSEN, H. W. *Realistic Image Synthesis Using Photon Mapping*. Natick, MA, USA: AK Peters, Ltd., 2001. ISBN: 1-56881-147-0 2, 9.

[Jen96] JENSEN, H. W. "Global illumination using photon maps". *Proc. EGWR*. Vienna: Springer-Verlag, June 1996. ISBN: 978-3-211-82883-0. DOI: 10/fzc6t9 1, 2.

[JMLH01] JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., and HANRAHAN, P. "A practical model for subsurface light transport". *Proc. SIGGRAPH*. ACM Press, Aug. 2001. ISBN: 1-58113-374-X. DOI: 10/chdvp7 1, 2.

[JNSJ11] JAROSZ, W., NOWROUZEZAHRAI, D., SADEGHI, I., and JENSEN, H. W. "A comprehensive theory of volumetric radiance estimation using photon points and beams". *ACM TOG* 30.1 (Jan. 1, 2011). DOI: 10/fcdh2f 2.

[JNT*11] JAROSZ, W., NOWROUZEZAHRAI, D., THOMAS, R., SLOAN, P.-P., and ZWICKER, M. "Progressive photon beams". *Proc. SIGGRAPH Asia* 30.6 (Dec. 2011). DOI: 10/fn5xzj 2.

[JZJ08] JAROSZ, W., ZWICKER, M., and JENSEN, H. W. "The beam radiance estimate for volumetric photon mapping". *Proc. EG* 27.2 (Apr. 2008). DOI: 10/bjsfsx 2.

[Kaj86] KAJIYA, J. T. "The rendering equation". *Proc. SIGGRAPH* 20.4 (Aug. 1986). DOI: 10/cvf53j 1, 3.

[Kel97] KELLER, A. "Instant radiosity". *Proc. SIGGRAPH*. ACM Press, Aug. 1997. ISBN: 978-0-89791-896-1. DOI: 10/fqch2z 1, 2.

[KPS*14] KOERNER, D., PORTSMOUTH, J., SADLO, F., ERTL, T., and EBERHARDT, B. "Flux-limited diffusion for multiple scattering in participating media". *CGF* 33.6 (Sept. 1, 2014). DOI: 10/f6kmzs 2.

[LHW21] LEONARD, L., HÖHLEIN, K., and WESTERMANN, R. "Learning multiple-scattering solutions for sphere-tracing of volumetric subsurface effects". en. *Proc. EG* 40.2 (May 2021). DOI: 10/h2gr 2.

[LO07] LEE, R. T. and O'SULLIVAN, C. "Accelerated light propagation through participating media". *Proc. Eurographics / IEEE VGTC Conference on Volume Graphics*. The Eurographics Association, 2007. ISBN: 978-3-905674-03-3. DOI: 10/gfzq7x 2.

[LW93] LAFORTUNE, E. P. and WILLEMS, Y. D. "Bi-directional path tracing". *Proc. the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)* (Alvor, Portugal). Vol. 93. Alvor, Portugal, Dec. 1993 4.

[MPG*16] MÜLLER, T., PAPAS, M., GROSS, M., JAROSZ, W., and NOVÁK, J. "Efficient rendering of heterogeneous polydisperse granular media". *Proc. SIGGRAPH Asia* 35.6 (Nov. 2016). DOI: 10/f9cm65 2.

[MPH*15] MENG, J., PAPAS, M., HABEL, R., DACHSBACHER, C., MARSCHNER, S., GROSS, M., and JAROSZ, W. "Multi-scale modeling and rendering of granular materials". *Proc. SIGGRAPH* 34.4 (July 2015). DOI: 10/gfzndr 2.

[Mul56] MULLER, M. E. "Some continuous monte carlo methods for the Dirichlet problem". *Annals of Mathematical Statistics* 27.3 (Sept. 1956). DOI: 10/cpxd3d 1.

[MWM07] MOON, J. T., WALTER, B., and MARSCHNER, S. R. "Rendering discrete random media using precomputed scattering solutions". *Proc. EGSR*. Eurographics Association, June 2007. ISBN: 978-3-905673-52-4. DOI: 10/gfzp5n 2.

[NNDJ12a] NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. "Progressive virtual beam lights". *Proc. EGSR* 31.4 (June 2012). DOI: 10/gfzndw 2.

[NNDJ12b] NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. "Virtual ray lights for rendering scenes with participating media". *Proc. SIGGRAPH* 31.4 (July 2012). DOI: 10/gbbwk2 2.

[OBW*08] ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., and SALESIN, D. "Diffusion curves: a vector representation for smooth-shaded images". *Proc. SIGGRAPH* 27.3 (Aug. 1, 2008). DOI: 10/fj92zf 1, 8.

[PJH16] PHARR, M., JAKOB, W., and HUMPHREYS, G. *Physically Based Rendering: From Theory to Implementation*. 3rd. Cambridge, MA: Morgan Kaufmann, 2016. ISBN: 978-0-12-800645-0 1.

[Rei92] REICHERT, M. C. "A Two-Pass Radiosity Method Driven by Lights and Viewer Position". M.Sc. Thesis. Ithaca, NY: Program of Computer Graphics, Cornell University, Jan. 1992 1, 2, 9.

[Saw*21] SAWHNEY, R. et al. *fcpw*. 2021. URL: https://github.com/rohan-sawhney/fcpw 9.

[SC20] SAWHNEY, R. and CRANE, K. "Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains". *Proc. SIGGRAPH* 39.4 (2020). DOI: 10/ghw7t7 1, 2, 5.

[SSJC22] SAWHNEY, R., SEYB, D., JAROSZ, W., and CRANE, K. "Grid-free Monte Carlo for PDEs with spatially varying coefficients". *Proc. SIGGRAPH* 41.4 (July 2022). DOI: 10.1145/3528223.3530134 1, 2, 5, 9.

[Sta95] STAM, J. "Multiple scattering as a diffusion process". *Proc. EGWR*. Ed. by HANRAHAN, P. M. and PURGATHOFER, W. Springer-Verlag, 1995. DOI: 10/gfzqh2 1, 2.

[Vea97] VEACH, E. "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford University, Dec. 1997 1, 4.

[VG95a] VEACH, E. and GUIBAS, L. J. "Bidirectional estimators for light transport". *Proc. EGWR*. Springer-Verlag, 1995. ISBN: 978-3-642-87825-1. DOI: 10/gfznbh 4.

[VG95b] VEACH, E. and GUIBAS, L. J. "Optimally combining sampling techniques for Monte Carlo rendering". *Proc. SIGGRAPH*. Vol. 29. ACM Press, Aug. 1995. ISBN: 978-0-89791-701-8. DOI: 10/d7b6n4 2, 4.

[WABG06] WALTER, B., ARBREE, A., BALA, K., and GREENBERG, D. P. "Multidimensional lightcuts". *Proc. SIGGRAPH* 25.3 (July 2006). DOI: 10/dzgsz7 2.

[WFA*05] WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., and GREENBERG, D. P. "Lightcuts: a scalable approach to illumination". *Proc. SIGGRAPH* 24.3 (Aug. 2005). DOI: 10/dhp5d3 2.

[WKB12] WALTER, B., KHUNGURN, P., and BALA, K. "Bidirectional lightcuts". *Proc. SIGGRAPH* 31.4 (July 2012). DOI: 10/gfzrcx 2.