Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions

ROHAN SAWHNEY*, Carnegie Mellon University and NVIDIA, USA
BAILEY MILLER*, Carnegie Mellon University, USA
IOANNIS GKIOULEKAS†, Carnegie Mellon University, USA
KEENAN CRANE†, Carnegie Mellon University, USA

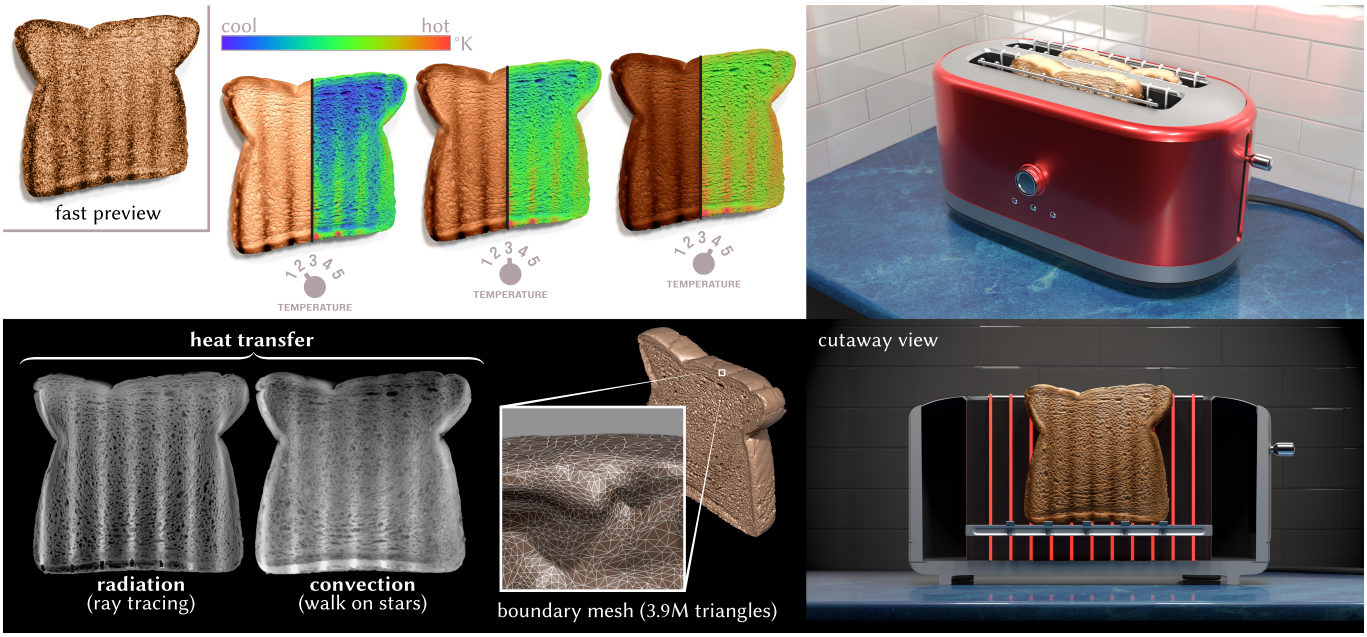


Fig. 1. The walk on stars (WoSt) method handles mixed Dirichlet and Neumann boundary conditions, enabling it to model a richer class of problems than the original walk on spheres (WoS) method. Here for instance we simulate diffusive convective heat transfer from a toaster (Dirichlet) to a piece of bread (Neumann) by solving a Laplace equation with mixed boundary conditions *(top and bottom right)*, complementing the radiative transfer computed via ray tracing *(bottom left)*. As with ray tracing, we can simulate directly on the full high-resolution data *(bottom center)* without generating a volume mesh or forming a global stiffness matrix. Since results are progressive, we can get a preview of how the toast will look faster than it takes to toast a real piece of bread *(top left)*.

Grid-free Monte Carlo methods based on the *walk on spheres (WoS)* algorithm solve fundamental partial differential equations (PDEs) like the Poisson equation without discretizing the problem domain or approximating functions in a finite basis. Such methods hence avoid aliasing in the solution, and evade the many challenges of mesh generation. Yet for problems with complex geometry, practical grid-free methods have been largely limited to basic Dirichlet boundary conditions. We introduce the *walk on stars (WoSt)* algorithm, which solves linear elliptic PDEs with arbitrary mixed Neumann and Dirichlet boundary conditions. The key insight is that one can efficiently simulate reflecting Brownian motion (which models Neumann conditions) by replacing the balls used by WoS with *star-shaped* domains. We identify such domains via the closest point on the visibility silhouette, by simply augmenting a standard bounding volume hierarchy with normal information. Overall, WoSt is an easy modification of WoS, and retains the many attractive features of grid-free Monte Carlo methods such as progressive and view-dependent evaluation, trivial parallelization, and sublinear scaling to increasing geometric detail.

CCS Concepts: • **Mathematics of computing** → **Partial differential equations**; **Integral equations**; **Probabilistic algorithms**.

Additional Key Words and Phrases: Monte Carlo methods, walk on spheres

ACM Reference Format:
Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 20 pages. https://doi.org/10.1145/3592398

*and † indicate equal contribution.

Authors' addresses: Rohan Sawhney, rohansawhney@cs.cmu.edu, Carnegie Mellon University and NVIDIA, USA; Bailey Miller, bmmiller@andrew.cmu.edu, Carnegie Mellon University, USA; Ioannis Gkioulekas, igkioule@cs.cmu.edu, Carnegie Mellon University, USA; Keenan Crane, kmcrane@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA.

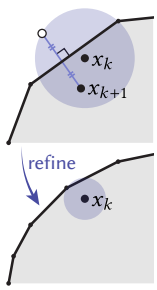
0730-0301/2023/8-ART1
https://doi.org/10.1145/3592398

1 INTRODUCTION

Systems throughout nature—and in our everyday lives—exhibit vast geometric and material complexity (Figures 1, 5). Although Monte Carlo methods have been enormously successful for photorealistic rendering, there remains a large divide between our ability to *visualize* and *simulate* natural phenomena. To make complex simulation problems feasible, a common approach is to simplify the original model, *e.g.*, via coarsening, homogenization, or learning. Yet in many physical systems, subtle differences in fine-scale geometry have a major impact on large-scale behavior. Hence, even if the end goal is to use a lower-dimensional model, one must have tools that can accurately fit such a model, starting from the original geometry. More broadly, models of real physical systems must often integrate disparate phenomena (say, light transport and heat transfer), which classically demand very different computational tools (Figure 20). For all these reasons, we tend to shy away from simulating the natural world at its original level of complexity, either by making gross approximations—or by tempering our ambition.

Prompted by the disparity between rendering and simulation, Sawhney and Crane [2020] advocate the use of *grid-free Monte Carlo methods* to solve partial differential equations (PDEs) on domains of extreme geometric complexity. Such methods need not discretize the problem domain (as in finite difference methods), or even pick a finite basis of functions (as in finite element and boundary element methods). Instead, like ray tracing methods, they require only pointwise access to geometry via closest point queries. This setup makes grid-free methods especially attractive in simulation scenarios where solution values or derivatives (*e.g.*, forces) need only be evaluated at a few key points of interest, rather than densely over the entire domain. Yet grid-free Monte Carlo methods have one major shortcoming: since their development in the 1950s, they have not been extended to many PDEs beyond the original Dirichlet Laplace problem studied by Muller [1956]. In this paper we take a basic but important step forward by developing a practical strategy for incorporating *Neumann boundary conditions*—which are a basic component of virtually every real physical model.

Basic Approach. The original WoS method solves Dirichlet problems by simulating random walks that ultimately get absorbed into the boundary (Figure 3, *top left*). Rather than simulate many small steps of an isotropic *Brownian motion* (Figure 2), this process is greatly accelerated by sampling the next point from the largest empty ball around the current point (Section 3.4). To model Neumann conditions, one must also simulate *reflecting* random walks that "bounce" off the boundary (Figure 2, *top right & bottom*) [Grebenkov 2006, 2007]. In a half space, reflecting walks amount to just taking the absolute value of Brownian motion in one coordinate direction. Hence, for polyhedral domains, a naïve strategy for simulating reflections would be to sample the largest ball that intersects only a single boundary face, and perform a reflection across the boundary plane if the sampled point falls outside the domain. However, the efficiency of this strategy quickly drops as the boundary mesh is refined.



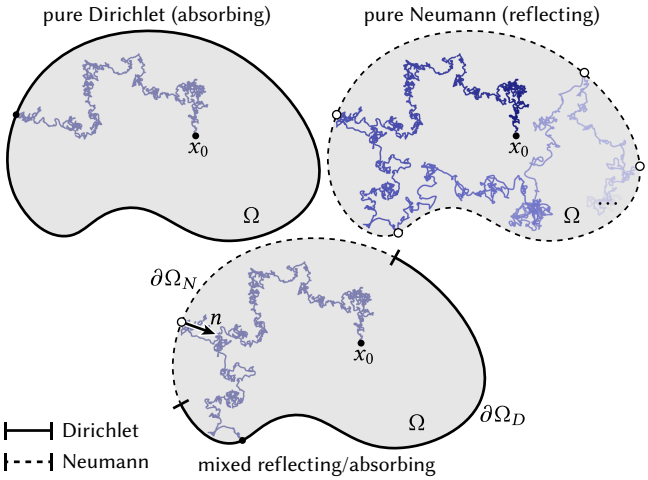


Fig. 2. A Brownian random walk terminates when it hits an absorbing Dirichlet boundary $\partial\Omega_D$ (*top left*), but is pushed back into the domain along the inward normal to a reflecting Neumann boundary $\partial\Omega_N$ (*bottom*). The walk continues forever if the boundary is only reflecting (*top right*).

Our strategy, which we call *walk on stars (WoSt)*, is both more efficient and more general. In short, we identify a large star-shaped region around the current point, and sample a point on its boundary by picking a random direction (Figure 3, Section 4). This strategy can be viewed as a Monte Carlo estimator for the *boundary integral equation (BIE)* of a Laplace problem (Section 3). WoSt hence takes steps that are independent of the level of tessellation, and are typically much larger than the empty balls used by WoS (Sections 3.4.2 & 6.3). Moreover, this strategy applies to domains that are not polyhedral, and unlike past WoS-based strategies is not limited to convex domains (Section 4.4). The only question that must be answered is: how do we find star-shaped regions? In this paper we propose one strategy, using the *visibility silhouette*, which is easy to implement efficiently without much overhead (Section 5). Fundamentally, however, the WoSt approach relies only on the use of star-shaped regions—not on any particular method used to compute them. Importantly, WoSt requires only few modifications to an existing WoS implementation, and achieves sublinear scaling to geometric detail using essentially the same data structures as WoS. It thus provides the same advantages as WoS (progressive evaluation, trivial parallelization, robustness to defective geometry, *etc.*), while being applicable to a broader class of problems.

Limitations. For problems where boundary conditions are mostly Neumann, WoSt can take very long walks: it must reflect at the Neumann boundary, and can terminate only on the Dirichlet boundary (Figure 2, *bottom*). This situation is analogous to path tracing a scene where all materials have albedo one—such as a room of perfect mirrors (Figure 22). Likewise, pure Neumann conditions are uncommon in many real physical scenarios—corresponding to, *e.g.*, perfect insulators. Support for more general *Robin boundary conditions* would hence both improve modeling realism and increase efficiency, as more walks could terminate early (Section 7). In concurrent work, we also present a *boundary value caching (BVC)* strategy

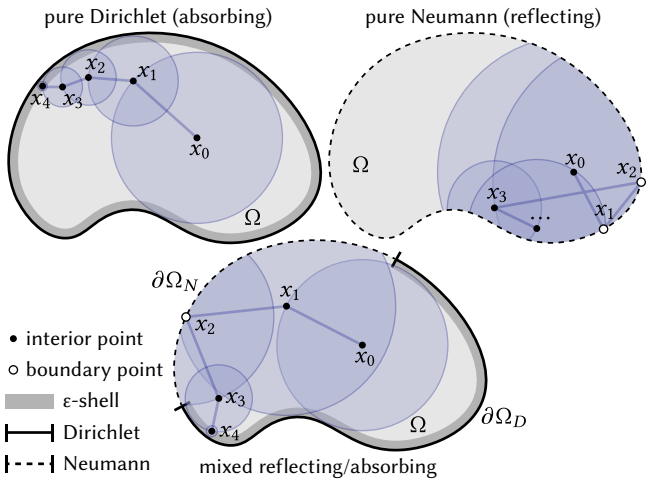


Fig. 3. *Top left:* Walk on spheres simulates a Brownian random walk inside an absorbing Dirichlet boundary $\partial\Omega_D$, repeatedly jumping to a random point on the largest sphere centered at the current walk location. The walk is terminated when it enters an ε-shell $\partial\Omega_D^\varepsilon$ around the boundary. *Top right and bottom:* Our walk on stars algorithm generalizes WoS to domains with a reflecting Neumann boundary $\partial\Omega_N$, using a sphere that can contain a subset of the reflecting boundary inside it. The next location of the walk is determined by intersecting a ray with a randomly-sampled direction against the sphere and the visible portions of $\partial\Omega_N$ it contains, picking the first hit point. The walk continues forever if the boundary is only reflecting (*top right*), and terminates inside $\partial\Omega_D^\varepsilon$ otherwise (*bottom*).

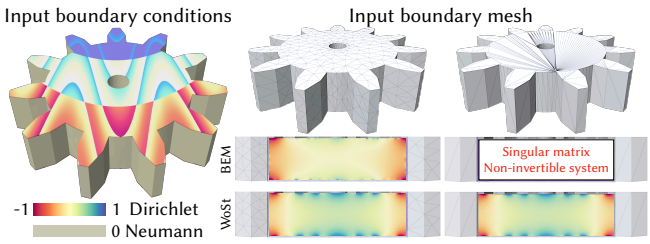


Fig. 4. Finite element technology like BEM suffers from large global errors in the PDE solution without significant mesh refinement due to local aliasing of boundary data, and can fail completely on domains with irregular elements (*middle row*). In contrast, our method solves PDEs without any aliasing artifacts irrespective of tesselation quality as it decouples problem inputs from the boundary representation (*bottom row*). It can also handle source terms without requiring a volumetric mesh.

that greatly amortizes the cost of long walks, even for Neumann-dominated problems [Miller et al. 2023]. Here we focus purely on enriching boundary conditions in grid-free Monte Carlo methods. WoSt is otherwise limited to the same class of PDEs as WoS—namely linear elliptic PDEs such as the Poisson equation (Equation 1). However, boundary integral formulations are readily available for the Helmholtz equation [Hunter and Pullan 2001, Chapter 3], linear elasticity [Hunter and Pullan 2001, Chapter 4] and even fluids (via *stochastic* integral equations) [Busnello et al. 2005; Rioux-Lavoie et al. 2022]). We hence expect that WoSt will provide a valuable starting point for handling more general boundary conditions in these broader problems.

2 RELATED WORK

We briefly discuss alternative strategies for solving PDEs; Sawhney and Crane [2020, Sections 1 and 7] and Sawhney et al. [2022, Section 7] describe in depth the tradeoffs between grid-free Monte Carlo and conventional, discretization-based methods like FEM, BEM, and meshless FEM, and provide extensive numerical comparisons.

Walk on Spheres. Monte Carlo estimators for linear elliptic PDEs with Dirichlet boundaries, such as WoS, date back to Muller [1956], and have recently received renewed interest following their introduction to graphics by Sawhney and Crane [2020]. There have been rapid advances along two main thrusts: first, increasing efficiency through optimized implementation [Mossberg 2021], bidirectional formulations [Qi et al. 2022], and sample caching techniques [Miller et al. 2023]. Second, increasing generality through new estimators that can solve PDEs in infinite domains [Nabizadeh et al. 2021] or with variable coefficients [Sawhney et al. 2022], that simulate fluid equations [Rioux-Lavoie et al. 2022], and that enable differentiability for inverse problems [Yılmazer et al. 2022]. Our focus is to further push the envelope along the second thrust by developing the first Monte Carlo estimator that can solve Neumann and mixed-boundary problems on general, nonconvex domains while providing a performance-bias tradeoff comparable to classic WoS.

Grid-based PDE Solvers. Like WoS, WoSt does not require a volume mesh or background grid—offering critical advantages relative to grid-based methods such as finite differences and finite element methods. Namely, grid-free estimators provide output sensitivity (*i.e.*, the ability to focus computation only on regions of interest), progressive evaluation (*i.e.*, the ability to preview solutions as they improve), support for general geometric representations (*e.g.*, meshes, implicit surfaces, or instanced geometry), trivial parallelization, and excellent scaling with increasing geometric detail. Whereas grid-based methods are often faster on simple models that can be easily meshed, WoSt easily handles problems whose real-world complexity places them out of reach for traditional techniques—at least not without critical sacrifices in accuracy (Figure 5). On the flip side, grid-based methods can share global information about solution values (via a coupled linear solve), whereas classic grid-free Monte Carlo methods must make independent estimates at each point—leading to fairly redundant computation. As we show in concurrent work [Miller et al. 2023], information sharing via sample reuse can dramatically accelerate grid-free methods as well.

Boundary Element Methods. Among grid-based techniques, the boundary element method (BEM) is most closely related to WoSt, as it too solves boundary integral equations (Section 3.3). Whereas grid-free Monte Carlo can solve problems with volumetric data (*e.g.*, source terms or variable PDE coefficients), BEM can do so only when coupled with a secondary solver such as FEM or radial basis function techniques [Coleman et al. 1991; Hunter and Pullan 2001; Partridge et al. 2012]. As a result, BEM inherits the shortcomings of the grid-based approaches discussed above. Even in the absence

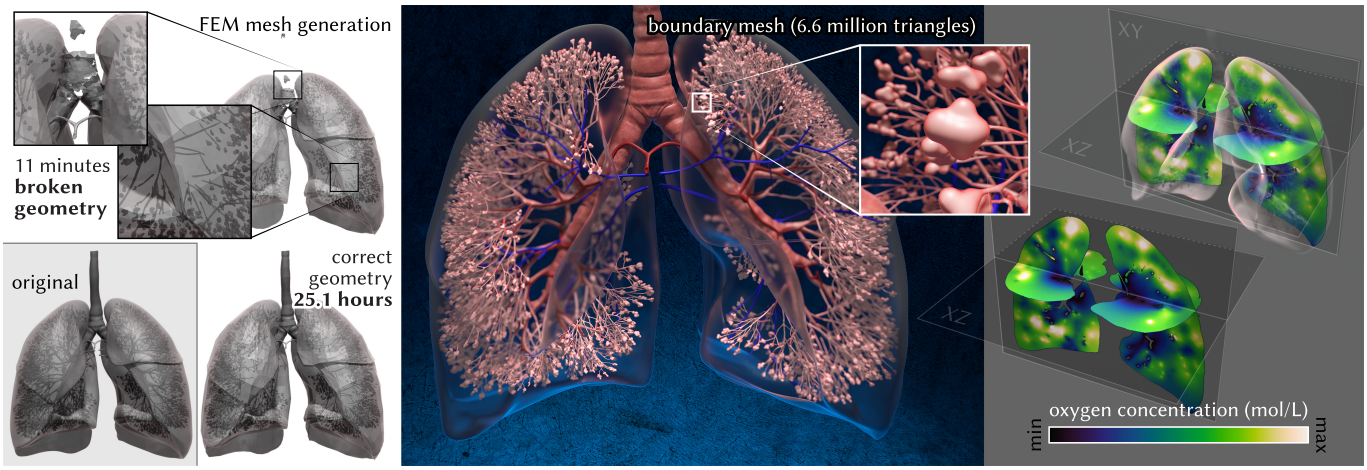


Fig. 5. Where will oxygen flow at the beginning of a breath? Here we use walk on stars to simulate gas exchange via *Laplacian transport* [Grebenkov 2006], directly on a detailed lung model with thin features *(center)*. The output-sensitivity of our method enables us to focus computation purely on the slice planes used for visualization *(right)*, rather than needing to solve over the whole domain. Attempting simulation on the same model using FEM leads to significant problems, either because meshing destroys critical details *(top left)*, or takes more than 25 hours to produce a mesh that captures the original geometry *(bottom left)*. In contrast, walk on stars provides near-immediate feedback that reliably reflects the true geometry and solution.

of volumetric terms, BEM discretizes the domain boundary to construct, then invert, a *dense* linear system. Hence, computational complexity scales quadratically with geometric detail, requiring specialized techniques like *hierarchical matrix approximation* to scale to large geometries [Hackbusch 2015]. Finally, BEM must encode boundary conditions via the discretized boundary representation. The resulting aliasing can yield global artifacts or singular systems—and in some cases an inability to even produce a solution (Figure 4).

Monte Carlo Methods for Neumann Problems. Neumann conditions introduce additional complexity into random walk methods due to the need to reflect, rather than absorb, random walks at the boundary. Stochastic differential equation (SDE) integrators such as Euler-Maruyama [Higham 2001] use discretized random walks, modeling reflections by projecting walks back into the domain [Costantini et al. 1998]. Such reflections introduce discretization error and greatly increase walk length (Figure 17), further exacerbating the challenges already associated with SDE integrators for Dirichlet problems: a fixed step size means either very slow runtimes (due to a large number of steps inside the domain) or large estimation bias (due to discretization error). Unlike SDE integrators, WoS (for pure Dirichlet problems) and WoSt (for mixed-boundary problems) use adaptive—and generally large—step sizes, avoid discrete approximation, and thus offer a much more favorable runtime-to-bias tradeoff (see Section 6.3 and Sawhney et al. [2022, Section 7.1.3]).

For pure Dirichlet problems, WoSt reduces to the standard WoS estimator (Section 3.4), *i.e.*, it uses the largest sphere that does not intersect the boundary. Thus, walks converge quickly to the Dirichlet boundary, where they are absorbed. For mixed-boundary problems, augmented WoS methods use finite-difference approximations of Neumann conditions [Mascagni and Simonov 2004; Maire and Tanré 2013; Zhou et al. 2017]. However, near the Neumann boundary sphere sizes become very small, and walks behave much as in SDE integrators (Figure 6): numerous consecutive reflections result in slow runtimes (due to long walk lengths) and large estimation bias (due to accumulation of discretization error). In contrast, WoSt uses non-spherical, star-shaped regions that can contain large pieces of the Neumann boundary. Unlike WoS, it hence continues to take large steps near the Neumann boundary, while avoiding discretization. These differences translate into order-of-magnitude improvements in both runtime performance and estimation accuracy (Section 6.3).

WoSt builds on the techniques of Simonov [2008] and Ermakov and Sipin [2009], which also sample successive random walk locations on regions that can contain the Neumann boundary. However, these techniques are restricted to problems with convex Neumann boundaries. By sampling walk locations using star-shaped regions (Section 4.4), WoSt can solve problems on general nonconvex domains, while also being significantly faster for convex ones.

Ding et al. [2022] recently proposed a hybrid WoS-BEM solver to handle Neumann problems. The hybrid nature of their solver means that it sacrifices key properties of Monte Carlo-only techniques, such as progressive evaluation, robustness to poorly tessellated surface geometry, trivial parallelization, and output sensitivity.

Finally, the walk on boundary (WoB) method is an alternative Monte Carlo approach for mixed-boundary problems. WoB recursively evaluates single and double layer potentials by tracing rays that reflect off the boundary [Sabelfeld and Simonov 2013]. Similar to Simonov [2008] and Ermakov and Sipin [2009], this method currently works reliably only on problems with convex Neumann boundaries, suffering from high variance and bias in nonconvex domains—Section 4.4.1 explains why all three of these methods struggle with nonconvex domains; see also Section 6.2 for numerical experiments. In concurrent work, Sugimoto et al. [2023] introduce WoB to the graphics community—their accessible overview and improvements to the technique likely open avenues to further, more efficient estimators that span the space between WoSt and WoB.

Accelerated Distance and Silhouette Queries. Much like path tracing, both WoS and WoSt require iterative applications of a basic set of geometric queries and sampling operations. WoS uses closest point queries to the Dirichlet boundary to determine random walk step sizes. These queries can be accelerated with a bounding volume hierarchy (BVH) [Intel 2013; Sawhney 2021; Krayer and Müller 2021], achieving sublinear scaling relative to geometric detail. WoSt additionally uses *closest silhouette point queries* to the Neumann boundary to define star-shaped regions. As we show in Section 5, these queries can likewise be accelerated using a BVH augmented to include orientation information for the geometry inside each node. We use the *spatialized normal cone hierarchy (SNCH)* of Johnson and Cohen [2001], which has previously been used to accelerate culling of back-facing geometry, local minimum distance queries, and *next-event estimation* (NEE) for rendering scenes with many lights [Estevez and Kulla 2018]. More recently, silhouette queries have become a critical subroutine for differentiable Monte Carlo rendering [Li et al. 2018], where they are used to estimate radiometric integrals arising from visibility discontinuities. Thus, acceleration schemes for silhouette queries in these emerging rendering algorithms are likely to benefit WoSt, and vice versa.

3 BACKGROUND

We first review linear elliptic PDEs and their boundary integral equation (BIE) representation, which serves as the starting point for our WoSt estimator in Section 4. We also demonstrate how to derive the WoS estimator from the BIE, to aid comparison with WoSt.

3.1 Notation

For any set $A \subset \mathbb{R}^N$, we use ∂A and $|A|$ for its boundary and volume, *resp.* We use p^A to denote a probability density function on A, and write $x \sim p^A$ for a random point $x \in A$ drawn from p^A. For any point $z \in \partial A$, we use n_z for the unit outward normal at z. We say ∂A is *convex* at z if all principal curvatures are positive (relative to n_z), and nonconvex otherwise. For any point $x \in A$, $\overline{x}(\partial A) \coloneqq \operatorname{argmin}_{y \in \partial A} \|y - x\|$ is the closest point to x on ∂A, and $B(x, r)$ is a ball with center x and radius r. We drop arguments for $\overline{x}$ and B when the context is clear. We define the *ε-shell* around ∂A as $\partial A^\varepsilon \coloneqq \{x \in \Omega : \|x - \overline{x}(\partial A)\| < \varepsilon\}$. Finally, we use Δ for the negative-semidefinite Laplace operator on $\mathbb{R}^N$.

3.2 Linear Elliptic Equations

Linear elliptic partial differential equations have broad utility in geometry processing, simulation, graphics, and scientific computing in general. A prototypical example is the *Poisson equation*

$$\begin{aligned} \Delta u(x) &= f(x) && \text{on } \Omega, \\ u(x) &= g(x) && \text{on } \partial\Omega_D, \\ \tfrac{\partial u(x)}{\partial n_x} &= h(x) && \text{on } \partial\Omega_N, \end{aligned} \tag{1}$$

which describes, *e.g.*, the steady-state temperature distribution over a domain $\Omega \subset \mathbb{R}^N$. Here $u : \Omega \to \mathbb{R}$ is the unknown solution, and $f : \Omega \to \mathbb{R}$ is a *source term*, analogous to a heat source or sink. We partition the boundary $\partial\Omega$ into a subset $\partial\Omega_D$ where the solution has *Dirichlet boundary conditions*, *i.e.*, prescribed values $g : \partial\Omega_D \to \mathbb{R}$, and a subset $\partial\Omega_N$ where it has *Neumann boundary conditions*, *i.e.*, prescribed derivatives $h : \partial\Omega_N \to \mathbb{R}$ in the normal direction n. Either subset can be empty, in which case we say the equation has *pure* Dirichlet or Neumann conditions. A function u is *harmonic* if it satisfies Equation 1 for $f = 0$, *i.e.*, if $\Delta u = 0$ on Ω.

A *screened Poisson equation* adds a constant *absorption term* σu, $\sigma \in \mathbb{R}_{>0}$, modeling a medium that dampens or "cools" the solution:

$$\Delta u(x) - \sigma u(x) = f(x) \quad \text{on } \Omega, \tag{2}$$

subject to the same boundary conditions as in Equation 1. One can form more general linear elliptic equations by adding variable diffusion, drift and absorption coefficients to Equation 2—see Sawhney et al. [2022, Section 2.2] for a detailed exposition. For simplicity our exposition focuses on the (screened) Poisson equation, though much of the material presented here applies to more general linear PDEs (see Section 7 for further discussion).

3.2.1 Green's Function. A *Green's function* captures the influence of the source term f—in particular, it describes the solution when the source is a Dirac delta distribution δ_x centered at a single point $x \in \Omega$ [Evans 1998; Duffy 2015]. For instance, in the case of Equation 1 the Green's function $G^\Omega(x, y)$ is the solution to the Poisson equation $\Delta u(y) = \delta_x(y)$. In general, a Green's function will depend on the shape of the domain Ω and the choice of boundary conditions. Typically, Green's functions are not available in closed-form—however, explicit expressions are available for important special cases, *e.g.*, the *free-space Green's function* $G^{\mathbb{R}^N}$ on $\Omega = \mathbb{R}^N$, and the Green's function G^B for a ball $\Omega = B$ with zero-Dirichlet boundary conditions (see Appendix A.1). The WoS and WoSt methods effectively provide a bridge between closed-form Green's functions on special domains, and solutions to PDEs on more general domains.

3.2.2 Poisson Kernel. The *Poisson kernel* likewise captures the influence of the boundary conditions on the solution, *e.g.*, when the function g is a Dirac delta distribution δ_x centered on a single boundary point $x \in \partial\Omega$. At any point $y \in \Omega$ with associated normal n_y, it can be expressed as the normal derivative of a Green's function:

$$P^\Omega(x, y) \coloneqq \frac{\partial G^\Omega(x, y)}{\partial n_y}. \tag{3}$$

As with Green's functions, common Poisson kernels are known explicitly in free space and for a ball (see Appendix A.2).

3.3 Boundary Integral Equation

The solution u to a linear PDE can be expressed via a *boundary integral* involving the associated Green's function and Poisson kernel. Assume for now that Ω is a watertight domain with smooth boundary $\partial\Omega$, and let x be an evaluation point on the interior of Ω. We first multiply the Poisson equation in Equation 1 with its Green's function G^Ω and integrate over Ω to get

$$0 = \int_\Omega G^\Omega(x, y)\ \Delta u(y)\, \mathrm{d}y \ - \ \int_\Omega G^\Omega(x, y)\ f(y)\, \mathrm{d}y. \tag{4}$$

Applying integration by parts to the first integral, we have

$$0 = \int_{\partial\Omega} G^\Omega(x, z)\ \frac{\partial u(z)}{\partial n_z}\, \mathrm{d}z \ - \ \int_\Omega \nabla G^\Omega(x, y)\ \cdot \nabla u(y)\, \mathrm{d}y \\ - \ \int_\Omega G^\Omega(x, y)\ f(y)\, \mathrm{d}y. \tag{5}$$

Applying integration by parts again to the second integral in Equation 5 and rearranging terms then yields

$$\int_\Omega u(y)\ \Delta G^\Omega(x,y)\,\mathrm{d}y = \int_{\partial\Omega} P^\Omega(x,z)\ u(z)\ -\ G^\Omega(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z + \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y. \tag{6}$$

From the definition $\Delta G^\Omega(x,y) = \delta_x^\Omega(y)$ we arrive at

$$u(x) = \int_{\partial\Omega} P^\Omega(x,z)\ u(z)\ -\ G^\Omega(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z + \int_\Omega G^\Omega(x,y)\ f(y)\,\mathrm{d}y. \tag{7}$$

This equation determines the solution u at x entirely through the solution values $u(z)$ and normal derivatives $\partial u(z)/\partial n_z$ on the boundary $\partial\Omega$, and the source values $f(y)$ inside the domain Ω. From Equation 1, the Dirichlet and Neumann parts of the boundary have prescribed values g and h, *resp.*, while f is specified inside the domain. To use Equation 7, we must then determine unknown solution values $u(z)$ on the Neumann boundary $\partial\Omega_N$, and unknown derivative values $\partial u(z)/\partial n_z$ on the Dirichlet boundary $\partial\Omega_D$.

3.3.1 General Setting. In practice, Equation 7 cannot be used directly since the Green's function and Poisson kernel for an arbitrary domain Ω are unknown. Fortunately, this equation can be generalized to the *boundary integral equation* (BIE) [Costabel 1987, Section 2] where these functions are no longer tied to the domain Ω. Instead one may use, *e.g.*, the closed-form Green's function and Poisson kernel for a ball, or for $\mathbb{R}^N$. Moreover, while we ultimately seek a solution on Ω, the BIE applies to arbitrary subdomains in Ω:

Boundary Integral Equation

For any two sets $A, C \subset \Omega$, and for any point $x \in \mathbb{R}^N$, the solution to the Poisson equation in Equation 1 satisfies:

$$\alpha(x)\ u(x) = \overbrace{\int_{\partial A} P^C(x,z)\ u(z)\ -\ G^C(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z}^{\text{boundary}} + \overbrace{\int_A G^C(x,y)\ f(y)\,\mathrm{d}y}^{\text{interior}}, \tag{8}$$

where

$$\alpha(x) \coloneqq \begin{cases} 1, & x \in A, \\ 1/2, & x \in \partial A, \\ 0, & x \notin A. \end{cases} \tag{9}$$

Hunter and Pullan [2001, Chapter 3.3] provide a derivation. For screened Poisson equations we instead use the Green's function and Poisson kernel from Appendix A.2. The BIE can be extended further—for instance, if ∂A is a non-smooth curve in the plane, then $\alpha = 1 - \theta/2\pi$ at a corner with interior angle θ. Likewise, in Appendix B we generalize the BIE to double-sided boundary conditions. To keep things simple, we will assume in Section 4 that ∂A is smooth, letting $\alpha = 1/2$ at all boundary points.

Both BEM and WoS can be interpreted as methods for solving the BIE, for different choices of sets A and C. Throughout we highlight boundary terms (in blue) and interior terms (in gray) to make the correspondence with Equation 8 clear.

3.3.2 Boundary Element Method. BEM integrates Equation 8 over the PDE domain ($A = \Omega$) using free-space kernels ($C = \mathbb{R}^N$). BEM does not directly support source terms f, leading to the integral

$$\alpha(x)\ u(x) = \overbrace{\int_{\partial\Omega} P^{\mathbb{R}^N}(x,z)\ u(z)\ -\ G^{\mathbb{R}^N}(x,z)\ \frac{\partial u(z)}{\partial n_z}\,\mathrm{d}z}^{\text{boundary}}. \tag{10}$$

To determine the unknown data u on $\partial\Omega_N$ and $\partial u/\partial n$ on $\partial\Omega_D$, BEM uses a finite basis of functions (associated with mesh nodes on a discretized boundary) to solve a dense linear system—resulting in the tradeoffs discussed in Section 2.

3.3.3 Walk on Spheres. WoS instead integrates the BIE over a ball $B(x,r) \subset \Omega$ centered at x, adopting kernels from the ball ($A = C = B(x,r)$). At points $z \in \partial B$, these kernels simplify to $G^B(x,z) = 0$ and $P^B(x,z) = 1/|\partial B|$ (Appendix A.1), yielding an integral

$$u(x) = \overbrace{\frac{1}{|\partial B(x,r)|}\int_{\partial B(x,r)} u(z)\,\mathrm{d}z}^{\text{boundary}} + \overbrace{\int_{B(x,r)} G^B(x,y)\ f(y)\,\mathrm{d}y}^{\text{interior}}. \tag{11}$$

This setup greatly simplifies the BIE by eliminating dependence on $\partial u/\partial n$ (hence avoiding the need for any branching estimates). Unlike BEM, the source term f is accounted for, and one does not need to discretize the domain boundary $\partial\Omega$ nor solve a global system of equations. Instead, the solution is evaluated by recursively using Equation 11 to estimate $u(z)$ on the ball boundary ∂B, leading to the WoS algorithm (Section 3.4.2).

3.4 Walk on Spheres Estimator

WoS is a *Monte Carlo estimator* for Equation 11, which means that it approximates the boundary and interior integrals using random samples of the integrands. WoSt follows the same basic recipe, but for a different version of the BIE. Before discussing these estimators in detail, we first review Monte Carlo integration [Fishman 2006], which is the numerical foundation for both methods.

3.4.1 Monte Carlo Integration. For an L^1-integrable function $\phi : A \to \mathbb{R}$, the Monte Carlo method approximates the integral

$$I \coloneqq \int_A \phi(x)\,\mathrm{d}x \tag{12}$$

using the sum

$$\widehat{I}_N \coloneqq \frac{1}{N}\sum_{n=1}^{N} \frac{\phi(x_n)}{p^A(x_n)}, \quad x_n \sim p^A, \tag{13}$$

where x_n are independent samples randomly drawn from a probability density p^A that is nonzero on the support of ϕ. An estimator is *unbiased* if its expected value equals the true value, $\mathbb{E}[\widehat{I}_N] = I$. We can quantify the accuracy of an estimator using its expected squared error $\mathbb{E}[(\widehat{I}_N - I)^2]$, which for an unbiased estimator equals its *variance* $\mathrm{Var}[\widehat{I}_N] \coloneqq \mathbb{E}[(\widehat{I}_N - \mathbb{E}[\widehat{I}_N])^2]$. Assuming independent

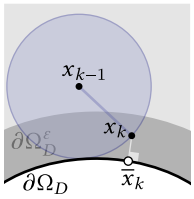
samples x_n, variance goes to zero at a rate $O(1/N)$ [Pharr et al. 2016, Chapter 13]. We express all estimators in this paper as *single-sample estimates* $\widehat{I}$, dropping the subscript $N = 1$ for brevity. Averaging these estimates over many trials improves accuracy.

3.4.2 WoS Estimator. A recursive single-sample estimate for Equation 11 at a point $x_k \in \Omega$ is given by

$$\widehat{u}(x_k) := \underbrace{\frac{1}{|\partial B(x_k)|}\frac{\widehat{u}(x_{k+1})}{p^{\partial B(x_k)}(x_{k+1})}}_{\text{boundary}} + \underbrace{\frac{G^{B(x_k)}(x_k, y_{k+1}) f(y_{k+1})}{p^{B(x_k)}(y_{k+1})}}_{\text{interior}}, \quad (14)$$

where $x_{k+1} \in \partial B$ and $y_{k+1} \in B$ are sampled from probability densities $p^{\partial B}$ and p^B, *resp.* Typical choices are the uniform density $p^{\partial B} := 1/|\partial B|$, and the normalized Green's function $p^B := G^B(x_k, y_{k+1})/|G^B(x_k)|$, where $|G^B(x_k)|$ is the integral of G^B over $B(x_k)$ (see Appendix A and Sawhney and Crane [2020, Section 4.2]). Recursive evaluation of Equation 14 determines a *random walk* on the points $x_0 \to x_1 \to \ldots$, where each point x_k sits on a sphere centered at the previous point x_{k-1}—hence the name *walk on spheres* (Figure 3). This walk terminates if $u(x_k)$ is known; otherwise, the process repeats. To take big steps, WoS typically uses the largest sphere centered at x_k and contained entirely in Ω (*i.e.*, $r = \|x - \overline{x}\|$).

3.4.3 Boundary Conditions for WoS. How a walk terminates depends on the particular boundary conditions, enumerated below.



Pure Dirichlet Conditions. For pure Dirichlet conditions ($\partial\Omega = \partial\Omega_D$), walks terminate in the ε-shell $\partial\Omega_D^\varepsilon$ and the solution estimate is set to the boundary value g at the closest point $\overline{x}_k$, *i.e.*, $\widehat{u}(x_k) := g(\overline{x}_k)$. Terminating walks in the shell introduces negligible bias of order $O(1/\log \varepsilon)$ [Binder and Braverman 2012]; Sawhney and Crane [2020, Figure 14] show experimentally that shrinking ε has little impact on runtime cost. WoSt likewise uses an ε-shell to absorb walks near the Dirichlet boundary, reducing to WoS for pure Dirichlet problems.

Mixed Dirichlet-Neumann Conditions. The standard WoS approach for mixed boundary problems [Mascagni and Simonov 2004] also performs a random walk as above, again terminating on $\partial\Omega_D^\varepsilon$. If the walk ever reaches a point $\widetilde{x}_k$ in the ε-shell $\partial\Omega_N^\varepsilon$ around the Neumann boundary, then the Neumann value h at the closest point $\overline{x}_k \in \partial\Omega_N$ is approximated via finite differences, *e.g.*,

$$h(\overline{x}_k) \approx \frac{u(\overline{x}_k + \zeta n_{\overline{x}_k}) - u(\overline{x}_k)}{\zeta}, \quad (15)$$

where $\zeta > \varepsilon$ is a constant. The solution estimate at x_k is then

$$\widehat{u}(x_k) := \widehat{u}(\overline{x}_k + \zeta n_{\overline{x}_k}) - \zeta h(\overline{x}_k) \approx u(\overline{x}_k). \quad (16)$$

In other words, $-\zeta h(\overline{x}_k)$ is added to the running estimate, and the walk continues as usual from the point $\overline{x}_k + \zeta n_{\overline{x}_k}$ obtained by nudging $\overline{x}_k$ back into the domain by a fixed distance ζ along the inward unit normal (Figure 6, *top*). Mascagni and Simonov [2004] call this procedure a *boundary reflection*; Maire and Tanré [2013] and Zhou et al. [2017] provide more sophisticated approximations using higher-order differences.

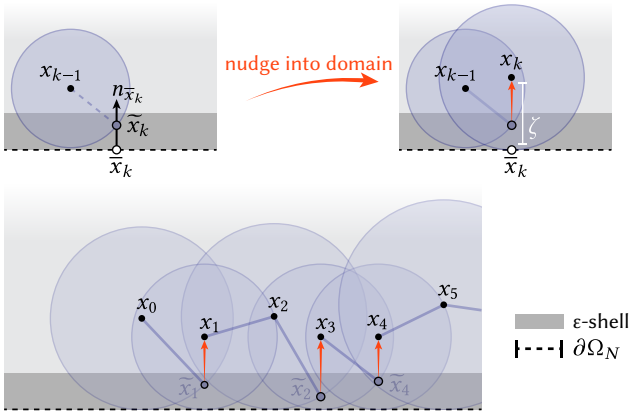


Fig. 6. To simulate reflecting random walks with WoS, a standard approach [Mascagni and Simonov 2004; Maire and Tanré 2013] is to offset a walk that approaches the Neumann boundary back into the domain by a fixed distance along the inward normal n to the boundary (*top*). This approach introduces discretization error into the reflecting walk simulation. Moreover, the resulting walks have a tendency to cling to the boundary as they are naturally attracted to it, leading to long walk lengths (*bottom*).

Unfortunately, such reflections are often impractical for problems with a large Neumann boundary $\partial\Omega_N$: the finite difference approximation introduces significant bias if ζ is much larger than ε—yet if ζ is only slightly larger than ε, random walks "stick" to $\partial\Omega_N^\varepsilon$, taking many small steps before escaping toward the interior (Figure 6, *bottom*). Figure 16 shows that, in practice, boundary reflections yield both slow runtime and large accumulated bias. WoSt avoids these issues by considering larger spheres that contain the Neumann boundary (Section 4), greatly improving both accuracy and efficiency.

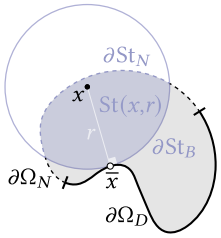
Pure Neumann Conditions. The solution to a Poisson equation with pure Neumann boundary conditions is determined only up to an additive constant. From the random walk perspective, there is no Dirichlet boundary to terminate on, hence contributions from h and f accumulate forever. However, shorter walks tend to resolve high-frequency details in the solution, whereas the contribution from independent longer walks is more spatially uniform (see Figure 18). Based on this observation, Maire and Tanré [2013] describe a WoS estimator that stops the simulation once walks become longer than a certain length, pinning an additive constant to the solution. For WoSt we instead apply *Tikhonov regularization*, which makes the solution unique by adding a small absorption term σu to the PDE (resulting in a screened Poisson equation). In particular, we switch to this PDE when a walk gets longer than a user-specified length (Figure 19), which adds a small but controlled amount of bias. We then terminate walks via *Russian roulette* [Pharr et al. 2016, Section 13.7], using a termination probability proportional to the Poisson kernel of a screened Poisson equation (Equation 39).

4 THE WALK ON STARS ALGORITHM

In this section, we develop *walk on stars* (WoSt), a recursive estimator for mixed boundary-value problems like Equation 1. Like WoS, WoSt takes large steps inside the domain Ω to quickly reach the Dirichlet boundary—yet unlike WoS, WoSt can also take large steps near the Neumann boundary without incurring large bias. Our method is still entirely grid-free, *i.e.*, neither Ω nor $\partial\Omega$ has to be discretized; we need only query the boundary geometry $\partial\Omega$ via ray intersections and modified distance queries (Section 5). Here we assume for simplicity that the domain Ω is a compact subset of $\mathbb{R}^N$; see Appendix B for extensions to open domains and double-sided boundary conditions. Detailed pseudocode is provided in Algorithm 1.

4.1 Star-Shaped Subdomains

In lieu of balls, WoSt considers regions that are *star-shaped* with respect to a point x [Hansen et al. 2020]: any ray cast from x must intersect the region boundary only once. Though in principle any star-shaped subdomain could be used, we use regions $\mathrm{St}(x,r)$ given by the component of $B(x,r) \cap \Omega$ containing x, for a particular choice of ball B (see Section 4.4.2). Similar to Equation 1, we partition the region boundary into a Neumann part $\partial\mathrm{St}_N := \partial\Omega_N \cap \partial\mathrm{St}$ with prescribed normal derivatives $\partial u/\partial n = h$ from Equation 1, and a spherical part $\partial\mathrm{St}_B := \partial B \cap \partial\mathrm{St}$ (see inset).



4.2 Boundary Integral Formulation

Letting $A := \mathrm{St}$ and $C := B$ in Equation 8, the BIE becomes

$$\alpha(x)u(x) = \underbrace{\int_{\partial\mathrm{St}(x,r)} P^B(x,z)\, u(z) - \int_{\partial\mathrm{St}_N(x,r)} G^B(x,z)\, h(z)\,\mathrm{d}z}_{\text{boundary}} + \underbrace{\int_{\mathrm{St}(x,r)} G^B(x,y)\, f(y)\,\mathrm{d}y}_{\text{interior}}. \tag{17}$$

As with the BIE for WoS (Equation 11), the solution value $u(z)$ is the only unknown in this equation: at points $z \in \partial\mathrm{St}_N$ the normal derivative $\partial u/\partial n$ is given by the fixed Neumann data along $\partial\Omega$, and is not needed at points $z \in \partial\mathrm{St}_B$, where $G^B(x,z) = 0$. Since only one quantity is unknown, estimators for this equation need not branch.

Simonov [2008], and later Ermakov and Sipin [2009], take a parallel approach on domains Ω with *convex* Neumann boundaries $\partial\Omega_N$. In particular, they use regions formed by intersecting Ω with a ball B whose radius is the distance to the Dirichlet boundary, $d_{\mathrm{Dirichlet}} := \|x - \overline{x}(\partial\Omega_D)\|$. Hence, B can contain a subset of the Neumann boundary $\partial\Omega_N$. In the convex case, such regions are automatically star-shaped. To handle arbitrary domains, WoSt instead uses visibility information to obtain star-shaped regions even near nonconvex Neumann boundaries (which in general can yield a radius $r \leq d_{\mathrm{Dirichlet}}$), as we will see in Section 4.4.2.

4.3 The WoSt Estimator

Walk on Stars Estimator

A recursive single-sample estimator for Equation 17 is given by

$$\widehat{u}(x_k) := \underbrace{\frac{P^B(x_k,x_{k+1})\,\widehat{u}(x_{k+1})}{\alpha(x_k)\,p^{\partial\mathrm{St}(x_k,r)}(x_{k+1})} - \frac{G^B(x_k,z_{k+1})\,h(z_{k+1})}{\alpha(x_k)\,p^{\partial\mathrm{St}_N(x_k,r)}(z_{k+1})}}_{\text{boundary}} + \underbrace{\frac{G^B(x_k,y_{k+1})\,f(y_{k+1})}{\alpha(x_k)\,p^{\mathrm{St}(x_k,r)}(y_{k+1})}}_{\text{interior}}, \tag{18}$$

where

- the points $x_{k+1} \in \partial\mathrm{St}$, $z_{k+1} \in \partial\mathrm{St}_N$, and $y_{k+1} \in \mathrm{St}$ are sampled from the probability densities $p^{\partial\mathrm{St}}$ (Section 4.4), $p^{\partial\mathrm{St}_N}$ (Section 4.5), and p^{St} (Section 4.6), *resp.*
- r is chosen so that $\mathrm{St}(x_k,r)$ is star-shaped (Section 4.4).

At a high level, each step of WoSt accumulates contributions from the Neumann data h and source term f. For mixed Dirichlet-Neumann problems, the walk terminates in $\partial\Omega_D^\epsilon$, using the Dirichlet data g as the solution estimate, *i.e.*, $\widehat{u}(x_k) := g(\overline{x}_k(\partial\Omega_D))$. For pure Dirichlet problems, WoSt reduces to WoS; for pure Neumann problems we apply Tikhonov regularization (Section 3.4.2). We first discuss how to sample the next step x_{k+1} (Section 4.4), followed by sampling procedures for h and f (Sections 4.5, 4.6).

4.4 Random Walk on Star-Shaped Regions

The next walk location is importance sampled from the Poisson kernel for a ball centered at the current point x_k, *i.e.*, $x_{k+1} \sim p^{\partial\mathrm{St}} = P^B(x_k,x_{k+1})$. For a Poisson equation in $\mathbb{R}^3$, this kernel is given by

$$P^B_{3\mathrm{D}}(x_k,x_{k+1}) = \frac{n_{x_{k+1}} \cdot (x_{k+1} - x_k)}{4\pi\,\|x_{k+1} - x_k\|^3}. \tag{19}$$

We can use the same sampling density for a screened Poisson equation, since its corresponding kernel simply multiplies P^B by a constant in $[0,1)$ determined by the absorption coefficient (Equation 39).

Equation 19 coincides with the *signed solid angle* subtended by $\partial\mathrm{St}$ at x_{k+1} with respect to x_k [Barill et al. 2018; Feng et al. 2023]. In rendering, this term appears in the *light transport equation (LTE)* [Pharr et al. 2016, Equation 14.15]. Unlike the BIE, the LTE multiplies P^B with a binary *visibility* $V(x,y)$ that equals 1 if x and y are mutually visible. Visibility ensures the product $V(x_k,y)\,P^{B(x_k,r)}(x_k,y)$ is nonnegative: positive if y is visible from x_k; zero otherwise. Through a change of variables, this product can be importance sampled via *directional sampling*, *i.e.*, cast a ray from x_k in a direction $v \sim p^{\mathbb{S}^{N-1}}(v) = 1/|\mathbb{S}^{N-1}|$ uniformly sampled from the unit sphere, and find its first intersection with $\partial\mathrm{St}$:

$$x_{k+1} := x_k + t_\partial v,\ t_\partial := \min\{t \in [0,+\infty) : x_k + tv \in \partial\mathrm{St}(x_k,r)\}. \tag{20}$$

We refer to Veach and Guibas [1995b] for details on the relationship between area sampling and directional sampling.[1]

[1] If x_k lies on the boundary, then the ray origin should be offset slightly along the inward boundary normal to avoid self-intersections; see Wächter and Binder [2019].

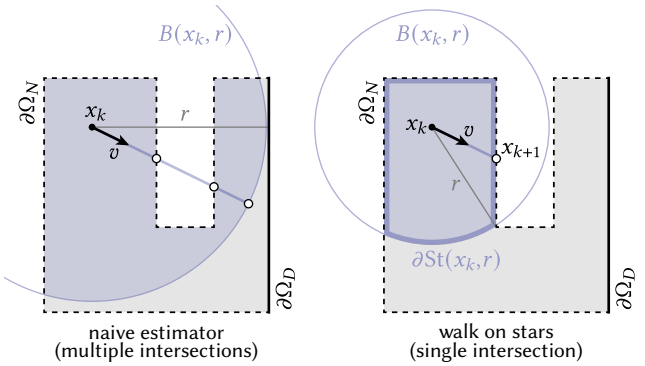


Fig. 7. *Left:* For a ball $B(x_k, r)$ whose radius r is the distance to the Dirichlet boundary $\partial\Omega_D$, the solution to a Poisson equation has to be estimated at all ray intersections, sampled proportional to signed solid angle. *Right:* WoSt instead restricts $B \cap \Omega$ to be *star-shaped* relative to x_k to avoid more than one intersection, estimating the PDE solution at the first intersection point x_{k+1} on either ∂B inside Ω, or the Neumann boundary $\partial\Omega_N$ inside B.

4.4.1 Non-Visible Regions. Since Brownian motion can effectively "walk around corners" (Figure 2), the BIE has no visibility term. Hence, the solution value at x_k can depend on non-visible points x_{k+1}, complicating use of directional sampling. In particular, if the subdomain around x_k is nonconvex, then a naïve strategy is to estimate u_{x+1} at *all* intersections along a ray from x_k (Figure 7, *left*), yielding a branching walk that increases in size exponentially. One could instead use just a single randomly selected intersection, but this approach yields extremely high variance (see Figure 14), for two reasons. First, the recursive solution estimate must be multiplied by the number of intersections to account for the expected contribution from each intersection, causing a blowup in value as walk length increases. Second, the Poisson kernel alternates sign along consecutive intersection points along a ray, yielding unstable estimates due to cancellation [Kalos and Whitlock 2009, Chapter 4].[2] These issues are also the root cause of high variance and bias in the walk on boundary method [Sabelfeld and Simonov 2013; Sugimoto et al. 2023]. Moreover, using just the first intersection leads to a biased estimator, as we explain in Appendix C.

4.4.2 Sampling Star-Shaped Regions. To avoid these issues, some past work assumes the *entire* Neumann boundary $\partial\Omega_N$ is convex [Simonov 2008; Ermakov and Sipin 2009], yielding only one intersection for any subdomain $B(x_k, r) \cap \Omega$ where $r := d_{\text{Dirichlet}}$. This assumption of course limits the applicability of such estimators.

We instead let r be the minimum of the distance $d_{\text{Dirichlet}}$ to the Dirichlet boundary, and the distance $d_{\text{silhouette}}$ to the closest point on the *visibility silhouette* of $\partial\Omega_N$ (Section 5). The connected component of $B(x_k, r) \cap \Omega$ containing x_k then defines a star-shaped region $\mathrm{St}(x_k, r)$. Figure 8 shows several examples. We can thus sample points on the region boundary $\partial\mathrm{St}$ by simply taking the first point along a ray from x_k that intersects either $\partial B(x_k, r)$ or $\partial\Omega$.

[2]One might overcome these issues by decomposing the BIE into independent integrals—each with a Poisson kernel that is entirely positive or negative—then randomly select one integral for sampling in proportion to the area over which it is integrated. Unfortunately, it is unclear how to efficiently perform such a decomposition.

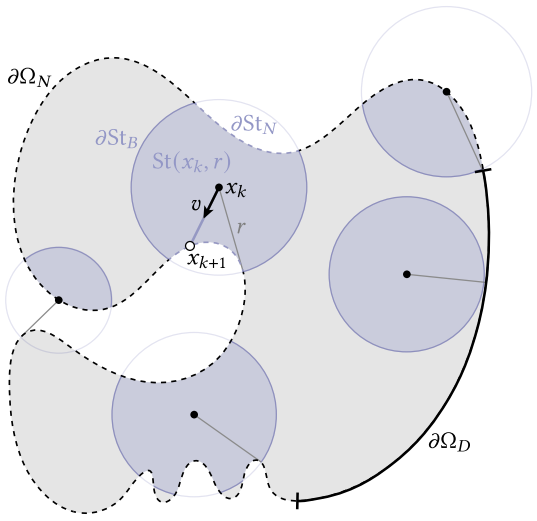


Fig. 8. We use star-shaped regions defined by intersecting a ball $B(x_k, r)$ with the domain Ω and taking the component connected to x_k. The ball does not contain the Dirichlet boundary $\partial\Omega_D$: only visible parts of the Neumann boundary $\partial\Omega_N$ and the spherical part of ∂B.

The use of star-shaped subdomains suggests the name *walk on stars*, in analogy with walk on spheres. Like the original WoS algorithm (and unlike the reflections in Figure 6) WoSt can take large steps when far from the Dirichlet boundary, using small steps mainly near termination. Though other star-shaped sets could be also used, our approach is motivated by the fact that the closest silhouette point is quite easy to compute—as will be discussed in Section 5.

4.4.3 Minimum Radius for Star-Shaped Regions. Near concave parts of the Neumann boundary, the distance to the closest silhouette point on $\partial\Omega_N$ shrinks to zero (Figure 9), stalling the progress of random walks. We hence limit the radius r used to define $\mathrm{St}(x_k, r)$ to a user-defined value $r^{\min}$, but still use only the first ray intersection to sample the next point x_{k+1}. This scheme incurs a small amount of bias when St is not star-shaped, since we effectively assume the solution u is zero on any piece of $\partial\mathrm{St}$ not visible from x_k (Figure 10, *left*)—Appendix C provides further discussion. As with the parameter ε for the Dirichlet ε-shell $\partial\Omega_D^\varepsilon$, a smaller $r^{\min}$ value reduces bias near concave regions of $\partial\Omega_N$ at the expense of performance. We study this performance-bias tradeoff in Section 6.1. In practice our star-shaped regions tend to be much larger than $r^{\min}$, even slightly away from a concave boundary.

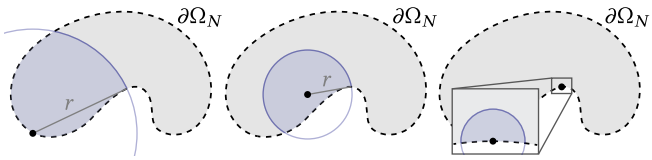


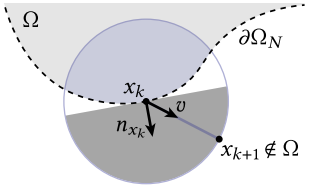
Fig. 9. The distance r to the visibility silhouette shrinks as a query point approaches a concave region on $\partial\Omega_N$.

Algorithm 1 WalkOnStars(x, n_x, onNeumann, ε, $r^{\min}$)

Input: A point x, the normal n_x at x (undefined if $x \notin \partial\Omega_N$), a flag onNeumann indicating whether $x \in \partial\Omega_N$, a termination parameter ε, and a minimum radius $r^{\min}$.
Output: A single-sample WoSt estimate $\widehat{u}(x)$ for Equation 1.

▷ Compute distance to $\partial\Omega_D$, or ∞ if $\partial\Omega_D = \emptyset$ ▷*Section 5*
$d_{\text{Dirichlet}}$, $\overline{x} \leftarrow$ DistanceDirichlet(x)
▷ Return boundary value g at $\overline{x}$ if $x \in \partial\Omega_D^{\varepsilon}$
if $d_{\text{Dirichlet}} < \varepsilon$ **then return** $g(\overline{x})$
▷ Compute distance to the visibility silhouette of $\partial\Omega_N$ (Alg. 2)
$d_{\text{silhouette}} \leftarrow$ SilhouetteDistanceNeumann(x, $d_{\text{Dirichlet}}$)
▷ Compute radius for star-shaped region $\text{St}(x)$
$r \leftarrow \max(r^{\min}, \min(d_{\text{Dirichlet}}, d_{\text{silhouette}}))$
▷ Uniformly sample a direction v on the unit sphere
$v \leftarrow$ SampleUnitSphere()
▷ If $x \in \partial\Omega_N$, sample v from hemisphere w/ axis $-n_x$ (Sec. 4.4.4)
if onNeumann **and** $n_x \cdot v > 0$ **then** $v \leftarrow -v$
▷ Intersect ray $x + rv$ w/ Neumann part ∂St_N, and get first hit
didIntersectNeumann, p, $n_p \leftarrow$ IntersectNeumann(x, v, r)
▷ If there is no hit, intersect spherical part ∂St_B instead
if not didIntersectNeumann **then** $p \leftarrow x + r\,v$
▷ Compute single-sample Neumann contribution
z, n_z, $\text{pdf}_z \leftarrow$ NeumannBoundarySample(x, r) ▷*Alg. 5*
isValid $\leftarrow \text{pdf}_z > 0$ **and** $\|z - x\| < r$ **and** ▷*Section 4.5*
  not IntersectNeumann(x, $z - x$, 1 − 1e-6)
$\alpha \leftarrow$ onNeumann ? 1/2 : 1
$\widehat{N} \leftarrow$ isValid ? $G^{B(x,r)}(x, z)\, h(z) \,/\, \alpha\, \text{pdf}_z : 0$
▷ Compute single-sample source contribution
$t_{\text{source}} \sim G^{B(x,r)}(x, y)/|G^{B(x,r)}(x)|$ ▷*Sec. 5.2*
$y \leftarrow x + t_{\text{source}}\, v$ ▷*Reuse v for source sample*
$\widehat{S} \leftarrow \|y-x\| < \|p-x\|$? $|G^B(x, r)|\, f(y) : 0$ ▷*$f = 0$ if $y \notin \text{St}(x)$*
▷ Update walk position and normal
$x, n_x \leftarrow p, n_p$ ▷*n_p is undefined if $x \notin \partial\Omega_N$*
onNeumann $\leftarrow$ didIntersectNeumann
return WalkOnStars(x, n_x, onNeumann, $\varepsilon, r^{\min}$) $- \widehat{N} + \widehat{S}$

4.4.4 Hemispherical Sampling on the Neumann Boundary. When x_k lies on $\partial\Omega_N$, sampling v from the unit sphere can yield points x_{k+1} outside Ω (see inset). Here we instead sample v from the hemisphere around n_{x_k}. This scheme effectively invokes the *reflection principle* of Brownian motion, across the halfplane at the base of the hemisphere [Jacobs 2010]. A useful consequence is that the $\alpha(x_k) = 1/2$ in the denominator of the first term in Equation 18 is canceled by the factor 1/2 we get from sampling a hemisphere rather than a sphere, preventing our recursive estimator from picking up a multiplicative factor of two each time a walk reaches $\partial\Omega_N$. Note that if $\partial\Omega_N$ is concave at x_k, we again incur a small amount of bias (Figure 10, *right*).



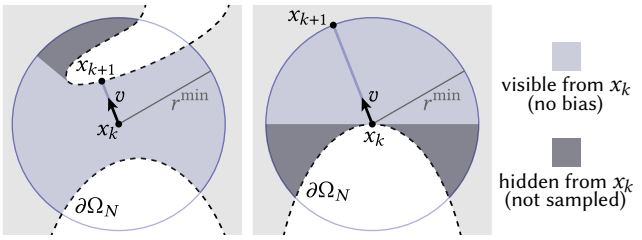


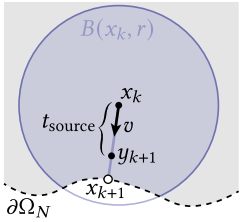
Fig. 10. WoSt uses balls with radius no smaller than $r^{\min}$ to prevent walks from stopping near concave Neumann boundaries. *Left:* We only sample parts of $\partial\Omega_N$ directly visible to x_k inside any ball $B(x_k, r^{\min})$, implicitly assuming the function u is zero elsewhere. *Right:* Hemispherical boundary sampling ensures the next walk location x_{k+1} does not leave the domain, but incurs a small bias in $\widehat{u}$ when x_k lies on a concave boundary.

4.5 Sampling Neumann Boundary Conditions

For problems with nonzero Neumann conditions, we must evaluate the second term in Equation 18. To do so, we sample a point z_{k+1} uniformly on the Neumann boundary $\partial\Omega_N$, adding a contribution $h(z_{k+1})$ only if z_{k+1} is also contained in ∂St_N. (Appendix D explains why this term is not estimated using the next location x_{k+1}.) This estimate remains unbiased, since we effectively integrate the same function (h restricted to ∂St_N) over a larger domain. Sampling the entire Neumann boundary leads to high variance in the estimator, as most samples will not lie on ∂St_N. Likewise, rejection sampling is prohibitively expensive since ∂St_N can be *much* smaller than $\partial\Omega_N$. In Section 5.2, we hence describe a strategy for efficiently generating samples z_{k+1} close to x_k, which significantly reduces variance.

4.6 Sampling the Source Term

Finally, we sample a point $y_{k+1} \in B(x_k, r)$ to estimate the interior integral in Equation 18 (Algorithm 1, *lines* 24-26). We reuse the ray direction v we sampled to generate x_{k+1} (Equation 20), and set $y_{k+1} := x_k + t_{\text{source}} v$, where we sample the distance $t_{\text{source}} \sim p(t_{\text{source}}) \propto G^B(x_k, x_k + t_{\text{source}} v)$. If the sampled distance t_{source} is greater than the distance $t_\partial := \|x_{k+1} - x_k\|$, then the point y_{k+1} is outside the star-shaped region $\text{St}(x_k, r)$, and we reject it (see inset). As in Section 4.4.4, (re)using a hemispherical direction cancels $\alpha(x_k) = 1/2$ when $x_k \in \partial\Omega_N$.



4.7 Final Estimator

Our final WoSt estimator is defined recursively as:

$$\widehat{u}(x_k) := \begin{cases} g(\overline{x}_k), & \overline{x}_k \in \partial\Omega_D^{\varepsilon}, \\ \widehat{u}(x_{k+1}) - \widehat{N} + \widehat{S} & \text{otherwise}, \end{cases} \tag{21}$$

where the next walk location x_{k+1} in Ω or on $\partial\Omega_N$ is sampled using the procedure in Section 4.4, and the non-recursive Neumann and source contributions $\widehat{N}$ and $\widehat{S}$ are provided in Algorithm 1, lines 22 and 26, *resp.* This estimator maintains the general structure of a WoS estimator, and thus introduces little implementation overhead.

5 GEOMETRIC QUERIES

In general, WoSt works with any boundary representation that supports the following queries:

- **Q.1** closest point queries to $\partial\Omega_D$,
- **Q.2** closest silhouette point queries to $\partial\Omega_N$,
- **Q.3** ray intersection queries against $\partial\Omega_N$, and
- **Q.4** point sampling queries on $\partial\Omega_N$.

Since none of these queries require the boundary to have a well-defined inside/outside, it need not be watertight, and can have cracks, holes, or self-intersections—see in particular Appendix B for a discussion of open domains and double-sided boundary conditions.

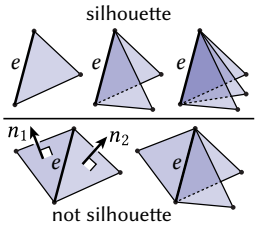
In principle, queries **Q.1**–**Q.4** could be evaluated for, say, spline patches or implicit surfaces (Section 7); we focus exclusively on triangle meshes. In particular, both **Q.1** and **Q.3** use standard closest point and ray intersection algorithms (not detailed here); Sections 5.1 and 5.2 describe closest silhouette point queries (**Q.2**) and point sampling queries (**Q.4**), *resp.* All queries use a *bounding volume hierarchy* (BVH) to achieve amortized sublinear scaling in the number of triangles; our basic approach is to add normal information to the BVH already used by WoS [Sawhney and Crane 2020, Section 5.1]. In particular, we build a standard BVH to perform closest point queries to the Dirichlet boundary (**Q.1**), and a separate SNCH (Section 5.1) for all queries of the Neumann boundary (**Q.2**-**Q.4**), using normal information only for **Q.2**. In practice, all queries needed to implement WoSt on triangle meshes are supported by the `FCPW` library of Sawhney [2021]; see also Appendix E for pseudocode.

5.1 Closest Silhouette Point Queries

The silhouette of a triangle mesh relative to a given direction v occurs along a set of edges e that satisfy a local silhouette condition. In particular, e is a silhouette edge if for each distinct pair of triangles containing e,

$$(v \cdot n_1) \cdot (v \cdot n_2) \leq 0, \tag{22}$$

where n_1, n_2 are consistently oriented normals (see inset). Note in particular that every boundary edge is a silhouette edge. In WoSt, v is the direction from the current walk location x to its closest point on e. A naïve strategy for finding the closest silhouette edge is to use a BVH to locate the closest point to x on all edges, skipping edges not contained in the silhouette. However, this strategy is highly inefficient when BVH nodes contain large, finely-tessellated regions that are all front- or back-facing (Figure 11): here each edge is examined (and rejected) exhaustively, whereas ideally the whole node should simply be culled.



5.1.1 Spatialized Normal Cone Hierarchy. To improve scaling, we hence augment our BVH with information about the orientation of the geometry inside each node. In particular, we adopt the *spatialized normal cone hierarchy* (SNCH) of Johnson and Cohen [2001]. Each node of a SNHC stores not only an axis-aligned bounding box (AABB), but also a *normal cone*. The cone axis is the average normal of all triangles in the node, and the cone half angle θ is the maximum angle between the axis and any triangle normal (Figure 12). Normal cones can be assembled during BVH construction. We currently use the *surface area heuristic (SAH)* [Wald 2007]; performance could be further improved via the *surface area orientation heuristic (SAOH)* of Estevez and Kulla [2018, Section 4.4], which clusters primitives according to both proximity and alignment.

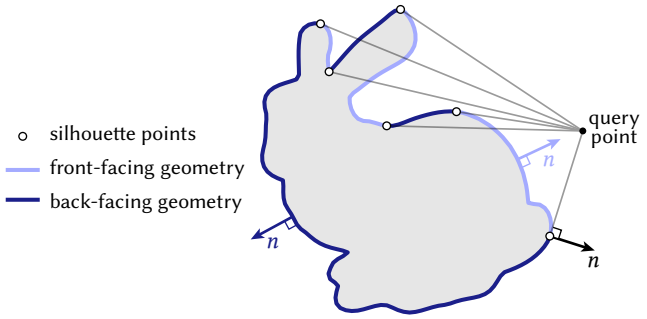


Fig. 11. An optimized procedure for finding silhouettes should avoid visiting finely-tessellated geometry that is entirely front- or back-facing relative to the query point.

5.1.2 Closest Silhouette Point Traversal. To perform a silhouette query, we traverse the SNCH in depth-first order (Algorithm 2). For each node N in this traversal we build a *view cone* rooted at x. The cone's axis points toward the center of N, and its half-angle tightly bounds the AABB (Figure 12); we then check if the view cone and the node's normal cone contain a pair of mutually orthogonal directions. If this test fails, all triangles in N must be front- or back-facing relative to the query point, and it can be skipped. In the context of WoSt, an upper bound on the size of a star-shaped region $\mathrm{St}(x)$ is given by the distance $d_{\mathrm{Dirichlet}}$ from x to the Dirichlet boundary (Section 4.4.2). To further improve query efficiency we can hence restrict the search to the radius $r^{\max} = d_{\mathrm{Dirichlet}}$.

5.2 Point Sampling Queries

Recall that for problems with nonzero Neumann conditions, we sample points from $\partial\Omega_N$ (Section 4.5). To increase the likelihood that these points lie on $\partial\mathrm{St}_N(x)$, we adopt a *hierarchical importance*

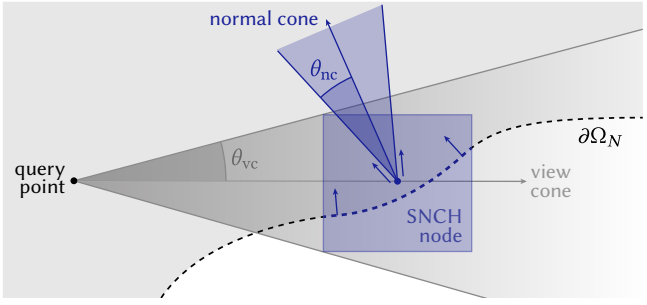


Fig. 12. A SNCH tests for a pair of mutually orthogonal directions in a view cone and a node's normal cone to determine whether the node contains a silhouette edge. The geometry inside the node can be skipped if no such pair of directions is found.

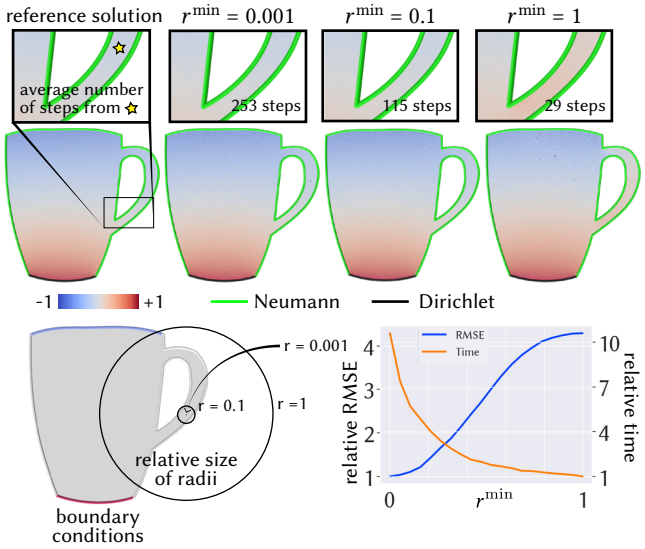


Fig. 13. Much like the parameter for the ε-shell, WoSt introduces an $r^{\min}$ parameter to help walks make progress near concave parts of the Neumann boundary. Walks generally converge faster with larger values for $r^{\min}$, with run-time improvements outweighing the relative increase in bias.

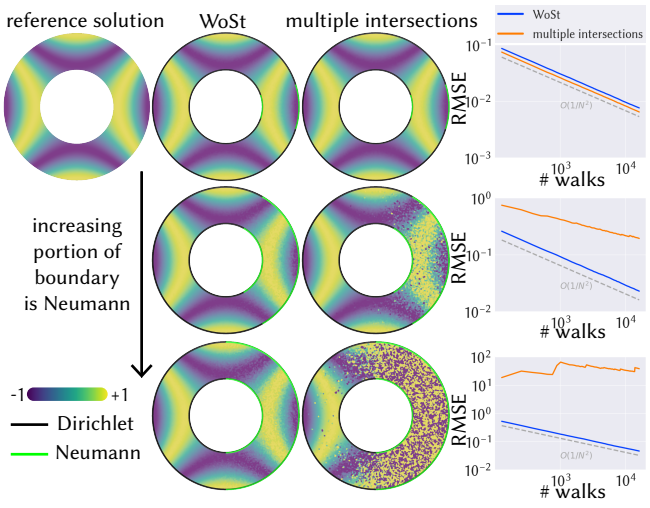


Fig. 14. Here we solve for a known reference function, using its normal derivatives to specify Neumann conditions on an increasingly large part of the boundary. WoSt exhibits the expected Monte Carlo convergence rate, whereas the estimator based on multiple ray intersections from Section 4.4.1 quickly blows up.

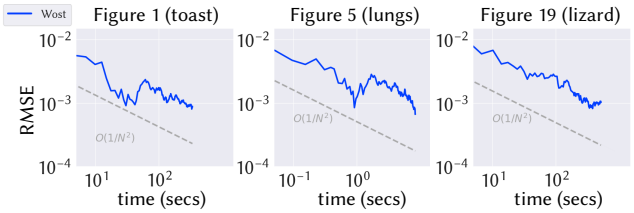


Fig. 15. WoSt exhibits the expected rate of convergence for a Monte Carlo estimator, shown here for eight fixed points on each example from Section 6.5. Reference solutions are also computed via WoSt with 2^{16} walks per point, as there is no analytical solution and no feasible alternatives to compute it. Timings were taken on an 8 core M1 MacBook Pro.

sampling strategy used in rendering to accelerate next-event estimation [Estevez and Kulla 2018]. In particular, during each step of BVH traversal we select only a single *random* child whose AABB intersects $\mathrm{St}(x)$. To give preference to nodes closer to the query point x, we sample according to the free-space Green's function $G^{\mathbb{R}^N}$ (Algorithm 5, *lines* 12-27), rather than the Green's function for a ball (which becomes negative outside $\mathrm{St}(x)$). Once we reach a leaf node, we uniformly sample a point from the leaf triangles with respect to surface area (see Algorithm 6, *lines* 2–9 and Algorithm 5, *line* 6–10). This point is not guaranteed to lie on $\partial\mathrm{St}_N$, but is much more likely to do so compared to uniformly sampling all of $\partial\Omega_N$. Estevez and Kulla [2018, Section 5.4] describe further improvements to this traversal strategy.

6 EVALUATION

In this section, we discuss practical considerations relating to our WoSt estimator such as stopping tolerances, and use several synthetic tests to evaluate its effectiveness. We use a multicore CPU-based implementation, and achieve essentially linear scaling; unless otherwise noted, all experiments used a 64-core 3rd Generation Intel Xeon workstation. As seen in, *e.g.*, Figure 5 *(left)* and discussed by Sawhney and Crane [2020, Section 7.5], the preprocessing cost of building a BVH is typically not significant (on the order of seconds), especially in contrast to finite element mesh generation (on the order of minutes to hours). Test problems are encoded much like a scene in a photorealistic renderer [Pharr et al. 2016], *e.g.*, using meshes or other boundary representations to describe $\partial\Omega$, and callback functions to encode the functions f, g, and h in Equation 1. See Sawhney and Crane [2020, Section 5] for further discussion.

6.1 Stopping Tolerances

Like WoS, WoSt terminates walks in an ε-shell $\partial\Omega_D^\varepsilon$ around the Dirichlet boundary, yielding a small, controllable amount of bias. The performance-bias tradeoff is also relatively insensitive to the parameter ε. See Binder and Braverman [2012] and Sawhney and Crane [2020, Section 6.1] for more detailed analysis and experiments.

The minimum radius parameter $r^{\min}$ from Section 4.4.3 also incurs bias near concave parts of $\partial\Omega_N$. Figure 13 examines the effect of this parameter—compared to ε, $r^{\min}$ typically exhibits a more sensitive performance-bias tradeoff, but with run-time improvements again outweighing the small increase in bias. In all other experiments we scale models to fit in a unit sphere, and use $\varepsilon = r^{\min} = 0.001$. Adaptively picking $r^{\min}$ based on local boundary curvature may yield even better performance/lower bias.

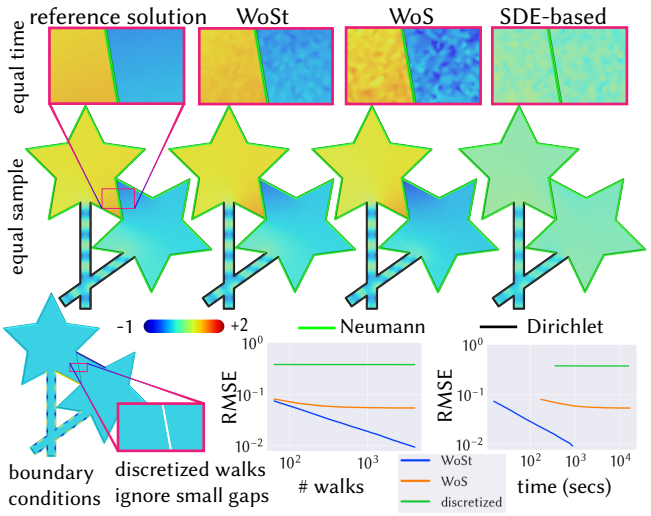


Fig. 16. For an equal number of walks, WoSt is significantly more efficient than both WoS with discretized boundary reflections (Section 3.4.2), as well as SDE-based estimators (Section 2). Timings were taken on an 8 core M1 MacBook Pro.

6.2 Convergence

As with most Monte Carlo estimators, WoSt exhibits error of magnitude about $O(1/\sqrt{N})$ with respect to the number of walks N (Figure 14), suggesting that any bias has little impact on overall accuracy. In general we observe that variance tends to be higher (but still predictable) in regions dominated by Neumann boundaries, due to longer walk lengths.

6.3 Comparisons

WoSt is both significantly faster and less biased than previous Monte Carlo approaches for solving mixed boundary-value problems with comparable parameter settings. For instance, Figure 16 compares several methods using comparable parameters: a minimum star radius $r^{\min} = 0.001$ for WoSt, reflection offset $\zeta = 0.01$ for WoS, and step size $l = 0.0001$ for the SDE-based method. As discussed inSection 3.4.2, WoS with boundary reflections suffers from bias buildup due to long walks that stick to the Neumann boundary. Methods based on reflecting SDEs perform even worse [Costantini et al. 1998] (see inset), as they incur bias not only on the boundary, but also in the interior. The naïve estimator from Section 4.4.1, which selects one intersection at random, also exhibits massive error as we increase the size of the Neumann boundary (Figure 14, *center right*). In contrast to all these methods, the star-shaped regions used by WoSt enable one to take large steps without incurring significant bias.

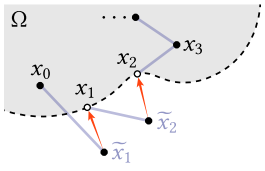


Fig. 17. SDE schemes project walks that leave the domain to the boundary, where the walk continues as usual.

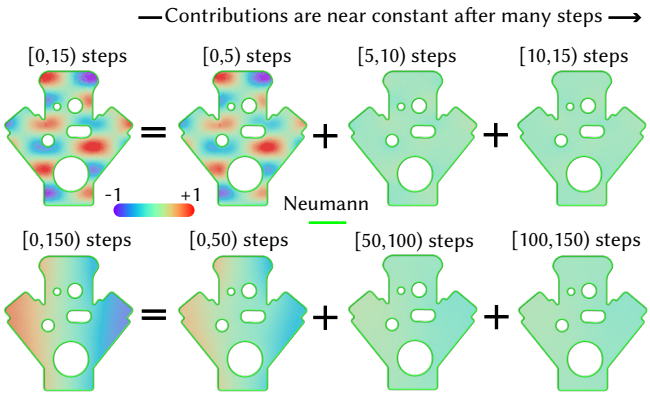


Fig. 18. The solution to a Poisson equation with pure Neumann conditions is uniquely defined up to an additive constant. *Top:* Local details in the PDE solution are often resolved by the first few steps of a WoSt random walk, with near-constant contributions from latter steps. *Bottom:* More steps are typically needed to resolve lower frequency global details.

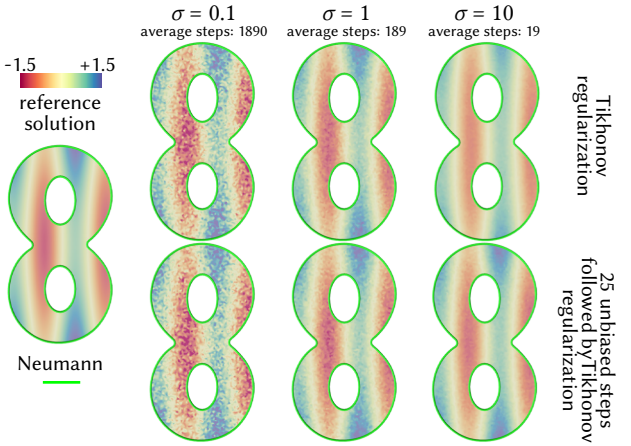


Fig. 19. *Top:* For pure Neumann problems, a small Tikhonov parameter σ yields long walks and high variance, while larger σ values produce shorter walks with less noise but more bias. *Bottom:* Since the solution is often well-resolved by short walks (Figure 18), we apply regularization only to walks longer than a given length—yielding both less noise and bias.

6.4 Pure Neumann Problems

The solution to a Poisson equation with pure Neumann conditions is determined only up to an additive constant. When we solve such a PDE with WoSt, we observe that high frequency details in the PDE solution are often resolved by the first few steps of a random walk, while the contribution from later steps is closer to constant (Figure 18). As discussed in Section 3.4.2, we use Tikhonov regularization to more effectively handle such problems—Figure 19 shows that this approach provides estimates with less noise and smaller bias even with substantial regularization, while ensuring that walk length is not unbounded. In general the number of steps needed to resolve the solution is problem-dependent—more steps are typically needed when the solution has low-frequency global features.

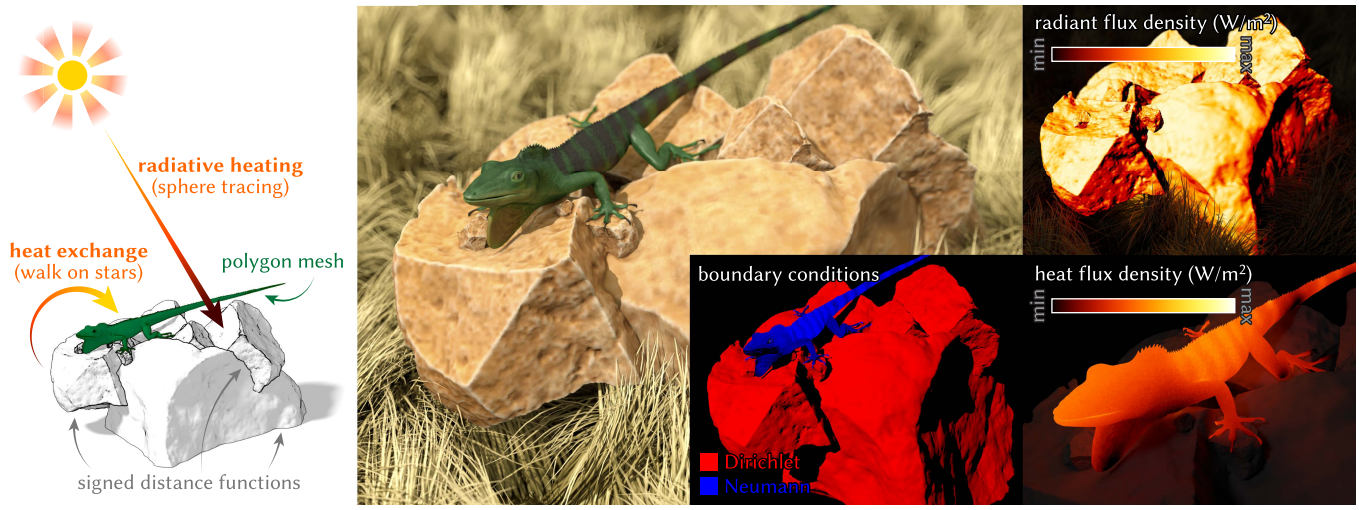


Fig. 20. Like Monte Carlo ray tracing, WoSt enables flexible modeling of scenarios that are geometrically and physically complex. *Center:* here an ectotherm *(anolis carolinensis)* warms itself on a rock, which in turn is heated by the sun. *Left:* Like WoS, WoSt can mix and match different geometric representations—here, a polygon mesh and a signed distance function. In this case, it is also easily combined with ray tracing used to determine Dirichlet boundary conditions for the heat transfer problem. *Right:* Unlike FEM and BEM, WoSt can handle highly-detailed boundary conditions without needing to resolve them on a mesh—here the lizard's texture controls the rate of heat absorption via Neumann boundary conditions, yielding stronger warming along the dark stripes.

6.5 Geometric Scaling and Flexibility

As noted in Section 1, a key motivation for developing grid-free Monte Carlo methods is to push simulation methods closer to the geometric complexity seen in photorealistic rendering—and in nature. To stress-test WoSt on complex geometry, we mock up three simulation examples here. Importantly, these examples do not aim to model exact physics or make quantitative predictions—we seek only to examine solver performance in the presence of (i) extremely complex geometry and (ii) large Neumann boundary regions, which are ubiquitous in real physical problems.

Heat transfer is a central topic in thermal engineering, with three basic modes: radiation, conduction, and convection. Thermal radiation is well-captured by 1st-order Monte Carlo light transport simulation, whereas conduction and convection involve diffusion, which must be simulated via a 2nd-order method like WoS or WoSt [Bati et al. 2023]. Thermal convection can also include turbulent advection *à la* Navier-Stokes, which can be solved via WoS [Rioux-Lavoie et al. 2022] but is not considered here.

Inspired by toast-darkening experiments of Myhrvold and Migoya [2017], Figure 1 models heat transfer from a toaster to a piece of bread, represented by a CT scan with 3.9 million boundary elements. To model diffusive convection, we solve a Laplace equation with large and small Dirichlet values on the heating elements and toaster cavity (*resp.*), and Neumann conditions on the bread. The solution is evaluated at roughly 2 million boundary points, using 1 walk per point for fast preview and 256 walks for the final solution; on average, WoSt takes 0.166 milliseconds per point for each walk (Figure 15 plots error versus time). A simple phenomenological model is used to translate surface temperature into color (though more principled models of *Malliard browning* could be used here [Chen et al. 2019]). We observe a marked difference between the temperature distribution resulting from radiation and convection—emphasizing the necessity of 2nd-order models for accurate thermal predictions.

Figure 20 shows another heat transfer experiment, where Dirichlet conditions induced by solar radiation are used to determine heat absorbed by an ectothermic lizard, modeled via detailed spatially-varying Neumann inflow conditions. Unlike FEM or BEM, where boundary data must be evaluated ahead of time, Dirichlet data is evaluated *on demand* via sphere tracing [Hart 1996]. Scene geometry is represented by a 1.2 million element boundary mesh (for the lizard) and implicit signed distance functions (for the rocks), highlighting the ability of WoSt to work with mixed boundary representations without global meshing. The solution is evaluated at 285k boundary points using 1024 walks per point, taking on average 0.121 milliseconds per point for each walk. As in Monte Carlo rendering (and unlike FEM/BEM), scene setup required no model conversion or meshing—even though data was pulled directly from the internet.

Finally, Figure 5 models oxygen diffusion in the lungs, one of many *Laplacian transport* phenomena with mixed boundary conditions [Grebenkov 2006]. We make a simplification by using Neumann rather than Robin boundary conditions (which are an important topic for future work—see Section 7). While FEM can solve such problems, *meshing* is both a major performance bottleneck and a hindrance for end-to-end robustness. In this case, even a state-of-the-art method [Hu et al. 2020] yields badly broken geometry; tweaking parameters to capture the correct geometry incurs a full day of compute time, eliminating any advantage of a fast solve. With WoSt we get feedback reliably and immediately in an output-sensitive fashion, here restricted to a cross section. In particular, we evaluate the solution on a 512×512 grid using 1024 walks per point; WoSt takes on average 0.021 milliseconds per point for each walk.

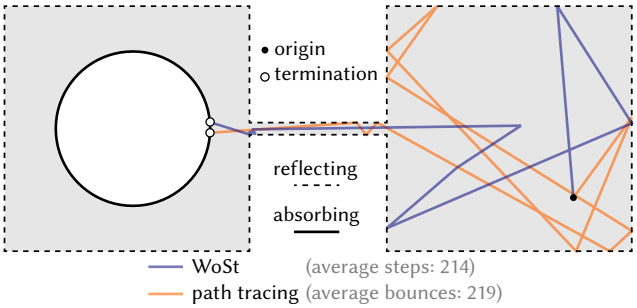


Fig. 21. Unidirectional random walk methods such as WoSt and path tracing require many steps or bounces to make their way through a key hole.

7 LIMITATIONS AND FUTURE WORK

The principal benefit of WoSt is not simply that it can solve Neumann problems, but rather that it exhibits a speed-bias tradeoff closer to the original WoS algorithm for Dirichlet problems, *i.e.*, it provides large steps and low-variance estimates away from the Dirichlet boundary—especially compared to SDE, WoS, or WoB approaches to the Neumann problem (Sections 2 and 6.3). However, Neumann and mixed boundary problems remain fundamentally more challenging than pure Dirichlet problems, with many opportunities for future extension and improvement.

Next-Event, Path-Space, and Bidirectional Estimators. For Neumann-dominated problems, forward random walk estimators must take many steps before obtaining a Dirichlet contribution, resulting in high computation time without a commensurate decrease in variance. This situation directly parallels forward rendering algorithms that produce long light paths in scenes with predominantly non-absorbing materials. For instance, Figure 21 replicates a classic "key-hole problem" using both 2D path tracing (assuming perfectly diffuse reflections) and WoSt (with Neumann-dominated boundary). In both cases, average walk length is greater than 200, highlighting a general challenge faced by unidirectional Monte Carlo methods.

In rendering, *next-event estimation* helps by adding a direct illumination contribution at each bounce [Whitted 1980; Cook et al. 1984]. One could likewise try adding a Dirichlet contribution at each step of WoSt, by allowing subdomains to contain part of the Dirichlet boundary. Such a scheme would simply need some way to estimate $\partial u/\partial n$ at Dirichlet points (Equation 17).

More generally, a *path-space formulation* [Pharr et al. 2016, Section 14.4.4] of the BIE may lead to estimators that make more global sampling decisions (*e.g.*, via Markov chain Monte Carlo [Veach and Guibas 1997; Kelemen et al. 2002]) or bidirectional estimators that help connect difficult-to-sample boundary and source data to arbitrary evaluation points [Lafortune and Willems 1993; Veach and Guibas 1995a]. The bidirectional WoS method of Qi et al. [2022] provides a concrete starting point—along with a rich literature from Monte Carlo rendering [Veach and Guibas 1997; Vorba and Křivánek 2016; Müller et al. 2017; Herholz et al. 2019; Müller et al. 2019, 2020].

Geometric Queries. The dominant cost in WoSt is evaluating geometric queries (Section 5). While ray- and closest-point queries are already well-optimized [Intel 2013; Wald et al. 2019; Krayer and Müller 2021], closest silhouette point queries could be further accelerated via, *e.g.*, better tree construction [Estevez and Kulla 2018, Section 4.4] or intelligent caching of silhouette edges. One might also extend silhouette queries to (neural) implicit surfaces by building on recent range analysis techniques [Sharp and Jacobson 2022]. Likewise, Neumann point sampling queries could be optimized *à la many-light sampling* [Estevez and Kulla 2018, Section 5.4], ensuring that Neumann boundary samples lie in star-shaped regions.

Concave Neumann Boundaries. To take larger steps near silhouette points, we could replace the fixed parameter $r^{\min}$ with an adaptive radius based on local curvature estimates [Pottmann et al. 2007]. Alternatively, one might apply *multi-level Monte Carlo* [Giles 2015; Misso et al. 2022] to reduce bias by aggregating estimates obtained via progressively smaller values of $r^{\min}$.

Global Information Sharing through Sample Reuse. WoSt estimates solution values independently at each point, often resulting in highly redundant computation. In concurrent work [Miller et al. 2023], we develop a grid-free sample reuse scheme for WoSt inspired by virtual point light (VPL) methods from rendering [Keller 1997; Dachsbacher et al. 2014]. This method gives unbiased solution estimates at arbitrary points by caching terms of the BIE on the domain boundary, and may open the door to domain decomposition strategies [Chan and Mathew 1994] for domains with thin features (Figure 21). Other reuse schemes from rendering such as photon mapping [Hachisuka et al. 2008; Hachisuka and Jensen 2009] and ReSTIR [Bitterli et al. 2020; Ouyang et al. 2021] surely provide similar opportunities.

Denoising and Geometric Prefiltering. Since elliptic PDEs have very regular solutions, high-frequency noise in WoSt estimates is nicely mitigated via denoising, as noted by Sawhney and Crane [2020, Figure 13]; here again methods from rendering provide a wealth of opportunities [Zwicker et al. 2015; Chaitanya et al. 2017; Schied et al. 2017, 2018; Gharbi et al. 2019; Kozlowski and Cheblokov 2021; NVIDIA 2022, 2017]. Another interesting challenge is how to detect—or even define—silhouettes for geometry with intricate microstructures [Neyret 1998], perhaps through some form of geometric prefiltering [Wu et al. 2019]. In particular, the SNCH itself provides a form of prefiltering: nodes higher than the leaves can provide conservative or approximate bounds on $d_{\text{silhouette}}$ that effectively amount to smoothing out fine-scale geometry (at the cost of bias). Likewise, SNCH nodes could be built by sampling a distribution (*à la* microflake models [Heitz et al. 2015]) rather than bounding explicit geometry.

Robin Boundary Conditions. In rendering, a room full of non-absorbing mirrors yields much longer path lengths than a scene with realistic, partially absorptive materials (Figure 22). Likewise, perfectly-reflecting Neumann conditions represent an "extreme case" for grid-free Monte Carlo methods, since materials in real physical problems are typically reflecting *and* absorbing. For instance, realistic thermal, electromagnetic, and fluid models often assume *Robin boundary conditions* of the form

$$\alpha u(x) + \beta \frac{\partial u(x)}{\partial n_x} = g(x) \quad \text{on } \partial\Omega, \qquad \alpha, \beta \in \mathbb{R} \tag{23}$$

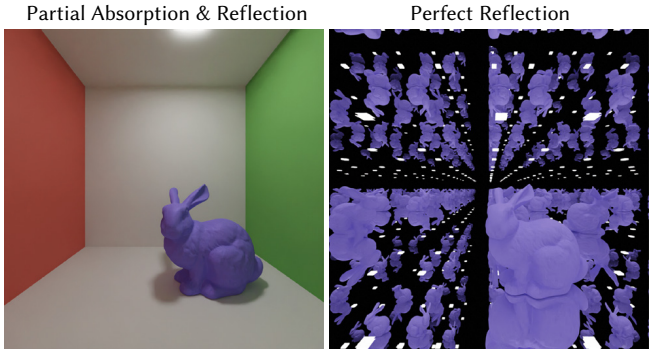


Fig. 22. Realistic scenes for visualization and analysis rarely have purely reflecting surfaces. *Left:* A rendered scene with both absorbing and reflecting surfaces. *Right:* A rendered scene of a room full of perfect mirrors.

[Senior 1960; Gustafson and Abe 1998; Grebenkov 2006; Hahn and Özisik 2012]. Incorporating such conditions into WoSt would result in shorter walks and greater efficiency, since (as in rendering) low-throughput walks could be terminated early via Russian roulette [Pharr et al. 2016, Section 13.7]. Support for Robin boundary conditions would enable WoSt to solve PDEs with variable coefficients [Sawhney et al. 2022] and exterior problems [Nabizadeh et al. 2021] with mixed boundary conditions, since the *Girsanov* and *Kelvin transformations* used by these methods (*resp.*) convert Neumann conditions into Robin conditions. Work by [Simonov 2017, Section 1] provides one possible starting point.

Extension to Other PDEs. Though we have developed WoSt in the context of (screened) Poisson equations, we believe the basic algorithmic strategy applies more broadly: fundamentally, the WoSt algorithm depends on the structure of the BIE, and BIE formulations are readily available for a variety of other PDEs, including the Helmholtz equation [Hunter and Pullan 2001, Chapter 3], linear elasticity [Hunter and Pullan 2001, Chapter 4] and the biharmonic equation [Ingham and Kelmanson 2012]. Stochastic integral formulations are also known for Navier-Stokes [Busnello et al. 2005; Rioux-Lavoie et al. 2022].

Even within the class of PDEs presented here, geometric scalability will likely pay dividends in well-chosen scientific and engineering contexts—just as it has for Monte Carlo simulation of light transport. In general, the very different capabilities of grid-free Monte Carlo methods are still largely unexplored in geometric, visual, and scientific computing, with many unique benefits and attractive use cases yet to be discovered.

ACKNOWLEDGMENTS

The authors thank Gautam Iyer for suggesting Tikhonov regularization, and Dario Seyb and Rasmus Tamstorf for fruitful conversations. This work was generously supported by nTopology and Disney Research, NSF awards 1943123, 2212290 and 2008123, Alfred P. Sloan Research Fellowship FG202013153, a Packard Fellowship, NSF Graduate Research Fellowship DGE2140739, and an NVIDIA Graduate Fellowship.

REFERENCES

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Trans. Graph.* 37, 4, Article 43 (2018), 12 pages.

Mégane Bati, Stéphane Blanco, Christophe Coustet, Vincent Eymet, Vincent Forest, Richard Fournier, Jacques Gautrais, Nicolas Mellado, Mathias Paulin, and Benjamin Piaud. 2023. Coupling Conduction, Convection and Radiative Transfer in a Single Path-Space: Application to Infrared Rendering. *ACM Trans. Graph.* 42, 4 (aug 2023).

Ilia Binder and Mark Braverman. 2012. The rate of convergence of the walk on spheres algorithm. *Geometric and Functional Analysis* 22, 3 (2012), 558–587.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Trans. Graph.* 39, 4, Article 148 (2020), 17 pages.

Andrei N Borodin and Paavo Salminen. 2015. *Handbook of Brownian motion-facts and formulae.* Springer Science & Business Media.

Barbara Busnello, Franco Flandoli, and Marco Romito. 2005. A probabilistic representation for the vorticity of a three-dimensional viscous fluid and for general systems of parabolic equations. *Proc. Edinburgh Math. Soc.* 48, 2 (2005), 295–336.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4 (2017).

Tony F Chan and Tarek P Mathew. 1994. Domain decomposition algorithms. *Acta numerica* 3 (1994), 61–143.

Peter Yichen Chen, Jonathan David Blutinger, Yorán Meijers, Changxi Zheng, Eitan Grinspun, and Hod Lipson. 2019. Visual modeling of laser-induced dough browning. *Journal of Food Engineering* 243 (2019), 9–21.

Chris J Coleman, David L Tullock, and Nhan Phan-Thien. 1991. An effective boundary element method for inhomogeneous PDEs. *J. App. Math. Phys. (ZAMP)* 42, 5 (1991).

Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. In *Proc. SIGGRAPH.* 137–145.

Martin Costabel. 1987. Principles of boundary element methods. *Computer Physics Reports* 6, 1-6 (1987), 243–274.

Cristina Costantini, Barbara Pacchiarotti, and Flavio Sartoretto. 1998. Numerical Approximation for Functionals of Reflecting Diffusion Processes. *SIAM J. Appl. Math.* 58, 1 (1998), 73–102.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable realistic rendering with many-light methods. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 88–104.

Cuiyang Ding, Changhao Yan, Xuan Zeng, and Wei Cai. 2022. A Parallel Iterative Probabilistic Method for Mixed Problems of Laplace Equations with the Feynman–Kac Formula of Killed Brownian Motions. *SIAM J. Sci. Comp.* 44, 5 (2022).

Dean G Duffy. 2015. *Green's functions with applications.* Chapman and Hall/CRC.

Sergey M Ermakov and Alexander S Sipin. 2009. The "walk in hemispheres" process and its applications to solving boundary value problems. *Vestnik St. Petersburg University: Mathematics* 42 (2009), 155–163. Issue 3.

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. 1, 2 (Aug. 2018), 25:1–25:17.

Lawrence C Evans. 1998. *Partial differential equations.* Vol. 19. Rhode Island, USA.

Nicole Feng, Mark Gillespie, and Keenan Crane. 2023. Winding Numbers on Discrete Surfaces. *ACM Trans. Graph.* (2023).

George Fishman. 2006. *Monte Carlo: concepts, algorithms, and applications.* Springer Science & Business Media.

Pau Gargallo, Emmanuel Prados, and Peter Sturm. 2007. Minimizing the reprojection error in surface reconstruction from images. In *IEEE Int. Conf. Comp. Vis.* 1–8.

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-Based Monte Carlo Denoising Using a Kernel-Splatting Network. *ACM Trans. Graph.* 38, 4, Article 125 (jul 2019), 12 pages. https://doi.org/10.1145/3306346.3322954

Michael B Giles. 2015. Multilevel monte carlo methods. *Acta numerica* 24 (2015), 259–328.

Denis S Grebenkov. 2006. Partially reflected Brownian motion: a stochastic approach to transport phenomena. *arXiv preprint math/0610080* (2006).

Denis S Grebenkov. 2007. NMR survey of reflected Brownian motion. *Reviews of Modern Physics* 79, 3 (2007), 1077.

Karl Gustafson and Takehisa Abe. 1998. The third boundary condition—was it Robin's? *The Mathematical Intelligencer* 20, 1 (1998), 63–71.

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–8. https://doi.org/10.1145/1618452.1618487

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive Photon Mapping. *ACM Trans. Graph.* 27, 5, Article 130 (2008), 8 pages.

Wolfgang Hackbusch. 2015. *Hierarchical matrices: algorithms and analysis.* Vol. 49. Springer.

David W Hahn and M Necati Özisik. 2012. *Heat conduction.* John Wiley & Sons.

Guillermo Hansen, Irmina Herburt, Horst Martini, and Maria Moszyńska. 2020. Star-shaped sets. *Aequationes mathematicae* 94, 6 (2020), 1001–1092.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.
Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX Microflake Distribution. *ACM Trans. Graph.* 34, 4, Article 48 (2015), 11 pages.
Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P A Lensch, and Jaroslav Křivánek. 2019. Volume Path Guiding Based on Zero-Variance Random Walk Theory. *ACM Trans. Graph.* 38, 3, Article 25 (2019), 19 pages.
Desmond J Higham. 2001. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review* 43, 3 (2001), 525–546.
Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4 (2020), 18 pages.
Peter Hunter and Andrew Pullan. 2001. Fem/bem notes. *Department of Engineering Science, The University of Auckland, New Zeland* (2001).
Derek B Ingham and Mark A Kelmanson. 2012. *Boundary integral equation analyses of singular, potential, and biharmonic problems*. Vol. 7. Springer Sci. Bus. Med.
Intel. 2013. Embree: High Performance Ray Tracing Kernels. http://embree.github.io/
Kurt Jacobs. 2010. *Stochastic processes for physicists: understanding noisy systems*. Cambridge University Press.
David E Johnson and Elaine Cohen. 2001. Spatialized normal cone hierarchies. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. 129–134.
Malvin H Kalos and Paula A Whitlock. 2009. *Monte carlo methods*. John Wiley & Sons.
Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 531–540.
Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 49–56.
Pawel Kozlowski and Tim Cheblokov. 2021. ReLAX: A Denoiser Tailored to Work with the ReSTIR Algorithm. *GPU Technology Conference* (2021).
Bastian Krayer and Stefan Müller. 2021. Hierarchical Point Distance Fields. In *International Symposium on Visual Computing*. Springer, 435–446.
Eric P Lafortune and Yves D Willems. 1993. Bi-directional path tracing. In *Proc. Int. Conf. Comp. Graph. Vis. Tech.* Alvor, Portugal.
Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph.* 37, 6 (2018).
Sylvain Maire and Etienne Tanré. 2013. Monte Carlo approximations of the Neumann problem. *Monte Carlo Methods and Applications* 19, 3 (2013), 201–236.
Michael Mascagni and Nikolai A Simonov. 2004. Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules. *SIAM J. Sci. Comp.* 26, 1 (2004).
Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4 (2023).
Zackary Misso, Benedikt Bitterli, Iliyan Georgiev, and Wojciech Jarosz. 2022. Unbiased and Consistent Rendering Using Biased Estimators. *ACM Trans. Graph.* 41, 4 (2022).
Linus Mossberg. 2021. *GPU-Accelerated Monte Carlo Geometry Processing for Gradient-Domain Methods*. Ph. D. Dissertation. Linköping University, Linköping, Sweden.
Mervin E Muller. 1956. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *Annals of Mathematical Statistics* 27, 3 (Sept. 1956), 569–589.
Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. In *Comp. Graph. Forum*, Vol. 36. Wiley Online Library.
Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM Trans. Graph.* 38, 5 (2019).
Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural Control Variates. *ACM Trans. Graph.* 39, 6 (2020).
Nathan Myhrvold and Francisco J Migoya. 2017. *Modernist bread*. Cooking Lab.
Mohammad Sina Nabizadeh, Ravi Ramamoorthi, and Albert Chern. 2021. Kelvin Transformations for Simulations on Infinite Domains. *ACM Trans. Graph.* 40, 4, Article 97 (jul 2021), 15 pages. https://doi.org/10.1145/3450626.3459809
Fabrice Neyret. 1998. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Trans. Vis. Comp. Graph.* 4, 1 (1998).
NVIDIA. 2017. NVIDIA OptiX AI-Accelerated Denoiser. https://developer.nvidia.com/optix-denoiser.
NVIDIA. 2022. NVIDIA Real-time denoisers (NRD). https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers.
Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 17–29.
Paul William Partridge, Carlos Alberto Brebbia, et al. 2012. *Dual reciprocity boundary element method*.
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.
Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu. 2007. Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design* 24, 8-9 (2007), 428–442.
Yang Qi, Dario Seyb, Benedikt Bitterli, and Wojciech Jarosz. 2022. A bidirectional formulation for Walk on Spheres. *Computer Graphics Forum* 41, 4 (2022), 51–62. https://doi.org/10.1111/cgf.14586 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14586
Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H. Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2022. A Monte Carlo Method for Fluid Simulation. *ACM Trans. Graph.* 41, 6, Article 240 (nov 2022), 16 pages. https://doi.org/10.1145/3550454.3555450
Karl K Sabelfeld and Nikolai A Simonov. 2013. *Random walks on boundary for solving PDEs*. De Gruyter.
Rohan Sawhney. 2021. FCPW: Fastest Closest Points In The West. https://github.com/rohan-sawhney/fcpw
Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Trans. Graph.* 39, 4, Article 123 (aug 2020), 18 pages. https://doi.org/10.1145/3386569.3392374
Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Trans. Graph.* 41, 4, Article 53 (jul 2022), 17 pages. https://doi.org/10.1145/3528223.3530134
Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*. 1–12.
Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 1–16.
Thomas BA Senior. 1960. Impedance boundary conditions for imperfectly conducting surfaces. *Applied Scientific Research, Section B* 8, 1 (1960), 418–436.
Nicholas Sharp and Alec Jacobson. 2022. Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis. *ACM Trans. Graph.* 41, 4, Article 107 (jul 2022), 16 pages. https://doi.org/10.1145/3528223.3530155
Nikolai A Simonov. 2008. *Walk-on-Spheres Algorithm for Solving Boundary-Value Problems with Continuity Flux Conditions*. Springer Berlin Heidelberg, Berlin, Heidelberg. 633–643 pages.
Nikolai A Simonov. 2017. Walk-on-spheres algorithm for solving third boundary value problem. *Applied Mathematics Letters* 64 (2017), 156–161.
Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Trans. Graph.* 42, 4 (aug 2023), 16 pages. https://doi.org/10.1145/3592109
Eric Veach and Leonidas Guibas. 1995a. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.
Eric Veach and Leonidas J Guibas. 1995b. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 419–428.
Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 65–76. https://doi.org/10.1145/258734.258775
Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Trans. Graph.* 35, 4, Article 42 (jul 2016), 11 pages. https://doi.org/10.1145/2897824.2925912
Carsten Wächter and Nikolaus Binder. 2019. A fast and robust method for avoiding self-intersection. *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs* (2019), 77–85.
Ingo Wald. 2007. On fast construction of SAH-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing*. IEEE, 33–40.
Ingo Wald, Will Usher, Nathan Morrical, Laura Lediaev, and Valerio Pascucci. 2019. RTX Beyond Ray Tracing: Exploring the Use of Hardware Ray Tracing Cores for Tet-Mesh Point Location.. In *High Performance Graphics (Short Papers)*. 7–13.
Turner Whitted. 1980. An Improved Illumination Model for Shaded Display. *Commun. ACM* 23, 6 (jun 1980), 343–349. https://doi.org/10.1145/358876.358882
Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. 2019. Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces. *ACM Trans. Graph.* 38, 4, Article 137 (jul 2019), 14 pages. https://doi.org/10.1145/3306346.3322936
Ekrem Fatih Yılmazer, Delio Vicini, and Wenzel Jakob. 2022. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. *arXiv preprint arXiv:2208.02114* (2022).
Yijing Zhou, Wei Cai, and Elton Hsu. 2017. Computation of the local time of reflecting brownian motion and the probabilistic representation of the neumann problem. *Communications in Mathematical Sciences* 15, 1 (2017), 237–259.
Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. In *Computer graphics forum*, Vol. 34. Wiley Online Library, 667–681.

A GREEN'S FUNCTIONS AND POISSON KERNELS

Here we provide expressions for Green's functions and Poisson kernels in free-space and over a ball in 2D and 3D. Our WoSt estimator uses these functions to solve the Poisson and screened Poisson equations via their boundary integral formulation.

A.1 Poisson Equation

The free-space Green's functions in 2D and 3D for two points x and y equal:

$$G^{\mathbb{R}^2}(x,y) = \frac{\log(r)}{2\pi}, \quad G^{\mathbb{R}^3}(x,y) = \frac{1}{4\pi r}, \tag{24}$$

where $r := \|y - x\|$. The corresponding Poisson kernels equal:

$$P^{\mathbb{R}^2}(x,y) = \frac{n_y \cdot (y-x)}{2\pi r^2}, \quad P^{\mathbb{R}^3}(x,y) = \frac{n_y \cdot (y-x)}{4\pi r^3}, \tag{25}$$

where n_y is the unit normal at y.

For a ball $B(x,R)$ of radius R centered at x, we can derive 2D and 3D Green's functions from the corresponding free-space expressions using the *method of images* [Duffy 2015]. This gives:

$$G^B_{2\mathrm{D}}(x,y) = \frac{\log(R/r)}{2\pi}, \quad G^B_{3\mathrm{D}}(x,y) = \frac{1}{4\pi}\left(\frac{1}{r} - \frac{1}{R}\right). \tag{26}$$

These functions integrate over the ball $B(x,R)$ to:

$$|G^B_{2\mathrm{D}}(x)| := \int_{B(x,R)} G^B_{2\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{R^2}{4}, \tag{27}$$

$$|G^B_{3\mathrm{D}}(x)| := \int_{B(x,R)} G^B_{3\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{R^2}{6}. \tag{28}$$

The corresponding Poisson kernels are the same as their free-space counterparts. For any point $y \in \partial B$ where $n_y = (y-x)/R$, they simplify to:

$$P^B_{2\mathrm{D}}(x,y) = \frac{1}{2\pi R}, \quad P^B_{3\mathrm{D}}(x,y) = \frac{1}{4\pi R^2}. \tag{29}$$

A.2 Screened Poisson Equation

Let I_n and K_n (for $n = 0, 1, \ldots$) denote modified Bessel functions of the first and second kind, *resp.* The free-space Green's functions in 2D and 3D for a screened Poisson equation with a positive screening coefficient σ then equal:

$$G^{\sigma,\mathbb{R}^2}(x,y) = \frac{K_0(r\sqrt{\sigma})}{2\pi}, \quad G^{\sigma,\mathbb{R}^3}(x,y) = \frac{e^{-r\sqrt{\sigma}}}{4\pi r}. \tag{30}$$

The corresponding Poisson kernels are:

$$P^{\sigma,\mathbb{R}^2}(x,y) = Q^{\sigma,\mathbb{R}^2}(x,y)\; P^{\mathbb{R}^2}(x,y), \tag{31}$$

$$P^{\sigma,\mathbb{R}^3}(x,y) = Q^{\sigma,\mathbb{R}^3}(x,y)\; P^{\mathbb{R}^3}(x,y), \tag{32}$$

where

$$Q^{\sigma,\mathbb{R}^2}(x,y) := K_1(r\sqrt{\sigma})\; r\sqrt{\sigma}, \tag{33}$$

$$Q^{\sigma,\mathbb{R}^3}(x,y) := e^{-r\sqrt{\sigma}}\left(r\sqrt{\sigma} + 1\right). \tag{34}$$

For a ball $B(x,R)$, the 2D and 3D Green's functions are:

$$G^{\sigma,B}_{2\mathrm{D}}(x,y) = \frac{1}{2\pi}\left(K_0(r\sqrt{\sigma}) - I_0(r\sqrt{\sigma})\frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})}\right), \tag{35}$$

$$G^{\sigma,B}_{3\mathrm{D}}(x,y) = \frac{1}{4\pi}\left(\frac{\sinh((R-r)\sqrt{\sigma})}{r\sinh(R\sqrt{\sigma})}\right). \tag{36}$$

Their integral over the ball $B(x,R)$ equals:

$$|G^{\sigma,B}_{2\mathrm{D}}(x)| := \int_{B(x,R)} G^{\sigma,B}_{2\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{1}{\sigma}\left(1 - \frac{1}{I_0(R\sqrt{\sigma})}\right), \tag{37}$$

$$|G^{\sigma,B}_{3\mathrm{D}}(x)| := \int_{B(x,R)} G^{\sigma,B}_{3\mathrm{D}}(x,y)\,\mathrm{d}y = \frac{1}{\sigma}\left(1 - \frac{R\sqrt{\sigma}}{\sinh(R\sqrt{\sigma})}\right). \tag{38}$$

Finally, the corresponding Poisson kernels at any point $y \in B$ equal

$$P^{\sigma,B}_{2\mathrm{D}}(x,y) = Q^{\sigma,B}_{2\mathrm{D}}(x,y)\; P^{\mathbb{R}^2}(x,y), \tag{39}$$

$$P^{\sigma,B}_{3\mathrm{D}}(x,y) = Q^{\sigma,B}_{3\mathrm{D}}(x,y)\; P^{\mathbb{R}^3}(x,y), \tag{40}$$

where

$$Q^{\sigma,B}_{2\mathrm{D}}(x,y) := \left(K_1(r\sqrt{\sigma}) + I_1(r\sqrt{\sigma})\frac{K_0(R\sqrt{\sigma})}{I_0(R\sqrt{\sigma})}\right) r\sqrt{\sigma}, \tag{41}$$

$$Q^{\sigma,B}_{3\mathrm{D}}(x,y) := e^{-r\sqrt{\sigma}}\left(r\sqrt{\sigma}+1\right) + \left(\cosh(r\sqrt{\sigma})\; r\sqrt{\sigma} - \sinh(r\sqrt{\sigma})\right)\frac{e^{-R\sqrt{\sigma}}}{\sinh(R\sqrt{\sigma})}. \tag{42}$$

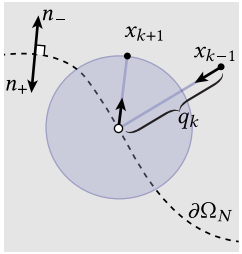
We note that $Q^{\sigma,B} \in [0,1)$ for $\sigma > 0$. Using Equation 39, we introduce a multiplicative weight of $Q^{\sigma,B}$ in the solution estimate at every step of a WoSt random walk for a screened Poisson equation, since directions are sampled proportionally to $P^{\mathbb{R}^N}$ (Section 4.4). As we mention in Section 3.4.2, the cumulative product of $Q^{\sigma,B}$ can be used as a Russian roulette probability to terminate walks.

B OPEN DOMAINS AND DOUBLE-SIDED BOUNDARIES

Here we describe how to modify Algorithm 1 to support double-sided boundary conditions in an open domain $\Omega \subset \mathbb{R}^N$. We start with the BIE for double-sided boundary conditions [Costabel 1987]:

$$\alpha(x)u(x) = \overbrace{\int_{\partial\Omega} P^+(x,z)\left[u^+(z) - u^-(z)\right] - G(x,z)\left[\frac{\partial u^+(z)}{\partial n_z^+} - \frac{\partial u^-(z)}{\partial n_z^-}\right]\mathrm{d}z}^{\text{boundary}} + \overbrace{\int_\Omega G(x,y)\; f(y)\,\mathrm{d}y}^{\text{domain}}, \tag{43}$$

where n^+ and n^- denote unit outward and inward facing normals on $\partial\Omega$, *resp.*, u^+ and u^- represent corresponding solution values on either side of $\partial\Omega$, and the kernel $P^+(x,z) := \partial G(x,z)/\partial n_z^+$. Since all points are either on the boundary or the domain interior, $\alpha = 1/2$ on $\partial\Omega$ and 1 otherwise.



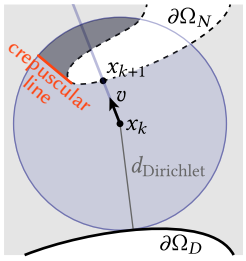
The high-level idea is to estimate Equation 43 by choosing an appropriate set of boundary conditions to use in a star-shaped region $\mathrm{St}(x_k, r)$ centered at the current walk location x_k, *i.e.*, g^+ or g^- as the Dirichlet data when $x_k \in \partial\Omega_D$, and h^+ or h^- as the Neumann data in St when $x_k \in \partial\Omega_N$. The choice of boundary conditions depends on whether the boundary $\partial\Omega$ is front- or back-facing relative to x_k or x_{k-1} for $k > 0$ (inset). We assume the boundary $\partial\Omega$ has a canonical orientation defined by the unit outward normal n_z^+ for any point $z \in \partial\Omega$, and that the walk's direction of approach towards $\partial\Omega$ equals $q_k := x_k - x_{k-1}$.

For $k = 0$, we require q_0 as input to the algorithm to determine the user-specified choice of boundary conditions to use; for instance, setting $q_0 = n^+_{x_0}$ uses g^+ if $x_0 \in \partial\Omega_D$.

For double-sided Dirichlet boundary conditions, we change line 4 in Algorithm 1 to return g^+ when $q_k \cdot n^+_{x_k} > 0$ and g^- otherwise; the dot product determines whether the boundary is back-facing relative to x_{k-1}. Before using hemispherical direction sampling on $\partial\Omega_N$ to determine the next walk location x_{k+1} (*line* 12), we flip the direction of the boundary normal $n^+_{x_k}$ if $q_k \cdot n^+_{x_k} < 0$; this ensures that x_{k+1} lies on the same side of the boundary as the direction from which the walk approached $\partial\Omega_N$. Finally, we let z_{k+1} be a Neumann sample on $\partial\mathrm{St}_N$ for double-sided Neumann conditions. In line 22, we then use h^+ on z_{k+1} if $(z_{k+1} - x_k) \cdot n^+_{z_{k+1}} > 0$ and h^- otherwise when $x_k \notin \partial\Omega_N$; the boundary orientation in this case is determined relative to x_k. When $x_k \in \partial\Omega_N$, we use h^+ as the Neumann data on z_{k+1} if the boundary normal is flipped and h^- otherwise.

There is a non-zero probability for a random walk to wander off to infinity when using WoSt in an open domain or in the exterior of a closed domain; this corresponds to the *non-recurrent* behavior of Brownian motion in 2D and 3D [Borodin and Salminen 2015, Chapter 2]. Nabizadeh et al. [2021] use WoS to solve exterior problems with pure Dirichlet boundary conditions outside a closed domain by performing a spherical inversion of the domain. We leave extending their approach to mixed boundary-value problems to future work.

C WHY NOT ALWAYS USE THE FIRST INTERSECTION



At first glance, the discussion in Section 4.4.1 prompts a simple idea for generalizing the technique of Simonov [2008] and Ermakov and Sipin [2009] to nonconvex domains (which unfortunately does not work): consider the subdomain $B(x_k, d_{\text{Dirichlet}}) \cap \Omega$, and sample the next point x_{k+1} by taking the *first* intersection with $\partial\left(B(x_k, d_{\text{Dirichlet}}) \cap \Omega\right)$ along a ray v from x_k, ignoring any later intersections. This is analogous to choosing a BIE subdomain $A \subset B(x_k, d_{\text{Dirichlet}}) \cap \Omega$ comprising only points in $B(x_k, d_{\text{Dirichlet}}) \cap \Omega$ visible to x_k. The boundary of such a subdomain A is made up of two components: first, the part of $\partial\left(B(x_k, d_{\text{Dirichlet}}) \cap \Omega\right)$ that is visible to x_k; second, any crepuscular rays [Gargallo et al. 2007] corresponding to the visibility silhouette on $\partial\left(B(x_k, d_{\text{Dirichlet}}) \cap \Omega\right)$ with respect to x_k (see inset).

The second component is the reason why just sampling the first intersection would produce biased results, as one would never sample any points on the crepuscular rays. Even if we considered a modified sampling strategy that allowed for generating points z on these rays, we would additionally need to estimate the normal derivative $\partial u(z)/\partial n_z$ of the solution at z as this data is only known on the Neumann boundary. (An estimate of the solution $u(z)$ is not needed, as the Poisson kernel is zero on crepuscular rays.) By constrast, we shrink the subdomain A to $B(x_k, \min(d_{\text{Dirichlet}}, d_{\text{silhouette}})) \cap \Omega$ in Section 4.4.2 so that it does not contain any crepuscular rays—using just the first intersection then evades the aforementioned problems.

D WHY SAMPLE THE NEUMANN TERM SEPARATELY

We use two separate samples x_{k+1} and z_{k+1} to estimate the first and second terms in Equation 18 corresponding to the unknown solution value u and known Neumann data h, *resp.* However, we could in theory use x_{k+1} to estimate both boundary terms. To do so, we would rewrite the WoSt estimator as follows:

$$\widehat{u}(x_k) := \underbrace{\frac{P^B(x_k, x_{k+1})\,\widehat{u}(x_{k+1})}{\alpha(x_k)\,p^{\partial\mathrm{St}(x_k,r)}(x_{k+1})} - \frac{G^B(x_k, x_{k+1})\,\tilde{h}(x_{k+1})}{\alpha(x_k)\,p^{\partial\mathrm{St}(x_k,r)}(x_{k+1})}}_{\text{boundary}} + \underbrace{\frac{G^B(x_k, y_{k+1})\,f(y_{k+1})}{\alpha(x_k)\,p^{\mathrm{St}(x_k,r)}(y_{k+1})}}_{\text{interior}}, \tag{44}$$

where

$$\tilde{h}(x_{k+1}) := \begin{cases} h(x_{k+1}), & x_{k+1} \in \partial\mathrm{St}_N(x_k, r), \\ 0, & x_{k+1} \in \partial\mathrm{St}_B(x_k, r). \end{cases} \tag{45}$$

In practice, this approach is problematic as the second term in Equation 44 is biased: the direction sampling procedure in Section 4.4 never samples a point x_{k+1} on a flat Neumann boundary when $x_k \in \partial\Omega_N$, even if h is non-zero there. This is because the Poisson kernel P^B in Equation 19 (which serves as our sampling density $p^{\partial\mathrm{St}}$) is zero for points x_{k+1} where $n_{x_{k+1}} \perp (x_{k+1} - x_k)$. More generally, even for non-flat boundaries, the ratio $G^B/p^{\partial\mathrm{St}} := G^B/P^B$ in the second term in Equation 44 results in high-variance estimates as the Poisson kernel can take on both very large and small values. These issues motivate our use of a separate sample z_{k+1} from the probability density function $p^{\partial\mathrm{St}_N(x_k,r)}(z_{k+1})$ (Equation 18), which we generate using the procedure described in Section 4.5.

E PSEUDOCODE

Here we provide pseudocode for the geometric queries in Section 5.

Algorithm 2 SilhouetteDistanceNeumann(x, $r^{\max} = \infty$)

Input: A query point $x \in \mathbb{R}^3$ & a radius r around x to search in.
Output: The closest point to x on the visibility silhouette of $\partial\Omega_N$.

1: **return** DistClosestSilhouette(snch.root, x, $r^{\max}$)

Algorithm 3 DistClosestSilhouette($T, x, r, d_T^{\min} = 0$)

Input: A spatialized normal cone hierarchy T, a query point $x \in \mathbb{R}^3$, a radius r around x to search in, and optionally the minimum distance to T's AABB from x (0 if x is inside AABB).
Output: Distance from x to closest point on visibility silhouette.

1: **if** $d_T^{\min} > r$ **then return** r ▷*Ignore nodes outside search radius*
2: **if** T.isLeaf **then**
3: ▷Return distance to closest silhouette edge in leaf node
4: **for** e **in** T.edges **do**
5: $p_e \leftarrow$ ClosestPointOnEdge(e, x)
6: $v \leftarrow p_e - x$
7: $d_e \leftarrow |v|$
8: **if** $d_e < r$ **then**
9: $\mathtt{hasTri}_0$, $n_0 \leftarrow$ GetAdjacentTriangleNormal(e, 0)
10: $\mathtt{hasTri}_1$, $n_1 \leftarrow$ GetAdjacentTriangleNormal(e, 1)

```
            isSilhouetteEdge ← not hasTri_0 || not hasTri_1 ||
                               (v · n_0) · (v · n_1) ≤ 0
            if isSilhouetteEdge then return d_e
    else
        ▷Intersect AABBs with sphere formed by x and r
        L, R ← T.left, T.right
        visit_L, d_L^min ← IntersectAABBSphere(L.aabb, x, r)
        visit_R, d_R^min ← IntersectAABBSphere(R.aabb, x, r)
        ▷Cull nodes with only front- or back-facing triangles
        if visit_L and d_L^min > 0 then
            visit_L ← HasSilhouette(L.aabb, L.cone, x)   ▷Alg. 4
        if visit_R and d_R^min > 0 then
            visit_R ← HasSilhouette(R.aabb, R.cone, x)   ▷Alg. 4
        if visit_L and visit_R then
            if d_L^min < d_R^min then                    ▷Traverse closer subtree first
                r ← DistClosestSilhouette(L, x, r, d_L^min)
                return DistClosestSilhouette(R, x, r, d_R^min)
            else
                r ← DistClosestSilhouette(R, x, r, d_R^min)
                return DistClosestSilhouette(L, x, r, d_L^min)
        else if visit_L then
            return DistClosestSilhouette(L, x, r, d_L^min)
        else if visit_R then
            return DistClosestSilhouette(R, x, r, d_R^min)
    return r
```

Algorithm 4 HasSilhouette(aabb, $\text{cone}_{\text{normal}}$, x)

Input: An AABB aabb, a $\text{cone}_{\text{normal}}$ encoding normal information for the triangles in aabb, and a query point x.

Output: *Conservative* guess of whether aabb contains a silhouette with respect to x, achieved by checking if the normal and view cones associated with the AABB contain orthogonal directions.

```
▷aabb may contain silhouette if cone_normal's half angle is ≥ 90°
a_nc ← cone_normal.axis
θ_nc ← cone_normal.halfAngle
if θ_nc ≥ π/2 then return true
▷Set view cone axis from x to aabb's center
a_vc ← Centroid(aabb) − x
l_vc ← |a_vc|
a_vc ← a_vc / l_vc                                   ▷View cone axis
▷aabb may contain silhouette if a_vc is ⊥ to directions in cone_normal
φ ← ArcCos(a_nc · a_vc)                              ▷Angle between normal & view axes
if π/2 ≥ φ − θ_nc and π/2 ≤ φ + θ_nc then return true
▷Compute view cone half angle w.r.t. aabb's bounding sphere
r ← BoundingSphereRadius(aabb)
if l_vc ≤ r then return true                         ▷invalid cone, sphere contains x
θ_vc ← ArcSin(r / l_vc)                              ▷View cone half angle
▷aabb may contain silhouette if cones contain ⊥ directions
θ_sum ← θ_nc + θ_vc
if θ_sum ≥ π/2 then return true
return π/2 ≥ φ − θ_sum and π/2 ≤ φ + θ_sum
```

Algorithm 5 NeumannBoundarySample(x, r)

Input: A sphere with center $x \in \mathbb{R}^3$ and radius r.

Output: A point $z \in \partial\Omega_N$ if the sphere is nonempty, the unit outward normal n_z and the probability pdf_z. The sample is not guaranteed to lie inside the sphere, but is likely to be near x.

```
▷Sample a random triangle inside or intersecting with the sphere
t ← null
pdf_t ← 0
SampleTriangleInSphere(snch.root, x, r, t, pdf_t)
if t not null then
    ▷Sample a random point z on the triangle t
    z, n_z, pdf_z ← SamplePointOnTriangle(t)
    return z, n_z, pdf_t · pdf_z
return null, null, 0
```

Algorithm 6 SampleTriangleInSphere($T, x, r, t, \text{pdf}_t, \text{pdf}_T = 1$)

Input: A binary tree T, a sphere with center $x \in \mathbb{R}^3$ and radius r, a triangle t yet to be selected and its sampling pdf_t. The optional argument computes pdf_T of traversing a random branch in T.

Output: A randomly selected triangle t in the sphere & its sampling pdf_t; no triangle is selected if the sphere does not intersect $\partial\Omega_N$.

```
if T.isLeaf then
    ▷Select a random triangle proportionally to its area
    totalArea ← 0
    for t_T in T.triangles do
        if IntersectTriangleSphere(t_T, x, r) then
            totalArea ← totalArea + Area(t_T)
            if Rand() · totalArea < Area(t_T) then
                t ← t_T
                pdf_t ← pdf_T · Area(t)
    if totalArea > 0 then pdf_t ← pdf_t / totalArea
else
    ▷Select subtree to traverse weighted by its proximity to x
    L ← T.left
    R ← T.right
    weight_L ← IntersectAABBSphere(L.aabb, x, r) ?
                   G^ℝ³(x, Centroid(L.aabb)) : 0
    weight_R ← IntersectAABBSphere(R.aabb, x, r) ?
                   G^ℝ³(x, Centroid(R.aabb)) : 0
    totalWeight ← weight_L + weight_R
    if totalWeight > 0 then
        prob_L ← weight_L / totalWeight
        if Rand() < prob_L then
            pdf_L ← pdf_T · prob_L
            SampleTriangleInSphere(L, x, r, t, pdf_t, pdf_L)
        else
            pdf_R ← pdf_T · (1 − prob_L)
            SampleTriangleInSphere(R, x, r, t, pdf_t, pdf_R)
```