Eurographics Symposium on Rendering (2023)
T. Ritschel and A. Weidlich (Guest Editors)

Mean Value Caching for Walk on Spheres

Ghada Bakbouk Pieter Peers

College of William & Mary

Abstract

Walk on Spheres (WoS) is a grid-free Monte Carlo method for numerically estimating solutions for elliptical partial differential equations (PDE) such as the Laplace and Poisson PDEs. While WoS is efficient for computing a solution value at a single evaluation point, it becomes less efficient when the solution is required over a whole domain or a region of interest. WoS computes a solution for each evaluation point separately, possibly recomputing similar sub-walks multiple times over multiple evaluation points. In this paper, we introduce a novel filtering and caching strategy that leverages the volume *mean value property (in contrast to the* boundary *mean value property that forms the core of WoS). In addition, to improve quality under sparse cache regimes, we describe a weighted mean as well as a non-uniform sampling method. Finally, we show that we can reduce the variance within the cache by recursively applying the volume mean value property on the cached elements.*

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis;***

1. Introduction

Partial differential equations (PDEs) form the basis of many fundamental computer graphics problems. The recently introduced Monte Carlo Geometry Processing (MC-GP) framework [SC20] offers an exciting new strategy for solving PDEs defined over volumes without the need to discretize or create a grid. At its core, MC-GP builds on the Walk on Spheres (WoS) algorithm for solving PDEs. Due to its conceptual similarity to path tracing, many Monte Carlo innovations and solution strategies from rendering have been applied to MC-GP, such as importance sampling [SC20] and reverse and bidirectional algorithms [QSBJ22].

Monte Carlo techniques are very effective for computing a solution estimate at a single evaluation point. However, in many practical cases, the solution over the whole volume or region of interest in the volume is desired. Because the solution estimate is computed for each evaluation point separately and independently, many similar sub-walks are recomputed multiple times. To reduce recomputation, prior work looked at interpolation with Moving Least Squares [Nea04] of forward walks, and reusing reverse walks [QSBJ22]. While the former is biased, the latter still requires a large number of reverse walks to ensure a sufficiently dense overlap with each evaluation point.

In this paper we present a novel method for reusing *forward* walks that is easy to implement in existing WoS frameworks. At the core of our method is the *volume mean value property*, hence we call our method (volume) mean value caching. The volume mean value property is the volumetric counterpart of the (boundary) mean value property that enables walking on spheres. We show that simply performing the first step in a WoS-walk over the sphere's volume instead of its boundary leads to a filtering method that greatly reduces Monte Carlo noise over multiple parallel WoS estimates covering a volume. We also show that the volume mean value property leads to an efficient caching scheme that is unbiased for uniformly distributed cache samples. In addition, to equalize variance between evaluation points, we describe a (consistent) non-uniform sampling strategy. Furthermore, we introduce a weighted volume mean value property to improve the smoothness of the solution when using a low number of cache samples. Finally, we show that by recursively applying the volume mean value property, we can also reduce the variance in the cache itself.

We validate our mean value caching strategy on a variety of PDEs and show that we can reduce the required sample count by several orders of magnitude for equivalent error levels. In summary, our contributions are:

- A hybrid volume and boundary mean value formulation of WoS;
- An unbiased post-processing filtering method for reducing Monte Carlo noise over uniformly distributed WoS estimates;
- A mean value caching and gathering method for reusing WoS-walks that is unbiased for uniformly distributed cache samples, and consistent for a non-uniformly sampled cache;
- A weighted volume mean value formulation; and
- A recursive algorithm for reducing the variance *within* the cache.

2. Related Work

Walk on spheres [Mul56] was introduced to computer graphics in the context of grid-free Monte Carlo geometry process-

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

ing [SC20] for solving PDEs with Dirichlet boundary conditions, and it has been extended to solve PDEs with spatially varying coefficients [SSJC22] and Neumann boundary conditions [SMGC23]. WoS has been successfully applied to fluid simulation [RLSO*22] and inverse PDE problems [YVJ22]. In this paper, we introduce a caching and filtering strategy for reusing walks in grid-free Monte Carlo methods for solving PDEs. While we focus on standard WoS walks [SC20], our method only requires a Monte Carlo estimate and thus it is compatible with recent advances in this area [SMGC23].

In addition to introducing a Monte Carlo geometry processing framework, Sawhney et al. [SC20] explore multiple variance reduction strategies common in rendering such as importance sampling and control variates. Sawhney et al. also report that Monte Carlo rendering denoising methods [RMZ13, KBS15] are effective for Monte Carlo geometry processing. However, these denoising methods do not leverage the properties of the underlying PDE, and thus offer no guarantees on the accuracy of the denoised solution. A key difference between the rendering equation and PDEs is that once irradiance is recorded at the camera sensor, no further transport to neighboring pixels is possible. In contrast, the PDE solution represents an equilibrium, and solutions to neighboring points are related through the mean value property. In this paper, we will leverage this relation to provide unbiased filtering. Sawhney et al. also propose to reduce computation costs by adaptively sampling the domain and approximating solutions between samples using MLS interpolation [Nea04]. However, as noted by Sawhney et al., the resulting solution is biased. In contrast, we leverage the mean value property to obtain an unbiased solution from uniformly distributed cache samples, and a consistent solution for a non-uniformly sampled cache.

Qi et al. [QSBJ22] derive a bidirectional formulation for WoS that introduces reverse walks starting at the boundary. In addition, Qi et al. also introduce a two-pass reverse WoS algorithm, where the first pass consists of computing and storing reverse walks. During evaluation, all points that overlap with the largest ball are summed and weighted by the corresponding Green's function. Our algorithm also stores walks, but in contrast to Qi et al., we store forward paths. This gives us better control over the distribution of cache samples, and thus the number of samples per ball during the gathering stage. Furthermore, since our method only requires an unbiased estimate of the solution at each cache point, we can easily incorporate other estimation algorithms [SMGC23] without the need to derive a 'reverse' variant. Finally, our method supports variance reduction within the cache by leveraging the volume mean value property and the known distribution of sample points over the domain; this distribution is unknown when using reverse walks.

In concurrent work, Miller et al. [MSCG23] describe a boundary value cache method that stores solution values and their gradients at the boundary of the target domain or region of interest. Instead of storing the samples, Miller et al. follow a progressive approach of splatting the cache entries directly to evaluation points. Our method differs in that we store samples within the volume and that we do not require estimates of the gradients. We explicitly store the cache samples in a KD-tree alleviating the need to know the evaluation points a-priori. Furthermore, Miller et al. rely on the free-space Green's function which exhibits a singularity at cache points. Instead, we leverage the volume mean value property of harmonic functions which does not suffer from such singularities, although the cache could be undersampled close to the boundary. Finally, our method supports leveraging the samples in the cache itself to reduce the variance on each of the cache samples, which further translates to a reduction in variance when evaluating the PDE solution for points not in the cache.

Caching/sample reuse has been extensively explored in Monte Carlo rendering by using: virtual point lights [Kel97, DKH*14], (ir)radiance caching [WRC88, JDZJ08], photon maps [HJ09, HOJ08, Jen96], reservoir sampling [BWP*20, OLK*21], filtering [ZJL*15], etc. Similar to two-pass methods such as VPLs and photon mapping, we also store 'paths' for reuse in the second pass. Our method differs in that we store forward walks, and add connections at the front; in contrast, reverse paths are stored in rendering and then connected to a forward path. Our method is most similar to irradiance caching in that we also store *"forward paths"*. However, in contrast to irradiance caching, our "interpolation of samples" is based on the volume mean value property which does not introduce additional bias. Finally, our method can also be considered a special case of ReSTIR where the reweighting/resampling is not needed because nearby samples are valid thanks to the volume mean value property.

3. Background

In this paper, we will focus on solving Laplace and Poisson partial differential equations, for which the solution u should satisfy:

$$\begin{aligned} \Delta u &= f \quad \text{on } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where Δ is the Laplace operator defined as the sum of the unmixed second partial derivatives, Ω is the domain with boundary $\partial\Omega$, f is the source term, and g is the boundary condition; when $f = 0$, Equation (1) is a Laplace partial differential equation. Leveraging the mean value property, the solution value to Equation (1) at any point $x \in \Omega$ can be expressed as:

$$u(x) = \frac{1}{|\partial B_r(x)|} \int_{\partial B_r(x)} u(y)dy + \int_{B_r(x)} f(y)G_r(x,y)dy, \tag{2}$$

where $B_r(x)$ is a ball with radius r around x fully contained in Ω, and $G_r(x,y)$ is the Green's harmonics function on the ball $B_r(x)$.

Walk on Spheres (WoS) offers a convenient Monte Carlo-based algorithm for solving Equation (2), by sampling $x_{k+1} \sim p_{\partial B_{r_k}(x_k)}(x_{k+1})$ for the boundary term (1st term), and $y_k \sim p_{B_{r_k}(x_k)}(y_k)$ inside the ball $B_{r_k}(x_k)$ to estimate the source term (2nd term):

$$\langle u(x_k)\rangle = \frac{\langle u(x_{k+1})\rangle}{|\partial B_{r_k}(x_k)| p_{\partial B_{r_k}(x_k)}(x_{k+1})} + \frac{f(y_k)G_r(x_k,y_k)}{p_{B_{r_k}(x_k)}(y_k)}, \tag{3}$$

where x_0 is set to the evaluation point x, and $\langle u(x_k)\rangle = g(x_k)$ if $x_k \in \partial\Omega$. While the ball $B_{r_k}(x_k)$ in Equation (3) can be of any radius r_k as long as $B_{r_k}(x_k) \subseteq \Omega$ and centered around x_k, in practice we take the largest possible ball $B_{r_k}(x_k)$ such that it touches $\partial\Omega$ in order to minimize the number of steps needed to terminate the walk

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

at the boundary. Unless the boundary $\partial\Omega$ has a (partially) spherical boundary, the intersection $B_{r_k}(x_k) \cap \partial\Omega$ has zero area measure. Thus x_k is never sampled on the boundary $\partial\Omega$, and consequently the walk never terminates. Therefore, an ε-shell is defined around the domain boundary $\partial\Omega$, and if x_k falls within this shell, g is directly read out for the nearest point on the boundary and the walk is terminated. This approximation introduces a small bias [SC20]. While in theory, First Passage Surface methods can avoid the ε-shell bias, in practice these methods are limited to restricted polygonal boundary configurations [GHD97, MH03].

4. Volume Mean Value Property

The mean value theorem is critical in transforming the partial differential equation in Equation (1) into an integral form (Equation (2)) that can be numerically solved with Monte Carlo integration. However, there also exists a volume version of the mean value property[†]:

$$u(x) = \frac{1}{|B_r(x)|} \int_{B_r(x)} u(y)dy, \tag{4}$$

that takes the mean over the volume of the ball $B_r(x)$ instead of the boundary $\partial B_r(x)$. Note, for brevity we omit the source term contribution; we will discuss adding the source term in Section 8. The volume mean value property follows directly from the boundary mean value property:

$$\begin{aligned} \int_{B_r(x)} udV &= \int_0^r \left(\int_{\partial B_\rho(x)} udS \right) d\rho, && (5)\\ &= u(x) \int_0^r \left(\int_{\partial B_\rho(x)} dS \right) d\rho, && (6)\\ &= u(x) \int_{B_r(x)} dV, && (7) \end{aligned}$$

where V, S, and ρ are the volume, surface and radial measures, respectively. The first step in the above derivation follows from the co-area formula, and the second step from the boundary mean value property.

While we could, similarly to Equation (3), apply Monte Carlo integration on Equation (4), it would likely yield slower convergence in practice than regular (boundary) WoS. However, we can also combine both. One possible combination, that we will leverage in this paper, is to select $x_1 = z \sim p_{B_r(x)}(z)$ and $x_{k+1} \sim p_{\partial B_{r_k}(x_k)}(x_{k+1})$ for $k > 0$, such that:

$$\langle u(x_k) \rangle = \begin{cases} \frac{\langle u(z) \rangle}{|B_r(x)| p_{B_r(x)}(z)} & \text{if } k = 0, \\ \frac{\langle u(x_{k+1}) \rangle}{|\partial B_{r_k}(x_k)| p_{\partial B_{r_k}(x_k)}(x_{k+1})} & \text{otherwise.} \end{cases} \tag{8}$$

In other words, we only apply the volume mean value property to the first step in the recursion, and follow regular WoS for subsequent steps. Note, for clarity we use z to denote volume-sampled points and x_k for boundary-sampled points.

We can leverage this Monte Carlo formulation in two applications: filtering and caching.

[†] We will refer to the mean value property in Equation (2) as the *boundary mean value property*.

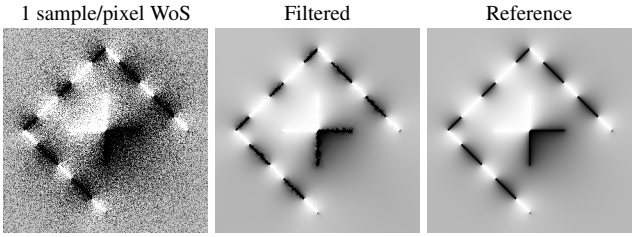

Figure 1: *Solutions to a Laplace PDE with* OPEN DIAMOND & CROSS *boundary conditions for* 500×500 *evaluation points computed with (left)* 1 *WoS-walk per evaluation point, (middle) filtered results of the* 1 *WoS-walk result, and a reference solution (right) computed with* 20,000 *WoS-walks per evaluation point.*

Filtering Suppose that an estimate of $u(x)$ is available for a dense set of N evaluation points $z_i, i \in [1...N]$ (with known distribution) computed with Equation (3). Each solution value $\langle u(z_i) \rangle$ will have some unbiased error proportional to the number of Monte Carlo samples. To reduce the Monte Carlo noise, one could increase the number of samples. However, an alternative strategy would be to leverage Equation (8), and define a secondary volume estimator for $u(x)$ for any $x \in \{z_i\}$:

$$\langle u(x) \rangle = \frac{1}{K} \sum_{z_i \in B_r(x)} \frac{\langle u(z_i) \rangle}{|B_r(x)| p_{B_r(x)}(z_i)}, \tag{9}$$

where K the number of evaluation points z_i in the ball $B_r(x)$. Note, that the computation of the estimators $\langle u(z_i) \rangle$ follows the second term in Equation (8) (i.e., regular WoS), and Equation (9) is related to the first term in Equation (8). Therefore, the average over all points z_i (appropriately weighted) within the ball $B_r(x)$ yields an estimate of the solution at x with lower variance than its original corresponding z_i sample. For example, assume the solution to a PDE is desired in the 2D image domain Ω_{2D}, and a single sample WoS estimate is computed for each pixel (i.e., z_i is uniformly distributed over Ω_{2D}: $p_{\Omega_{2D}}(z_i) \sim \frac{1}{|\Omega_{2D}|}$). The resulting image will be extremely noisy (e.g., Figure 1 (left) shows a 1 sample WoS solution of a Laplace PDE with OPEN DIAMOND & CROSS boundary conditions at 500×500 resolution). We can then *filter* the result by averaging all pixel estimates in the largest disc around each pixel, yielding an unbiased smoother result (middle) closer to the reference (right). Equation (9) provides an unbiased estimate with a significantly lower variance ($\sim 1/\sqrt{|B_r(x)|}$ when z_i is uniformly sampled), without computing any new samples. However, the variance reduction depends on the radius of the ball, hence evaluation points close to a boundary benefit less from filtering than evaluation points far away from the boundary.

Mean Value Caching We note that Equation (8) does not require that the evaluation point x coincides with an a-priori sampled z_i. Hence, Equation (8) can also be used to estimate solutions *in between* sampled points z_i. Furthermore, there is no restriction on using the same set of samples for computing the volume integral between two neighboring evaluation points. Leveraging both observations yields a straightforward and efficient 2-pass mean value caching strategy. In a first pass, we compute solution estimates

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

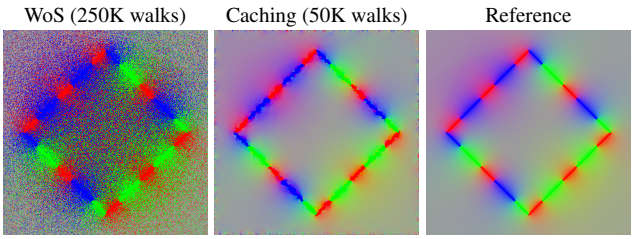

Figure 2: *Mean value caching (middle) demonstrated on a Laplace PDE with* COLORED DIAMOND *boundary conditions at* 500×500 *evaluation points, computed with on average one WoS-walk per* 5 *evaluation points, and compared to a regular WoS solution (left) computed with five times as many walks (*1 *per evaluation point).*

using WoS (or any other unbiased estimate via alternate methods such as Walk on Stars [SMGC23] or First Passage Surface methods [GHD97,MH03]) at N cache points $z_i, i \in [1...N]$ uniformly distributed over Ω and store the solution estimates in a cache for subsequent reuse in a second pass. In practice, similar to path tracing, we only take a single sample at each walk-step when recursively computing Equation (3) (i.e., the estimate from a single WoS-walk). During evaluation, we *gather* the solution at any point $x \in \Omega$ similar to Equation (9) using the largest possible ball $B_r(x) \subseteq \Omega$ around x. Because z_i is uniformly sampled in Ω, it follows that:

$$p_{B_r(x)}(z_i) = \frac{1}{|B_r(x)|}. \tag{10}$$

In practice, we store the WoS-walk estimates and corresponding positions z_i as keys in a KD-tree to quickly find all z_i within the ball $B_r(x)$. If the number of WoS-walk samples within a ball $B_r(x)$ falls below a user-set threshold (1 in our implementation), we switch to a regular WoS estimate; this often occurs close to the boundaries where the expected length of the WoS-walks is short, and thus an estimate can be obtained with few steps. In this paper when the ball is empty, we only use a single regular WoS-walk to better demonstrate the effects of mean value caching; in practice, more samples are needed to obtain high quality estimates.

Figure 2 compares a solution estimate of a Laplace PDE with the DIAMOND boundary conditions computed using a mean value cache constructed from 50,000 WoS-walks uniformly distributed over Ω against a regular WoS solution computed with five times as many walks (i.e., 250,000) and a converged WoS solution estimate computed with 20,000 WoS-walks per evaluation point (5,000M walks in total). Even though only $1/5$th of the WoS-walks are computed (compared to 1 WoS-walk per evaluation point), the mean value cache produces results with significantly less variance. Compared to the reference solution (computed with 100,000 times more walks), differences are mostly noticeable close to the boundary due to the reduced gathering radius.

Mean value caching is unbiased if the solution value estimates at the sample point z_i are unbiased. Sharing samples between the estimates of neighboring points does not introduce bias, but rather it correlates the unbiased errors between neighboring points making the result visually easier to interpret.

Region of Interest In order for Equation (9) to provide an unbiased estimate of $u(x)$ during gathering, it is important to ensure that each point in the ball $B_r(x)$ can potentially be covered by a cache sample z_i. In other words, the pdf $p_{B_r(x)}(z_i) > 0$. To ensure this condition is met, care needs to be taken when the region of interest or the domain Ω is *not* fully enclosed by the boundary $\partial\Omega$. In such a case, we must ensure that the ball $B_r(x)$ does not only stay within the boundary $\partial\Omega$, but that the ball also remains fully inside the region in which we sampled z_i. As a consequence, the gathering radius can become very small for points close to the boundary of the region of interest, and thus the variance of the solution estimate will increase for these points. To reduce variance at the region boundary, we recommend to sample z_i in a slightly larger area than the region of interest. However, to better demonstrate the quality of solutions obtained with mean value caching, all results shown in this paper are computed without oversampling, and we use a matching sampling and gathering region.

5. Weighted Volume Mean Value Property

When moving the ball $B_r(x)$ to a neighboring point x', some samples will enter the new ball $B_{r'}(x')$ and some will leave the ball $B_{r'}(x')$. At low sample rates, the number of sample points z_i within each ball $B_r(x)$ can be low, and hence the impact of samples entering and leaving the ball can induce an abrupt change in the solution. Due to the correlated sampling, such discontinuities can be visually noticeable. While the estimate is not incorrect, it goes against the expectation that $u(x)$ varies smoothly.

To address this issue, we introduce the weighted volume mean value property:

$$\int_{B_r(x)} w(\rho) u dV = \int_0^r w(\rho) \left(\int_{\partial B_\rho(x)} u dS \right) d\rho, \tag{11}$$

$$= u(x) \int_0^r w(\rho) \left(\int_{\partial B_\rho(x)} dS \right) d\rho, \tag{12}$$

$$= u(x) \int_{B_r(x)} w(\rho) dV. \tag{13}$$

In other words, weighting the solution value with any weighting function that only varies along the radius in the ball $B_r(x)$ yields an unbiased estimate of the solution value and the only difference is a change in normalization factor from $|B_r(x)|$ to $W = \int_{B_r(x)} w(\rho) dV$ (i.e., the integral of the weighting function over the ball), which can be computed analytically or by a Monte Carlo estimator:

$$\langle W \rangle = \frac{w(\rho_i)}{p_{B_r(x)}(y_i)}, \tag{14}$$

where $y_i \sim p_{B_r(x)}(y_i)$, and $\rho_i = ||x - y_i||$. At the cost of introducing some bias, this estimate can also be computed by reusing the weights $w(\rho_i)$ (i.e., the same samples as those used for computing the weighted average of the solution value $u(y_i)$). In our implementation, we employ the latter biased solution and normalize by the summed weights as this produces visually smoother results.

Figure 3 (1st vs. 3rd row) demonstrates weighting by a Gaussian function with standard deviation equal to the radius of the ball $\sigma^2 = r$. However, any arbitrary radial function is also valid. The differences are most visible for smaller cache sizes around the black arms of the cross.

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

6. Non-uniform Sampling

As noted before, the variance on the solution estimates gathered from the non-weighted mean value cache is inversely proportional to (the square root of) the volume of the ball when the sample points z_i are uniformly distributed in Ω. The radius of the ball is determined by the shortest distance to the boundary $\partial\Omega$. Consequently, the solution estimates closer to the boundary exhibit more variance. This is suboptimal as we would like to have a roughly constant number of samples for any ball in Ω. This can be achieved by using a non-uniform distribution for z_i that is inversely proportional to the volume of the largest ball at z_i. To avoid division by zero, we introduce a μ-shell around the boundary in which we sample uniformly. Furthermore, to avoid undersampling far away from the boundary, we impose a minimum sampling density λ. Combining all yields a sampling of z_i proportional to:

$$z_i \sim \begin{cases} \max\left(\frac{|B_\mu(z_i)|}{Q|B_{r_i}(z_i)|}, \frac{\lambda}{Q}\right) & \text{if } r_i > \mu \\ \frac{1}{Q} & \text{otherwise} \end{cases} \tag{15}$$

where Q is an appropriate normalization factor such that the pdf integrates to one over Ω. In practice, we directly sample z_i using rejection sampling. We first uniformly sample a candidate position, as well as an additional acceptance random variable $\xi \in [0,1]$. If ξ is less than the unnormalized pdf in Equation (15) (i.e., with $Q = 1$), then we accept the sample. In our implementation we set $\mu = 2\varepsilon \times 10^4 = 0.02$ and $\lambda = 0.02$;

A key challenge when evaluating Equation (8) for a point x is that we need to know the pdf $p_{B_r(x)}(z_i)$ for each sampled point z_i. This pdf has the same shape as Equation (15) *inside* the ball. Hence, only an appropriate normalization factor Q needs to be determined such that Equation (15) integrates to one over the ball. An analytical expression for Q is difficult because both the integration domain (i.e., ball) and Equation (15) depend on the geometry of the boundary $\partial\Omega$. At the cost of introducing some bias, we solve this problem by using weighted importance sampling [BSW00]:

$$u(x) \approx \frac{\sum_i \frac{\langle u(z_i)\rangle}{p_\Omega(z_i)}}{\sum_i \frac{1}{p_\Omega(z_i)}}. \tag{16}$$

Note that the normalization factor Q, the number of samples N, and the ball size $B_{r(x)}$ cancel out. As we will empirically show in Section 9, due to the smoothness of the solution $u(x)$, the impact of the bias is small in practice.

Figure 3 compares the mean value cache solution estimates of a Laplace PDE on the OPEN DIAMOND & CROSS boundary conditions obtained with uniform and non-uniform placement of 25,000, 50,000, and 150,000 cached WoS-walks. We show both weighted and unweighted solution estimates to better demonstrate the impact of non-uniform sampling. We include weighing in Equation (16) by replacing $\frac{1}{p_\Omega(z_i)}$ with its weighted version $\frac{w(z_i)}{p_\Omega(z_i)}$. At low sampling rates, uniform sampling provides a better estimate away from the boundary (best visible in the 25,000 WoS walk case). However, non-uniform sampling provides a better estimate close to the boundary, and it converges more quickly to an artifact-free solution; at 150,000 WoS-walks (a little bit more than a walk on average for every two evaluation points) there are still minor sampling artifacts

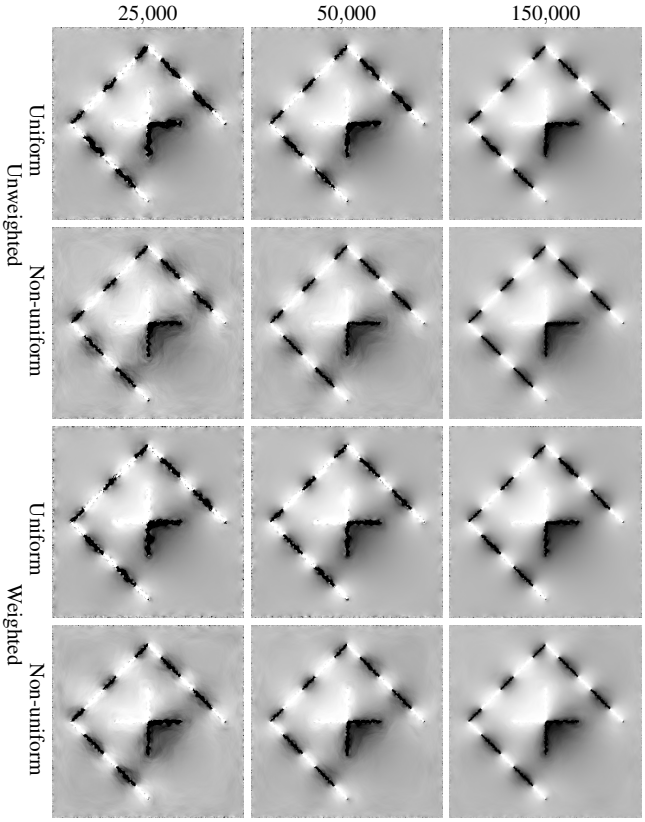

Figure 3: *Comparison of solution estimates of a Laplace PDE on the* OPEN DIAMOND & CROSS *boundary conditions computed with (weighted and non-weighted) mean value caching for uniform versus non-uniform placement of* 25,000*,* 50,000*, and* 150,000 *WoS-walks. Non-uniform sampling produces better results close to boundaries at the cost of a less smooth result away from the boundary. This can be alleviated with weighting (e.g., below the black leg of the cross at* 50,000 *non-uniform samples).*

visible for uniform sampling, while non-uniform sampling exhibits smooth boundaries.

7. Recursive Mean Value Caching

The above strategies aim to reduce the variance during gathering. However, we can also leverage the volume mean value property to reduce the variance of the cached WoS-walks. First, observe that Equation (9) also holds at the sample points z_i themselves. Hence, we can improve the solution value estimates $u(z_i)$ by gathering and averaging the neighboring samples. Second, the updated estimates $u(z_i)$ are also valid unbiased estimates of the solution, and hence the volume mean value property is also applicable, and thus Equation (9) can be applied again. This leads to a recursive algorithm where we diffuse the estimates to the cached points' solution values. Eventually, the estimates will converge to an equilibrium state.

Intuitively, recursively applying Equation (9) will compute the Poisson kernel over the union of all balls $B_{r_i}(z_i)$ for each sample

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

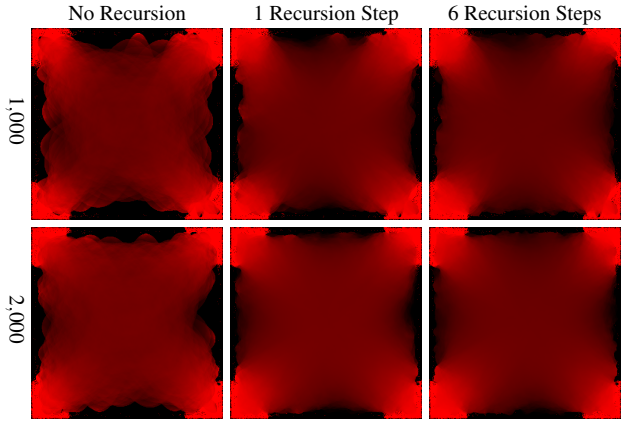

Figure 4: *Effect of recursive variance reduction* within *the cache on the solution estimate of a Laplace PDE with the* RED CORNERS *boundary conditions for* 500×500 *evaluation points and a cache size of* 1,000 *and* 2,000 *uniformly sampled WoS-walks. More recursion steps result in smoother solutions (best visible in darker areas near the boundaries).*

point z_i. Hence, the tighter the union of balls follows the boundary $\partial\Omega$ the closer the Poisson kernel will be to the Poisson kernel over Ω. However, the boundary conditions over the boundary of the union of balls are implicitly determined by the initial WoS-walks. For example, a boundary condition not sampled by the initial WoS-walks cannot contribute to the solution. Characterizing the relation between the boundary approximation on the union of balls and the boundary condition g is an interesting avenue for future research. Furthermore, recursive mean value caching also bears similarity to Finite Element Methods such as iterative radiosity [CW16] albeit starting from very different initial conditions (i.e., Monte Carlo estimates of the solution).

Figure 4 compares solution estimates from cached solution values without recursion, with one recursion step, and with 6 recursion steps for 1,000 and 2,000 samples on a Laplacian PDE with the RED CORNERS boundary conditions for 500×500 evaluation points. Recursive application of the volume mean value property leads to smoother solutions.

8. Source Term Contribution

The previous discussions only consider the case where there is no source term. To derive a volume variant of the source term, we integrate Equation (2) (scaled by $|\partial B_r|/|B_R|$) over all possible radii equal or smaller than the radius R of the largest ball $B_R(x)$ around x, such that the resulting integral equals Equation (4) plus an additional source term $S(x)$. The source term $S(x)$, corresponding to the integration of the 2nd term in Equation (2), can then be written as:

$$S(x) = \int_0^R \frac{|\partial B_r(x)|}{|B_R(x)|} \left(\int_{B_r(x)} f(y) G_r(x,y) dy \right) dr, \quad (17)$$

$$= \int_{B_R(x)} f(y) \left(\int_{||x-y||}^R \frac{|\partial B_r(x)|}{|B_R(x)|} G_r(x,y) dr \right) dy. \quad (18)$$

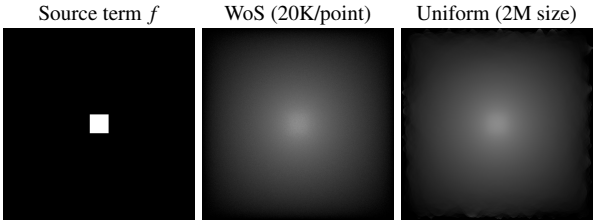

Figure 5: *Comparison of the solution estimate for* 500×500 *evaluation points for a Poisson PDE (left) with a boundary condition* $g = 0$ *around the perimeter, obtained with regular WoS with* 20,000 *samples per evaluation point (middle) and with uniform non-recursive mean volume caching (right) with* $2M$ *cache samples (i.e.,* $1/2{,}500$*th the number of WoS-walks).*

The second step leverages that $G_r(x,y)$ is only defined if $y \in B_r(x)$, i.e., the ball $B_r(x)$ must have radius r larger than the distance $||x-y||$. Denoting the inner integral as $\Gamma_R(x,y)$ yields an expression similar to the regular WoS source term:

$$S(x) = \int_{B_R(x)} f(y) \Gamma_R(x,y) dy. \quad (19)$$

The exact form of $\Gamma_R(x,y)$ depends on the definition of the Green's function $G_r(x,y)$ in the domain Ω. For example, substituting the 2D Green's function on a disc in Equation (18), yields:

$$\Gamma_R(x,y) = G_R(x,y) + \frac{1}{4|B_R(x)|} \left(||x-y||^2 - R^2 \right). \quad (20)$$

As before, we compute the cache samples $u(z_i)$ using a regular WoS-walk that includes the source term (Equation (3)). When gathering, we average all cached samples z_i within the ball $B_R(x)$ as before, but also add the contribution from the source term $S(x)$. We compute the source term similarly to regular WoS, except that we replace the Green's function by $\Gamma_R(x,y)$. In our implementation, we use an equal number of Monte Carlo samples y_k as the number of cache samples in $B_R(x)$ when evaluating $S(x)$.

Figure 5 compares the solution estimates of a Poisson PDE (with boundary conditions $g = 0$ around the perimeter) computed with 2,000,000 uniformly distributed cache samples (without recursion) versus 20,000 WoS-walks per evaluation point at 500×500 resolution (i.e., 2500 times more walks).

9. Analysis

We perform a number of sensitivity studies and analyses to better understand the properties and trade-offs of using mean value caching with respect to the different algorithm parameters.

Convergence Figure 6 plots the convergence of (unweighted) mean value caching with uniform and non-uniform sampling, with and without recursive evaluation, expressed by the mean square error for four Laplace PDEs with different boundary conditions: COLORED DIAMOND, COLORED BOX, ZEPHIR, and LADYBUG. The latter two are examples of a real world graphics application; namely, a vector representation of smooth-shaded images with diffusion curves [OBW*08]. The mean squared errors are computed

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

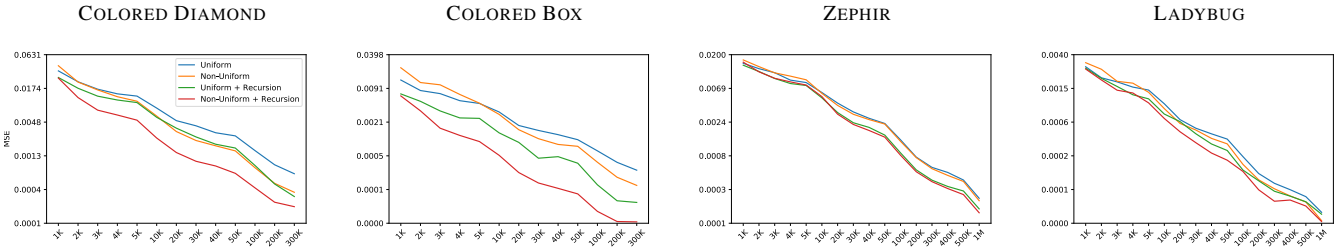

Figure 6: *(Log) MSE plots of mean value caching with uniform and non-uniform sampling, and with and without recursive evaluation of the cache estimates for Laplace PDEs with* 4 *different boundary conditions. PDEs with non-smooth boundary conditions (i.e.,* COLORED DIAMOND *and* COLORED BOX*) benefit more from non-uniform sampling and recursive evaluation than PDEs with smooth boundary conditions (i.e.,* ZEPHIR *and* LADYBUG*).*

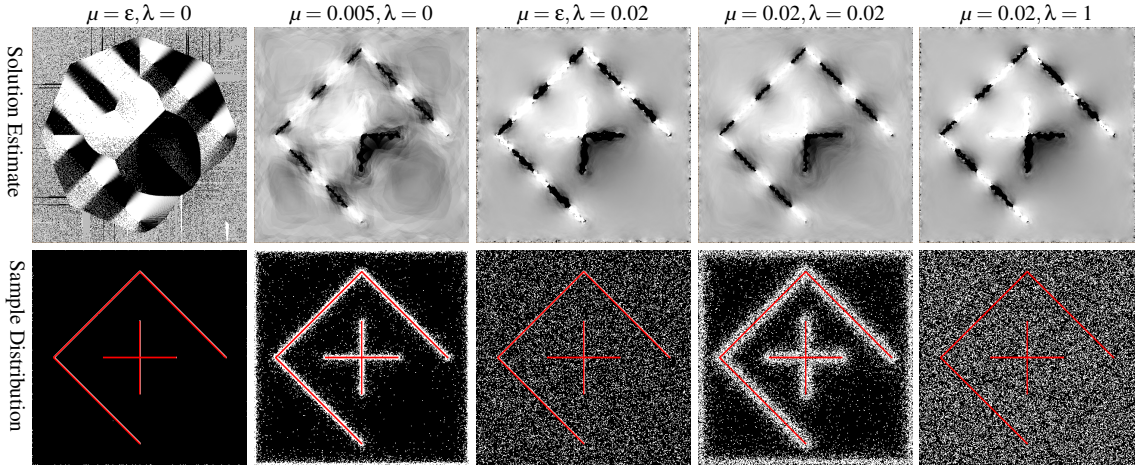

Figure 7: *Impact of* μ *and* λ *parameters on the non-uniform cache sample distribution demonstrated on a Laplace PDE with* OPEN DIAMOND & CROSS *boundary conditions with a cache size of* 30,000*. Setting* μ *and* λ *too low results in a clumping of samples around the boundaries. Conversely, setting* λ *too high converges to a uniform distribution.*

with respect to a solution estimate computed with 20,000 WoS-walks per evaluation point (1000×1000 for the LADYBUG and ZEPHIR and 500×500 for the COLORED DIAMOND and COLORED BOX). Note that we do not include a baseline error-plot for regular WoS as it is unclear how to compute the MSE when we are using fewer than 1 WoS-walk per evaluation point; the maximum number of samples in the above graphs corresponds to approximately 1 cache sample per evaluation point. Figure 8 shows solution estimates for each of the four PDEs for selected cache sizes. From these results, we observe that recursive evaluation produces the lowest MSE overall for an equal cache size. The overall error for non-uniform sampling (without recursion) is typically higher at low sampling rates because it focuses more effort on the boundary regions, whose contribution to the error is relatively small (i.e., fewer evaluation points are near a boundary than away from a boundary). However, visually, the variance around boundaries is more pronounced. For PDEs with relatively smooth boundaries, the difference between uniform and non-uniform sampling, as well as recursive evaluation, is small (e.g., LADYBUG and ZEPHIR). For PDEs with less smooth boundary conditions (e.g., COLORED DIAMOND and COLORED BOX), the differences between the variants are more significant and they benefit more from combining non-uniform sampling and recursive evaluation.

Impact of μ and λ Our prior analysis shows that non-uniform sampling can improve convergence. The non-uniform sampling distribution detailed in Section 6 is determined by two hyper-parameters: μ (the lower bound on the ball radius for which we want to equalize the sample density), and λ (the overall minimum sample density far away from the boundary). Figure 7 demonstrates the impact of μ and λ on a Laplace PDE with the OPEN DIAMOND & CROSS boundary conditions. When μ is set equal to the width of the ε-shell, and λ is set to zero (Figure 7, first column), we see that almost all samples are placed within the ε-shell. This violates the condition that the pdf of a sample z_i cannot be zero inside a ball, and consequently the solution estimate is incorrect. Increasing μ improves results, but places too few samples away from the boundary (2nd column). For a small cache size, this will again converge to the previous case. Setting a non-zero value for λ improves the results, but when μ is small (3rd column), we gain little benefit from the non-uniform sampling. In our implementation, we set both μ and λ to 0.02 (4th column), which strikes a good balance between clumping cache samples close to the boundary and not neglecting the re-

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

Figure 8: *Selected visualizations of (unweighted) mean value caching with uniform and non-uniform sampling and with and without recursive evaluation for a Laplace PDE with with four different boundary conditions.*

gions away from the boundary. Finally, setting $\lambda = 1$ (5th column) is equivalent to uniform sampling.

Statistics Empirically, we find that the average number of cache elements within a ball is linearly proportional to the size of the cache and dependent on the geometry of the boundary conditions. For the OPEN DIAMOND & CROSS boundary, the ratio is approximately 5.3% of the cache samples, while for the more complex geometry of the LADYBUG, the ratio is 10 times smaller (0.5%).

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

The average number of recursive iterations required to reach equilibrium in the cache is less scene-dependent, and in general equilibrium (using a 0.0001 MSE threshold) is reached in approximately 5 iterations.

As noted in Section 4, we switch to regular WoS when the number of cache elements falls below a user-defined threshold (1 in our implementation). The percentage of evaluation points below the threshold is dependent on the cache size. For a Laplace PDE with the OPEN DIAMOND & CROSS boundary conditions with 500×500 evaluation points, 2.6% of evaluation points fall in this category for a cache size of 5,000, 0.62% for a cache size of 25,000, and none for a cache size of 500,000 (i.e., on average 2 WoS-walks per evaluation point).

10. Discussion

Computation Cost The computation cost of mean value caching can be approximated as the sum of generating the cache and the cost of reusing the cached walks at each evaluation point:

$$\mathcal{O}(MVC) = (B \times W \times N) + ((B + C) \times M), \tag{21}$$

where B is the cost of a finding the nearest point on the boundary, W is the average WoS walk length, and C the cost of averaging the cache points in the ball, N is the number of cached points, and M is the number of evaluation points. Hence, the ratio of the number of evaluation points M versus the cache size N plays an important role, as well as the relative cost B of determining the size of the ball versus the cost C of gathering the samples from the cache within a ball. For comparison, the cost for regular WoS is:

$$\mathcal{O}(WoS) = B \times W \times M \times S, \tag{22}$$

where S is the number of Monte Carlo samples per evaluation point. An equal run-time comparison is difficult as timings are implementation and PDE dependent. For example, for a PDE with constant boundary values, regular WoS requires fewer samples to achieve low variance. Moreover, depending on the distribution of the cache samples, different data structures can yield more efficient results. For example, if the cache samples are densely and regularly distributed, then a grid structure instead of a KD-tree can reduce the cost of sample look-up in a ball to constant time complexity.

In our current implementation, uniformly distributed mean value caching is not competitive compared to regular WoS because we always gather as many cache samples as possible. To illustrate: in the COLORED BOX, the central evaluation point averages approximately $3/4$ of the whole cache, imposing a significant computational overhead. This could be resolved by imposing a maximum number of cache samples to average per evaluation point. The non-uniformly distributed mean value cache does not suffer from this issue because the distribution is chosen such that each ball contains approximately the same number of cache samples. Furthermore, for PDEs with boundaries with simple geometries (such as the ones shown in this paper), the cost B of determining the nearest point on the boundary is negligible and thus $B << C$ which disadvantages mean value caching: $\mathcal{O}(MVC) \sim C \times M$ versus $\mathcal{O}(WoS) \sim W \times M \times S$ with typically $C >> W \times S$; for complex boundary geometry we expect that $B \approx C$. However, even under these unfavorable conditions, we found that recursive mean value caching with non-uniformly distributed cache samples stored in a KD-tree is approximately twice as fast as WoS for equal error on the COLORED BOX boundary conditions.

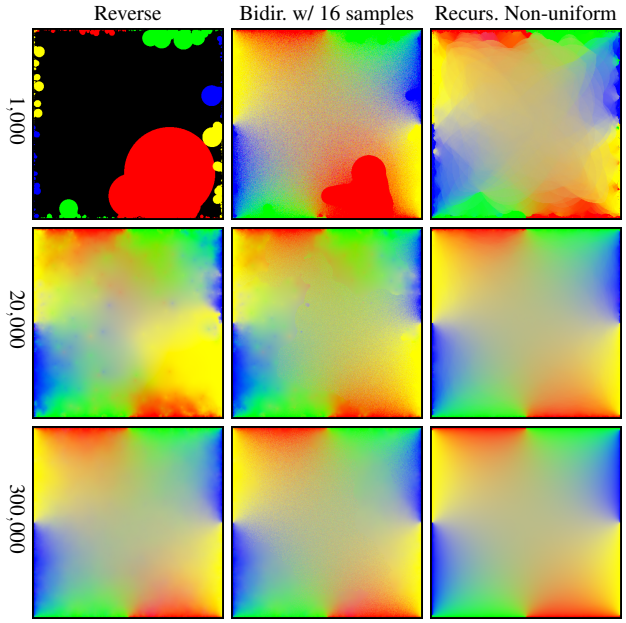

Figure 9: *Comparison between recursive non-uniform mean value caching, reverse WoS, and bidirectional WoS (with 16 forward samples) on a Laplace PDE with the* COLORED BOX *boundary conditions. Even at* 300,000 *walks, reverse WoS shows artifacts near boundaries (e.g., in the blue region at the top right), and bidirectional WoS exhibits significant Monte Carlo noise at the center.*

Similarly, a comparison with reverse WoS [QSBJ22] is difficult too. Reverse WoS is more efficient for sparse boundary conditions, but it becomes less efficient on dense boundary conditions. The bidirectional 2-pass WoS algorithm [QSBJ22] combines the strengths of both forward and reverse walks. However, if the reverse walk estimates are unreliable, the bidirectional algorithm reverts to classic WoS. Figure 9 shows a comparison of mean value caching versus reverse and bidirectional WoS on a Laplace PDE with the COLORED BOX boundary conditions for an equal number of cached versus reverse walks. This PDE has dense boundary conditions, and both reverse and bidirectional WoS exhibit slower convergence than mean value caching. We argue that reverse WoS and mean value caching are complementary algorithms; combining the strengths of both is an interesting avenue for future research.

Gathering vs. Splatting Reverse WoS (without caching) [QSBJ22] and the boundary value caching method [MSCG23] employ splatting instead of creating a cache. In contrast, our method, as well as the two-pass reverse WoS method [QSBJ22], rely on gathering samples from the cache during evaluation. Splatting has less memory overhead, and typically splatting is more efficient than gathering with a KD-tree and on-par with a grid data structure when used for dense caches. However, splatting requires prior knowledge of the evaluation

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.

points. In contrast, gathering is more effective when the evaluation points are irregularly distributed. Furthermore, gathering allows us to construct the cache before the evaluation points are known, and thus it enables the reuse of the cache over multiple simulations.

Caching sub-walks Currently, we only store the estimate of a complete WoS-walk in the cache. However, each sample point on the ball surface during a WoS-walk also yields an estimate at that point, and thus it could also be added to the cache. We observe that the resulting distribution of cache entries z_i tends to naturally cluster around the boundaries, which could be beneficial for mean value gathering near boundaries. However, a key challenge in caching sub-walks, and an interesting avenue for future research, is efficiently determining the resulting pdf $p_{B_r(x)}(z_i)$ for a cache point z_i in the ball centered around an evaluation point x.

11. Conclusion

We presented mean value caching, a method for reusing WoS-walks when evaluating a PDE over a domain or region of interest. At the core of our method lies the volume variant of the mean value property. Replacing the initial step in a WoS-walk with an estimate of the volume mean value allows us to connect any point within a ball to its center. Our method is efficient and easy to implement in existing WoS frameworks. To further improve efficiency, we explored the use of the weighted volume mean value property and non-uniform sampling. Finally, we showed that the estimates in the cache can be further improved by recursively applying the volume mean value property on the cache entries.

For future work, we like to explore how mean value caching can be adapted for reverse WoS-walks [QSBJ22], improve efficiency near boundaries by considering non-spherical filter kernels (cf. first pass surfaces [GHD97, MH03]), and investigate recursive filtering of noisy WoS estimates, e.g., using a small fixed ball size instead of the largest possible ball for fast parallel GPU implementations.

Acknowledgments This work was supported by NSF grant IIS-1909028.

References

[BSW00] BEKAERT P., SBERT M., WILLEMS Y. D.: Weighted importance sampling techniques for monte carlo radiosity. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000, Brno, Czech Republic, June 26-28, 2000* (2000), Eurographics, Springer, pp. 35–46. 5

[BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph. 39*, 4 (Aug 2020). 2

[CW16] COHEN M. F., WALLACE J. R.: *Radiosity and Realistic Image Synthesis*. Morgan Kaufmann Publishers Inc., 2016. 6

[DKH*14] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable realistic rendering with many-light methods. *Comput. Graph. Forum 33*, 1 (Feb 2014), 88–104. 2

[GHD97] GIVEN J. A., HUBBARD J. B., DOUGLAS J. F.: A first-passage algorithm for the hydrodynamic friction and diffusion-limited reaction rate of macromolecules. *The Journal of chemical physics 106*, 9 (1997), 3761–3771. 3, 4, 10

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. *ACM Trans. Graph. 28*, 5 (Dec 2009), 1–8. 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. 27*, 5 (Dec 2008). 2

[JDZJ08] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. *ACM Trans. Graph. 27*, 1 (2008), 7:1–7:11. 2

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques '96* (1996), pp. 21–30. 2

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph. 34*, 4 (Jul 2015). 2

[Kel97] KELLER A.: Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, p. 49–56. 2

[MH03] MASCAGNI M., HWANG C.-O.: ε-shell error analysis for "walk on spheres" algorithms. *Mathematics and computers in simulation 63*, 2 (2003), 93–104. 3, 4, 10

[MSCG23] MILLER B., SAWHNEY R., CRANE K., GKIOULEKAS I.: Boundary value caching for walk on spheres, 2023. arXiv:2302.11825. 2, 9

[Mul56] MULLER M. E.: Some continuous Monte Carlo methods for the dirichlet problem. *The Annals of Mathematical Statistics 27*, 3 (1956), 569 – 589. 1

[Nea04] NEALEN A.: An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. 1, 2

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (Aug 2008), 1–8. 6

[OLK*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Comp. Graph. Forum* (2021). 2

[QSBJ22] QI Y., SEYB D., BITTERLI B., JAROSZ W.: A bidirectional formulation for walk on spheres. *Comp. Graph. Forum 41*, 4 (2022), 51–62. 1, 2, 9, 10

[RLSO*22] RIOUX-LAVOIE D., SUGIMOTO R., ÖZDEMIR T., SHIMADA N. H., BATTY C., NOWROUZEZAHRAI D., HACHISUKA T.: A Monte Carlo method for fluid simulation. *ACM Trans. Graph. 41*, 6 (Nov 2022). 2

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust Denoising using Feature and Color Information. *Comp. Graph. Forum* (2013). 2

[SC20] SAWHNEY R., CRANE K.: Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Trans. Graph. 39*, 4 (Aug 2020). 1, 2, 3

[SMGC23] SAWHNEY R., MILLER B., GKIOULEKAS I., CRANE K.: Walk on stars: A grid-free Monte Carlo method for PDEs with neumann boundary conditions, 2023. arXiv:2302.11815. 2, 4

[SSJC22] SAWHNEY R., SEYB D., JAROSZ W., CRANE K.: Grid-free Monte Carlo for PDEs with spatially varying coefficients. *ACM Trans. Graph. 41*, 4 (Jul 2022). 2

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1988, Atlanta, Georgia, USA, August 1-5, 1988* (1988), Beach R. J., (Ed.), ACM, pp. 85–92. 2

[YVJ22] YILMAZER E. F., VICINI D., JAKOB W.: Solving inverse PDE problems using grid-free Monte Carlo estimators, 2022. arXiv:2208.02114. 2

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Comput. Graph. Forum 34*, 2 (2015), 667–681. 2

© 2023 The Authors.
Proceedings published by Eurographics - The European Association for Computer Graphics.