Projected Walk on Spheres: A Monte Carlo Closest Point Method for Surface PDEs

RYUSUKE SUGIMOTO, University of Waterloo, Canada
NATHAN KING, University of Waterloo, Canada
TOSHIYA HACHISUKA, University of Waterloo, Canada
CHRISTOPHER BATTY, University of Waterloo, Canada

We present *projected walk on spheres* (PWoS), a novel pointwise and discretization-free Monte Carlo solver for surface PDEs with Dirichlet boundaries, as a generalization of the *walk on spheres* method (WoS) [Muller 1956; Sawhney and Crane 2020]. We adapt the recursive relationship of WoS designed for PDEs in volumetric domains to a volumetric neighborhood around the surface, and at the end of each recursion step, we project the sample point on the sphere back to the surface. We motivate this simple modification to WoS with the theory of the closest point extension used in the closest point method. To define the valid volumetric neighborhood domain for PWoS, we develop strategies to estimate the local feature size of the surface and to compute the distance to the Dirichlet boundaries on the surface extended in their normal directions. We also design a mean value filtering method for PWoS to improve the method's efficiency when the surface is represented as a polygonal mesh or a point cloud. Finally, we study the convergence of PWoS and demonstrate its application to graphics tasks, including diffusion curves, geodesic distance computation, and wave propagation animation. We show that our method works with various types of surfaces, including a surface of mixed codimension.

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

Additional Key Words and Phrases: projected walk on spheres, walk on spheres, surface partial differential equations, closest point method, Monte Carlo methods

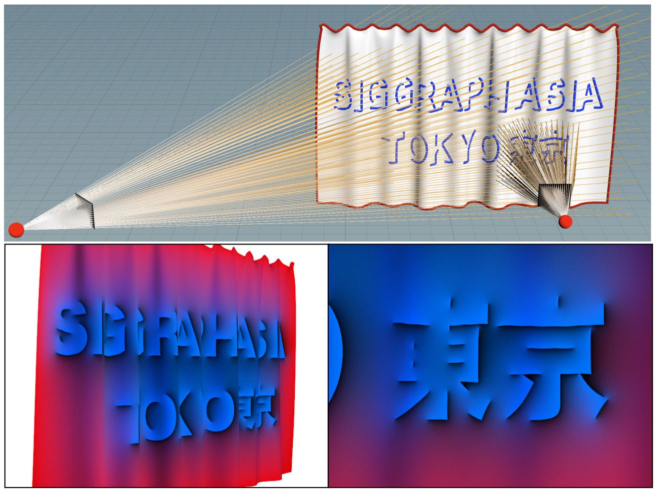


Fig. 1. View-dependent diffusion curves with PWoS. Using our method, we solve the Laplace equation on a curved surface in a view-dependent manner. The pointwise and discretization-free nature of PWoS allows for the evaluation of the colors only at visible points where the object color is required by a shading algorithm with stochastic pixel-filtering.

1 INTRODUCTION

Methods to solve *surface partial differential equations* (PDEs) have become ubiquitous tools in computer graphics research and production. They are used for surface editing [Desbrun et al. 1999], texture synthesis [Turk 1991], surface fluid animation [Stam 2003], geodesic distance computation [Crane et al. 2013], and diffusion curves on surfaces [Bartsch et al. 2023; De Goes et al. 2022] today. A common approach to tackle these problems is to discretize the surface and solve a globally coupled linear system using discrete surface differential operators.

For *volumetric* PDEs, Monte Carlo methods have recently garnered significant attention in the graphics community due to their unique advantages over traditional discretization-based PDE solvers, including the ability to estimate solution values in a pointwise, spatial discretization-free manner. One such method is the *walk on spheres* (WoS) method [Muller 1956] introduced to graphics by Sawhney and Crane [2020]. They primarily focused on the (constant coefficient) Poisson and screened Poisson equations in a volumetric domain, and follow-up work likewise emphasizes volumetric problems. In the present work, we consider instead the problem of solving *surface PDEs*.

Sawhney and Seyb et al. [2022] proposed an extension of WoS to support second-order linear elliptic PDEs with spatially varying coefficients, and as one application, they demonstrated a method to solve the Laplace equation on a 2D surface embedded in 3D. However, this approach requires that the conformal parametrization of the surface be readily available, limiting the method's applicability.

We propose a simpler generalization of WoS for surface PDEs, the *projected walk on spheres* (PWoS) method, which only assumes the availability of a closest surface point query and an unoriented surface normal direction query. PWoS supports Dirichlet boundary conditions and inherits the advantages of WoS: PWoS is a pointwise, discretization-free Monte Carlo method. Since our method does not require the meshing of the surface, it is particularly advantageous over traditional approaches, such as the finite element method, when the computation can be localized and when the surface is given as an implicit representation, such as a signed distance function. The resulting solution is also free of mesh-dependent discretization artifacts, such as from linear interpolation, as we show in Fig. 1. Compared to WoS, which performs walks on spheres inside the domain, PWoS performs walks on spheres inside a Cartesian embedding neighborhood domain around the surface. After each

Authors' addresses: Ryusuke Sugimoto, University of Waterloo, Canada, rsugimot@uwaterloo.ca; Nathan King, University of Waterloo, Canada, n5king@uwaterloo.ca; Toshiya Hachisuka, University of Waterloo, Canada, toshiya.hachisuka@uwaterloo.ca; Christopher Batty, University of Waterloo, Canada, christopher.batty@uwaterloo.ca.

step of the walk, it *projects* the sampled point on the sphere onto the surface. We motivate this simple modification to the original WoS through its connection to the *closest point method* (CPM) [März and Macdonald 2012; Ruuth and Merriman 2008].

Furthermore, inspired by the mean value filtering method for WoS by Bakbouk and Peers [2023], we design a mean value filtering method with a discrete basis function to allow more efficient estimation of solutions when the surface is represented as a polygonal mesh or a point cloud. To confirm PWoS's accuracy, we perform convergence studies of the method applied to the surface Poisson and screened Poisson equations. Finally, we demonstrate its use in several representative graphics applications, including diffusion curves, geodesic distance computation, and wave propagation animation.

In summary, our key contributions are:

- Our novel PWoS algorithm that generalizes the WoS method to solve surface PDEs with Dirichlet boundaries, supported by the theory of the closest point extension.
- A mean value filtering method for PWoS with a discrete basis for efficiency improvement.
- Evaluation of PWoS with convergence studies and graphics applications.

2 BACKGROUND

This section briefly reviews the two core mathematical ideas on which our method is based.

2.1 Walk on Spheres

WoS solves volumetric PDEs such as the Poisson equation over a Cartesian domain in $\mathbb{R}^d$. Consider a d-ball $B_r(\mathbf{x})$, centered at $\mathbf{x}$ with radius r, fully contained within the domain. The integral equation

$$u(\mathbf{x}) = \frac{1}{|\partial B_r(\mathbf{x})|} \int_{\partial B_r(\mathbf{x})} u(\mathbf{y})\,\mathrm{d}\mathbf{y} + \int_{B_r(\mathbf{x})} f(\mathbf{z}) G(\mathbf{x}, \mathbf{z})\,\mathrm{d}\mathbf{z} \tag{1}$$

holds for the Poisson equation $\Delta u = f$ in general, where $|\partial B_r(\mathbf{x})|$ denotes the surface area of the sphere that bounds the ball $B_r(\mathbf{x})$ and G denotes the Green's function for the Poisson equation on $B_r(\mathbf{x})$ [Sawhney and Crane 2020]. On the right-hand side, the first term is a boundary integral over the $(d-1)$-sphere, and the second term is a volume integral over the d-ball. If we perform Monte Carlo integration of the first term by uniformly sampling a point on the sphere and of the second term by sampling N_V points $\mathbf{z}_i$ inside the ball with probability density function (PDF) $p(\mathbf{z}_i)$, we get the recursive relationship used in WoS:

$$\widehat{u}(\mathbf{x}) = \widehat{u}(\mathbf{y}) + \frac{1}{N_V} \sum_{i=1}^{N_V} \frac{G(\mathbf{x}, \mathbf{z}_i) f(\mathbf{z}_i)}{p(\mathbf{z}_i)}, \tag{2}$$

where the hat notation indicates that a variable is a Monte Carlo estimate. The first term on the right-hand side is also a Monte Carlo estimate because the solution $u(\mathbf{y})$ is unknown at point $\mathbf{y}$ in general. At each recursion step, WoS applies this recursive relationship to the largest ball inside the domain bounded by Dirichlet boundaries. It terminates the recursion when the sample point $\mathbf{x}$ during the recursion falls within a small distance ϵ of the domain boundary, by using the known solution at the closest boundary point $\overline{\mathbf{x}}$ as the solution estimate: $\widehat{u}(\mathbf{x}) = u(\overline{\mathbf{x}})$. WoS thereby estimates the solution at each evaluation point independently, offering intrinsic parallelism. Our method generalizes WoS, originally proposed for volumetric PDEs, by incorporating the closest point extension theory of CPM.

2.2 Surface PDEs and Closest Point Extension

Consider the Poisson equation defined on a surface $\mathcal{S}$ embedded in $\mathbb{R}^d$ such that $\dim(\mathcal{S}) < d$:

$$\Delta_{\mathcal{S}} u_{\mathcal{S}}(\mathbf{y}) = f_{\mathcal{S}}(\mathbf{y}), \quad \mathbf{y} \in \mathcal{S}, \tag{3}$$

where $\Delta_{\mathcal{S}}$ denotes the Laplace-Beltrami operator. For convenience, we will use the word surface to refer to any nonzero codimension object in $\mathbb{R}^d$, including mixed-codimension objects. One typically solves such a surface PDE by discretizing the differential operator and solving a sparse linear system. For triangle meshes one can use the cotangent Laplacian [MacNeal 1949]; for other surface representations, a corresponding discrete Laplacian must be defined [Belkin et al. 2009; Bunge and Botsch 2023; Sharp and Crane 2020]. The closest point method [Ruuth and Merriman 2008] instead addresses the surface PDE (Eq. 3) in a more general way by changing the domain to an embedding space, which is a Cartesian neighborhood surrounding the original surface. CPM then solves an *embedding PDE* defined on the embedding space, whose solution when restricted to points $\mathbf{y} \in \mathcal{S}$ is the solution $u_{\mathcal{S}}(\mathbf{y})$ to the original surface PDE. We briefly summarize CPM theory and refer readers to work by King et al. [2023] for an in-depth review of CPM.

We first assume that $\mathcal{S}$ is smooth and define the closest point query to the surface, for $\mathbf{x} \in \mathbb{R}^d$, as

$$\mathrm{cp}_{\mathcal{S}}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{S}} \|\mathbf{x} - \mathbf{y}\|_2. \tag{4}$$

In general, the mapping $\mathrm{cp}_{\mathcal{S}}(\mathbf{x})$ may not be unique: there may exist more than one closest point for a given $\mathbf{x}$. We define the neighborhood $\mathcal{N}(\mathcal{S})$ where the closest point function is unique as $\mathcal{N}(\mathcal{S}) = \left\{\mathbf{x} \in \mathbb{R}^d \;\middle|\; \|\mathbf{x} - \mathrm{cp}_{\mathcal{S}}(\mathbf{x})\|_2 < \mathrm{LFS}\left(\mathrm{cp}_{\mathcal{S}}(\mathbf{x})\right)\right\}$, where $\mathrm{LFS}(\mathbf{y})$ is the local feature size at point $\mathbf{y} \in \mathcal{S}$ defined as the minimum Euclidean distance from $\mathbf{y}$ to the medial axis $\mathrm{med}(\mathcal{S})$ [Amenta and Bern 1999]. The medial axis $\mathrm{med}(\mathcal{S})$ is defined as the set of points in $\mathbb{R}^d$ where there is more than one closest point. Note that when $\mathcal{S}$ is a watertight surface the definition of the medial axis that we use contains both the interior part that is bounded by $\mathcal{S}$ and the exterior part that lies outside the bounded domain.

Within the neighborhood $\mathcal{N}(\mathcal{S})$, or a subset of it, surface differential operators can be replaced by Cartesian differential operators with *closest point extensions* (see [März and Macdonald 2012; Ruuth and Merriman 2008]). The closest point extension operator E extends surface functions onto $\mathcal{N}(\mathcal{S})$ to be constant in the normal direction of $\mathcal{S}$ and is defined as $Eu_{\mathcal{S}}(\mathbf{x}) = u_{\mathcal{S}}(\mathrm{cp}_{\mathcal{S}}(\mathbf{x}))$. For functions $u \in \mathcal{N}(\mathcal{S})$ the extension E acts on the restriction of u to the surface, i.e., $Eu = E(u|_{\mathcal{S}})$. The Laplace-Beltrami operator in Eq. 3 is equivalent to the following:

$$\Delta_{\mathcal{S}} u_{\mathcal{S}}(\mathbf{y}) = \Delta[Eu_{\mathcal{S}}](\mathbf{y}), \quad \mathbf{y} \in \mathcal{S}. \tag{5}$$

To define the embedding PDE on $\mathcal{N}(\mathcal{S})$, we also extend $f_{\mathcal{S}}$ as $f(\mathbf{x}) = Ef_{\mathcal{S}}(\mathbf{x})$. The equation $\Delta[Eu_{\mathcal{S}}](\mathbf{x}) = f(\mathbf{x})$, for $\mathbf{x} \in \mathcal{N}(\mathcal{S})$, is ill-posed because f is constant in the normal direction of $\mathcal{S}$ but $\Delta[Eu_{\mathcal{S}}]$ is not guaranteed to be. Therefore, the embedding PDE for Eq. 3 becomes

$$\Delta[Eu_{\mathcal{S}}](\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \quad \mathbf{x} \in \mathcal{N}(\mathcal{S}), \tag{6}$$

where $g(\mathbf{x})$ is a function that compensates for $\Delta[Eu_{\mathcal{S}}]$ not being constant in the normal direction of $\mathcal{S}$. The function $g(\mathbf{x})$ is nonzero for $\mathbf{x} \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$ and $g|_{\mathcal{S}} = 0$ to ensure Eq. 6 is consistent with the surface PDE (Eq. 3) on $\mathcal{S}$. Any function g with these conditions has the form $g(\mathbf{x}) = \gamma(v(\mathbf{x}) - Ev(\mathbf{x}))$, where $\gamma \in \mathbb{R}$ and $\gamma \neq 0$.

The Macdonald-Brandman-Ruuth approach (see [Chen and Macdonald 2015, Section 2.3]) takes $v|_{\mathcal{S}} = u_{\mathcal{S}}$ to allow Eq. 6 to be written as an equation in one unknown, $v(\mathbf{x})$, since $Eu_{\mathcal{S}} = Ev$ (but importantly $v \neq Ev$ except on $\mathcal{S}$). We instead do not restrict $v|_{\mathcal{S}}$ to be $u_{\mathcal{S}}$ and interpret $g(\mathbf{x})$ as a modification to the source term $f(\mathbf{x})$, then solve for the unknown solution $u(\mathbf{x}) = Eu_{\mathcal{S}}$ in Eq. 6. As proven by von Glehn et al. [2013, Section 3.2], $u(\mathbf{x}) = Eu(\mathbf{x})$ since $u(\mathbf{x})$ is the extension of a surface function. The property that $u(\mathbf{x}) = Eu(x) = u(\mathrm{cp}_{\mathcal{S}}(\mathbf{x}))$ allows our projection step during the walk in PWoS, detailed in Section 3.

We show through our numerical examples that taking $g(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{N}(\mathcal{S})$, provides qualitatively correct results for graphics applications and quantitatively convergent results in most examples in Section 5. However, the choice of $g(\mathbf{x}) = 0$ causes Eq. 6 to be ill-posed as discussed above, and we observe some bias in some convergence studies in Section 5.1 when f is complex. Interesting future work would involve constructing a more accurate $g(\mathbf{x})$ function to improve convergence.

In the traditional CPM, one solves the embedding PDE inside a narrow tubular subset of $\mathcal{N}(\mathcal{S})$ that is within a constant distance to $\mathcal{S}$. For the typical grid-based variant, the tubular subset is spatially discretized with a grid of uniform spacing. Interpolation and finite differences are applied on the grid to approximate the closest point extension and the spatial Cartesian differential operators, respectively, and then the resulting linear system is solved. Other variants of CPM [Cheung et al. 2015; Petras and Ruuth 2016; Piret 2012] also require some discretization within $\mathcal{N}(\mathcal{S})$. Thus, while traditional CPM is agnostic to the specific surface representation, it still discretizes the embedding space and solves a globally coupled system. Moreover, imposing exterior or interior boundary conditions requires tedious grid operations [King et al. 2023]. By contrast, we develop a spatial discretization-free, pointwise Monte Carlo estimator for surface PDEs by incorporating the closest point extension concept into WoS.

When there are Dirichlet boundaries $C \subset \mathcal{S}$, on which the solution $u_{\mathcal{S}}$ is given, one can geometrically extend the boundary itself out into the embedding space in the normal directions, assigning it the same boundary value in accordance with the closest point extension. Note that such boundaries need not coincide with the geometric boundaries of the surface itself. In the context of grid-based CPM, King et al. [2023] discuss how to impose such boundary conditions by duplicating degrees of freedom near the extended boundary. In our work, we devise a method that uses only the closest point function $\mathrm{cp}_C(\mathbf{x})$ to the (pre-extension) boundary C, without the need to construct the extended boundary geometry or perform any complex duplication of degrees of freedom.

3 METHOD

Input. While our algorithm is generalizable to other configurations, we describe our method for the case when $\mathcal{S}$ is embedded in $\mathbb{R}^3$ and $\dim(\mathcal{S}) = 1, 2$. Recall that we use the word surface to refer both to "surfaces" with $\dim(\mathcal{S}) = 1$ (curves) as well as $\dim(\mathcal{S}) = 2$ surfaces. We also allow mixed codimension where parts of the surface have $\dim(\mathcal{S}) = 1$ and the rest of the surface has $\dim(\mathcal{S}) = 2$. We assume that we can query the closest point function $\mathrm{cp}_{\mathcal{S}}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^3$. Additionally, for surfaces with $\dim(\mathcal{S}) = 2$, we assume that we can query the unoriented normal direction of the surface $\mathbf{n}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{S}$. For surfaces with $\dim(\mathcal{S}) = 1$, we assume that we can query the tangential direction of the surface $\mathbf{t}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{S}$. These assumptions are valid for common surface representations, including, but not limited to, polygonal meshes, oriented point clouds, and implicit functions. The theory discussed in Section 2.2 is based on the assumption that $\mathcal{S}$ is smooth; in practice, we observe that applying our technique on discretized surfaces with sharp features behaves well as we show in Section 5.1. Additionally, we assume the Dirichlet boundaries C have a lower dimension than the dimension of $\mathcal{S}$ and support the closest point query cp_C. When solving a two-sided boundary value problem for boundaries with $\dim(C) = 1$, we also assume that we can query the tangent direction of C.

Overview. The core idea of our method is to apply the WoS recursive relationship within $\mathcal{N}(\mathcal{S})$ while utilizing the closest point extension constraint that $u(\mathbf{x}) = u(\mathrm{cp}_{\mathcal{S}}(\mathbf{x}))$. To do so, we modify the walk process to use spheres contained within $\mathcal{N}(\mathcal{S})$ and to *project* the walk position at each recursion step, as illustrated in Fig. 2.

Algorithm 1: Projected Walk on Spheres

```
Input: surface $\mathcal{S}$, boundary $C$, evaluation point $\mathbf{x} \in \mathcal{S}$,
       sample walk count $N_P$, volume sample count $N_V$,
       tolerance $\epsilon$

Function EstimateSolution($\mathcal{S}, C, N_P, N_V, \mathbf{x}, \epsilon$):
    $\mathcal{M} \leftarrow$ medialAxisPointCloud($\mathcal{S}$)    // Section 3.1
    $\widehat{u}_{\text{sum}} \leftarrow 0$
    for $n \leftarrow 1$ to $N_P$ do
        $\widehat{u} \leftarrow$ RecursiveEstimate($\mathcal{S}, \mathcal{M}, C, N_V, \mathbf{x}, \epsilon$)
        $\widehat{u}_{\text{sum}} \leftarrow \widehat{u}_{\text{sum}} + \widehat{u}$
    end
    return $\widehat{u}_{\text{sum}}/N_P$

Function RecursiveEstimate($\mathcal{S}, \mathcal{M}, C, N_V, \mathbf{x}, \epsilon$):
    $\delta \leftarrow$ distanceToBoundary($\mathcal{S}, \mathcal{M}, C, \mathbf{x}$) // Section 3.2
    if $\delta < \epsilon$ then
        return $u(\mathrm{cp}_C(\mathbf{x}))$
    $l \leftarrow$ localFeatureSize($\mathcal{S}, \mathcal{M}, \mathbf{x}$)    // Section 3.1
    $r \leftarrow \min(l, \delta)$
    $\mathbf{y} \leftarrow$ uniformSphereSample(center=$\mathbf{x}$, radius=$r$)
    $\widehat{u}_{\text{sphere}} \leftarrow$ RecursiveEstimate($\mathcal{S}$, med, $C, N_V, \mathrm{cp}_{\mathcal{S}}(\mathbf{y}), \epsilon$)
    $\{\mathbf{z}_1, \ldots \mathbf{z}_{N_V}\} \leftarrow$ ballSample(center=$\mathbf{x}$, radius=$r$)
    $\widehat{u}_{\text{ball}} \leftarrow \frac{1}{N_V} \sum_{i=1}^{N_V} \frac{G(\mathbf{x},\mathbf{z}_i) f(\mathrm{cp}_{\mathcal{S}}(\mathbf{z}_i))}{p(\mathbf{z}_i)}$
    return $\widehat{u}_{\text{sphere}} + \widehat{u}_{\text{ball}}$
```

The problem we solve is the embedding PDE $\Delta u(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ within $\mathcal{N}(\mathcal{S})$. The Monte Carlo estimate of Eq. 2 holds by assuming $g(\mathbf{x}) = 0$ because the embedding PDE is defined with the Cartesian differential operator. To estimate the surface PDE's solution at point $\mathbf{x} \in \mathcal{S}$, we consider a 3D ball centered at $\mathbf{x}$ and fully contained inside $\mathcal{N}(\mathcal{S})$. Theoretically, it should be the largest ball fully contained inside $\mathcal{N}(\mathcal{S})$ that does not cross the extended Dirichlet boundaries C, to minimize the number of steps needed to reach the boundary. We determine the radius of such a ball by taking the minimum of a conservative (under-)estimate of the local feature size at point $\mathbf{x}$ (Section 3.1) and the distance to the (extended) Dirichlet boundaries (Section 3.2). In Eq. 2, the Monte Carlo estimate of the solution on the sphere, $\widehat{u}(\mathbf{y})$, needs to be evaluated at point $\mathbf{y}$, which does not lie on $\mathcal{S}$ in general. The closest point extension constraint of u provides a convenient relationship here: the embedding PDE's solution at point $\mathbf{y}$ should coincide with the surface PDE's solution at the projected point, $\mathrm{cp}_{\mathcal{S}}(\mathbf{y})$. We can therefore project the point $\mathbf{y}$ onto $\mathcal{S}$ at the end of each recursion step hence $\widehat{u}(\mathbf{y}) = \widehat{u}(\mathrm{cp}_{\mathcal{S}}(\mathbf{y}))$. After this projection at the end of each step, we continue the recursion. The source term similarly uses the closest point projection for the closest point extension, replacing the recursive relationship of WoS (Eq. 2) with

$$\widehat{u}(\mathbf{x}) = \widehat{u}(\mathrm{cp}_{\mathcal{S}}(\mathbf{y})) + \frac{1}{N_V} \sum_{i=1}^{N_V} \frac{G(\mathbf{x}, \mathbf{z}_i) f(\mathrm{cp}_{\mathcal{S}}(\mathbf{z}_i))}{p(\mathbf{z}_i)}. \quad (7)$$

We choose $p(\mathbf{z}_i) \propto 1/\|\mathbf{x} - \mathbf{z}_i\|_2$ in our implementation.

Analogous to the original WoS method, we terminate the recursion when the point $\mathbf{x}$ falls within a distance ϵ of the (extended) Dirichlet boundary by taking the boundary value $u(\mathrm{cp}_{\mathcal{S}}(\mathbf{x}))$. We provide pseudocode for an instance of our algorithm in Alg. 1, where we highlight the difference between our proposed method and WoS. We also provide a visualization of a potential path of our algorithm when $\mathcal{S}$ is embedded in $\mathbb{R}^2$ in Fig. 2. Notably, PWoS is a generalization of the WoS algorithm: when $\dim(\mathcal{S}) = d$ (i.e., the codimension-zero case), the local feature size is infinite, the distance to the Dirichlet boundary C can be computed with the closest point query cp_C, and the last closest point projection of $\mathbf{y}$ has no effect since $\mathrm{cp}_{\mathcal{S}}(\mathbf{y}) = \mathbf{y}$. When $\dim(\mathcal{S}) < d$, in addition to the closest point projection at the end of each recursion step, our algorithm utilizes two nontrivial steps: the local feature size estimation using a medial axis point cloud and the computation of the distance to the (extended) Dirichlet boundary. We discuss these in the following two subsections.

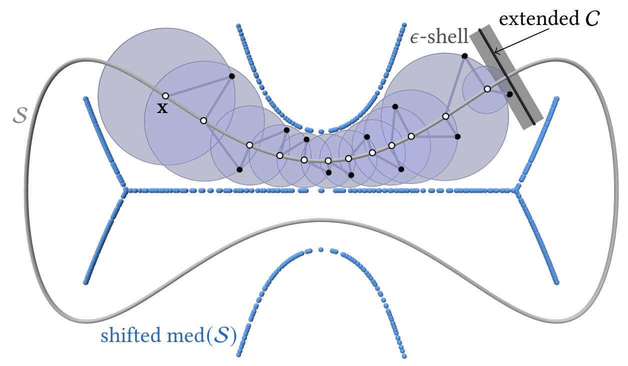


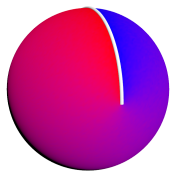
Fig. 2. A PWoS path for the Laplace equation on a gray 1D (curve) surface embedded in 2D space, starting from $\mathbf{x}$ and terminating at the extended Dirichlet boundary C.

3.1 Local Feature Size Estimation

To determine the radius of the sphere centered at $\mathbf{x} \in \mathcal{S}$ that is fully contained inside $\mathcal{N}(\mathcal{S})$ at each recursion step of the walk, we need a conservative lower bound estimate for the local feature size at $\mathbf{x}$. One naive approach would be to use a small enough positive constant value for all $\mathbf{x} \in \mathcal{S}$, similar to the grid-based CPM [Ruuth and Merriman 2008]. This is a valid strategy, but it would often yield a sphere radius smaller than necessary, requiring more recursion steps for the walks to reach a Dirichlet boundary and making the method inefficient. Fig. 3 illustrates a result of our algorithm on a unit sphere using different (artificial) local feature size estimates. The analytical local feature size of this surface is 1 everywhere. Although using any smaller value would still give consistent results, it significantly increases the average step count for the walks. For more complex shapes, one cannot compute the analytical local feature size in general and using a small constant value in its place is inefficient. This issue motivates our need for a better local feature size estimate.

To estimate the local feature size, we compute a point cloud approximation of the medial axis and estimate the local feature size as the distance from any query point $\mathbf{x} \in \mathcal{S}$ to its nearest point in the medial axis point cloud. One could use any local feature size estimation algorithm and/or medial axis extraction algorithm (see e.g., [Tagliasacchi et al. 2016]), such as one that outputs or uses the medial axis' connectivity. Since such methods are often comparatively costly, we employed a simple point cloud-based method, which we describe below.

Medial ball extraction. We first densely scatter points $\mathbf{x}_i$ inside a ball in $\mathbb{R}^3$ having a radius equal to half the length of the diagonal of the bounding box of $\mathcal{S}$, so the entire surface is fully enclosed. For each point $\mathbf{x}_i$, we use the closest point query $\mathrm{cp}_{\mathcal{S}}(\mathbf{x}_i)$ to compute its two opposing normal directions at $\mathrm{cp}_{\mathcal{S}}(\mathbf{x}_i)$. Specifically, we normalize the vector $\mathbf{x}_i - \mathrm{cp}_{\mathcal{S}}(\mathbf{x}_i)$ to get the first direction and invert its direction to get the second one. We then use the method of Ma et al. [2012] to extract a point cloud that represents the medial axis, as follows. For each side of the surface (i.e., each normal), we start with a large sphere tangent to $\mathrm{cp}_{\mathcal{S}}(\mathbf{x}_i)$, whose center necessarily lies on the normal ray. The initial radius of the ball is set to the length of



Local Feature Size	Avg. Step Count
0.99	31.1398
0.5	47.84
0.25	128.981
0.125	462.781
0.0625	1818.73

Fig. 3. Average number of steps required with different conservative local feature size estimates. While any positive value smaller than 1 is valid for this setup, using a local feature size estimate that is too small leads to excessively long walks.

the diagonal of the bounding box of $\mathcal{S}$. Then, we iteratively shrink the size of the sphere, moving its center to maintain tangency at the surface point $\mathrm{cp}_{\mathcal{S}}(\mathbf{x}_i)$, until the closest surface point from the center of the sphere does not change. This algorithm gives *medial balls*, i.e., balls with their centers on the medial axis. As the number of initial scattered points increases, the extracted point cloud balls tend to cover the entire medial axis. While Ma et al. [2012] assumed that the surface is represented as an oriented point cloud, we observed that the algorithm works well with other surface representations by adjusting its termination criteria. See the paper by Ma et al. [2012] and our implementation for details.

Scale axis pruning. Directly using the medial ball centers as the medial axis point cloud does not work well in general when $\mathcal{S}$ contains any noise or artificial sharp corners introduced by the discretization of a smooth surface. We therefore prefer to estimate (only) the *stable* part of the medial axis, which is not affected by any small perturbation of $\mathcal{S}$. A common solution is, therefore, to *prune* unstable components of the medial axis, which itself remains an active research topic [Tagliasacchi et al. 2016]. We take inspiration from the scale axis transform (SAT) [Giesen et al. 2009; Miklos et al. 2010], but design an alternative since SAT considers topology information of the medial axis, which is unnecessary for local feature size estimation. Our alternative is simpler and faster since topology information is omitted. We first scale all the medial balls by a constant factor $s > 1$. Then, for any pair of medial balls $B_{r_1}(\mathbf{x}_1)$ and $B_{r_2}(\mathbf{x}_2)$, we remove the smaller ball from the set of medial balls if it is fully contained in the other. That is, if $s \cdot r_1 < s \cdot r_2 + \|\mathbf{x}_1 - \mathbf{x}_2\|_2$, we remove the ball $B_{r_1}(\mathbf{x}_1)$.

Note that some of the medial balls may have a very large radius before pruning. For example, an exterior medial ball for a surface of a convex shape would have an infinite radius in theory (but our algorithm returns at most the length of the diagonal of the bounding box of $\mathcal{S}$). When such large medial balls are used in our pruning algorithm, they can easily (and undesirably) remove important, stable parts of the medial axis. The original SAT approach was applied only to the interior medial axis of closed surfaces. Therefore, this issue was not observed since the interior medial ball radii are always bounded and proportional to the size of local features of $\mathcal{S}$. To address this problem, we consider each pair of tangent balls generated at the same surface point, and replace the larger one with a tangent ball having the radius of the smaller one. In other words, before we prune the medial axis, we shift the medial ball centers of the larger medial ball in the pair and shrink its radius. In Fig. 2, we visualize the medial ball centers after this shifting operation.

After this pruning, the set of medial ball centers gives the medial axis point cloud we use to estimate the local feature size. Adjusting the scaling parameter s allows us to control the pruning strength. Unless otherwise stated, we use the value $s = 1.15$ for our results.

Conservative and nonzero local feature size. As the medial axis is represented as a point cloud and the nearest point distance may give a larger value than the actual local feature size, we multiply by a small constant (0.9 in our implementation) to ensure a conservative estimate of the local feature size. When there are sharp corners in the geometry, the analytical local feature size is zero, and the walk will become stuck. To prevent this problem, when the estimated local feature size is smaller than a positive constant threshold λ, we return λ as the local feature size estimate instead. This process essentially rounds sharp corners with rounding radius λ. The uniform grid size adopted in grid-based CPM has a similar effect. We do not observe any significant error due to this rounding, as we show in Section 5.1.

3.2 Distance to Extended Dirichlet Boundaries

Dirichlet boundaries C are extended in the normal direction of $\mathcal{S}$ and the solution in the embedding space on this extended boundary is determined by the closest point extension of Dirichlet values on C. Therefore, we need to compute the minimum distance to the extended Dirichlet boundaries, and limit the sphere radius in PWoS further if it is less than the local feature size. To determine the distance to the extended Dirichlet boundary from point $\mathbf{x} \in \mathcal{S}$, we first find the closest point that lies on the boundary before the extension, $\mathrm{cp}_C(\mathbf{x})$. The subset of the extended boundary that is extended from $\mathrm{cp}_C(\mathbf{x})$ takes the shape of a line segment when $\dim(\mathcal{S}) = 2$ and a disk when $\dim(\mathcal{S}) = 1$. We set the line segment's half-length or the disk radius to the local feature size at $\mathrm{cp}_C(\mathbf{x})$ using the algorithm in Section 3.1. We can compute the distance from the point $\mathbf{x} \in \mathcal{S}$ to this line segment or disk without explicitly constructing the extended boundary geometry. When $\dim(\mathcal{S}) = 2$, the distance δ to the extended boundary is given by

$$\begin{aligned} \mathbf{r} &= \mathrm{cp}_C(\mathbf{x}) - \mathbf{x}, \\ \delta &= \|\mathbf{r} - \mathrm{clamp}(\mathbf{r} \cdot \mathbf{n}, -l, l) \cdot \mathbf{n}\|_2, \end{aligned} \tag{8}$$

where $\mathbf{n}$ and l are the surface normal and the local feature size estimate at $\mathrm{cp}_C(\mathbf{x})$, respectively. When $\dim(\mathcal{S}) = 1$, the normal direction is not uniquely defined, so we instead use a similar method based on the surface tangent $\mathbf{t}$:

$$\begin{aligned} \mathbf{q} &= \mathbf{r} - (\mathbf{r} \cdot \mathbf{t})\mathbf{t}, \\ \delta &= \begin{cases} |\mathbf{r} \cdot \mathbf{t}|, & \text{if } \|\mathbf{q}\|_2 < l, \\ \|\mathbf{r} - l \cdot (\mathbf{q}/\|\mathbf{q}\|_2)\|_2, & \text{otherwise.} \end{cases} \end{aligned} \tag{9}$$

3.3 Generalizations

3.3.1 *Screened Poisson Equation.*

We have so far considered only the Poisson equation. For WoS, Sawhney and Crane [2020] proposed a generalization of WoS to the screened Poisson equation $\Delta u - \sigma u = f$, where σ is a positive constant. The embedding PDE for the screened Poisson equation is constructed similarly to Eq. 6 using closest point extensions [Chen and Macdonald 2015, Section 2.3]. Therefore, similar to WoS [Sawhney and Crane 2020], we replace the integral equation (Eq. 1) used in our recursive relationship with

$$u(\mathbf{x}) = \frac{c_{r,\sigma}}{|\partial B_r(\mathbf{x})|} \int_{\partial B_r(\mathbf{x})} u(\mathbf{y})\, \mathrm{d}\mathbf{y} + \int_{B_r(\mathbf{x})} f(\mathbf{z}) G_\sigma(\mathbf{x}, \mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{10}$$

where G_σ is the Yukawa potential and $c_{r,\sigma}$ is a positive number smaller than 1. To evaluate the first term, instead of multiplying the contribution from the recursion by $c_{r,\sigma}$ as suggested by Sawhney and Crane [2020], we use a Russian Roulette strategy: we terminate the path early with probability $1 - c_{r,\sigma}$ with zero contribution and otherwise we use the estimate of the solution without multiplying by $c_{r,\sigma}$. As PWoS is a generalization of WoS, we can apply this Russian Roulette strategy to WoS as well. This scheme allows us to terminate the PWoS path early without waiting for it to reach the boundary and without introducing additional bias. It also makes it possible to apply PWoS to problems without Dirichlet boundaries. We can even use this estimator for the screened Poisson equation as a Tikhonov regularization of the Poisson equation without boundaries: solving the screened Poisson equation with small σ yields an approximate solution to the Poisson equation. Similar regularization ideas appear in multiple contexts [Sabelfeld and Simonov 1994, Section 6.3; Sawhney and Miller et al. 2023].

3.3.2 Divergence Source Term and Gradient Estimation. Many graphics applications, such as the heat method for geodesic distance computation [Crane et al. 2013] and the projection step of fluid simulation [Foster and Metaxas 1996; Stam 1999], give rise to a Poisson equation with a source term expressed as the divergence of a vector field, $f_{\mathcal{S}} = \nabla_{\mathcal{S}} \cdot \mathbf{h}_{\mathcal{S}}$, where $\mathbf{h}_{\mathcal{S}}$ is a vector field defined over the surface. These applications also require the estimation of the gradient of the solution instead of the solution itself. With grid-based CPM, the differential operators defined in the embedding Cartesian domain can be used to solve such problems [Auer et al. 2012; King et al. 2023]. For our PWoS, however, we do not use any embedding grid structure, and we do not assume any specific surface representation, so we cannot use such discrete differential operators.

To solve a problem with a divergence of vector field as the source term, Sugimoto et al. [2024] proposed to use integration by parts to convert the volume integral arising from the source term to a form that does not explicitly depend on the divergence of the vector field. This was done in the context of the walk on boundary method [Sugimoto et al. 2023], which is a Monte Carlo volumetric PDE solver based on a different integral equation formulation, and we adapt this approach to (projected) WoS. To estimate the gradient with PWoS, we can use the gradient estimator for WoS [Sawhney and Crane 2020, Section 3.1], because the solution's gradient is zero in the surface normal directions due to the closest point extension constraint $u(\mathbf{x}) = u(\text{cp}_{\mathcal{S}}(\mathbf{x}))$. The gradient estimator replaces the first step of recursion based on Eq. 1 with one we can derive by taking the gradient of Eq. 1. We discuss the details of these estimators in the supplemental note and demonstrate an application for the geodesic distance computation in Section 5.2.2.

4 MEAN VALUE FILTERING WITH DISCRETE BASIS

The method we described in Section 3 is suitable for evaluating the solution at a few evaluation points independently. To improve the efficiency of the method for many evaluation points (i.e., mesh vertices), it is often desirable to utilize the spatial consistency of the solution. For WoS, a few such methods have been proposed, but those approaches are not trivially applicable to our setup, where we have additional closest point extension constraints between the surface and embedding PDEs. Specifically, the boundary value caching approach [Miller and Sawhney et al. 2023] is based on a boundary integral equation defined in the Cartesian coordinates and is not applicable to surface PDEs. Adaptation of reverse WoS [Qi et al. 2022] or mean value caching [Bakbouk and Peers 2023] to our setting would require a mapping of a PDF on the surface to a PDF on the spheres used in our walk process, and we found it difficult to design such an algorithm for general surfaces.

Inspired by the filtering method of Bakbouk and Peers [2023], we designed a filtering method that uses discrete basis functions defined over a surface represented as a polygonal mesh or point cloud. When we apply Eq. 7 at an evaluation point, if $\widehat{u}(\text{cp}_{\mathcal{S}}(\mathbf{y}))$ is a precomputed Monte Carlo estimate, we do not need to continue the recursion and can simply use the precomputed estimate in its place. In general, we do not have the estimate $\widehat{u}(\text{cp}_{\mathcal{S}}(\mathbf{y}))$ available at $\text{cp}_{\mathcal{S}}(\mathbf{y})$. Thus, we get the estimate by interpolating the estimate of solutions already computed at a discrete set of points by accepting the bias due to interpolation. For example, for a triangle mesh, we use barycentric coordinates as the interpolation basis. This can be interpreted as a PDE-aware smoothing filter of the solution when we consider this process as a weighted average of the solution estimates at nearby evaluation points.

To estimate the solution at all mesh vertices or all points in a point cloud, we first compute an approximate solution to the problem with a low sample count using the method described in Section 3 and apply this filtering step to get an updated estimate. We can also precompute the filtering weights per evaluation point first and apply the same filter a few times to achieve an even smoother solution without too much additional cost. We can similarly design a gradient estimation filter based on Section 3.3.2. While this filtering approach utilizes a discrete basis defined over the surface, we do not use any explicit discrete differential operators and do not need to solve a linear system, which is in contrast to the traditional methods based on discrete Laplacians.

The method of Bakbouk and Peers [2023] similarly designed a smoothing filter for volumetric PDEs. Their method can keep the estimate unbiased by assuming that the evaluation points are sampled with a known PDF and that the original estimates are unbiased, but our method introduces bias due to the use of a discrete basis. We nevertheless observe that, within a reasonable runtime budget, our biased filtering method can reduce the error compared to the PWoS algorithm without filtering. We leave the development of an unbiased variant to future work.

5 RESULTS

We implemented PWoS in Houdini 20.0 [Side Effects Software, Inc. 2023] without GPU acceleration, using its built-in closest point queries. Our implementation is provided as supplemental material.

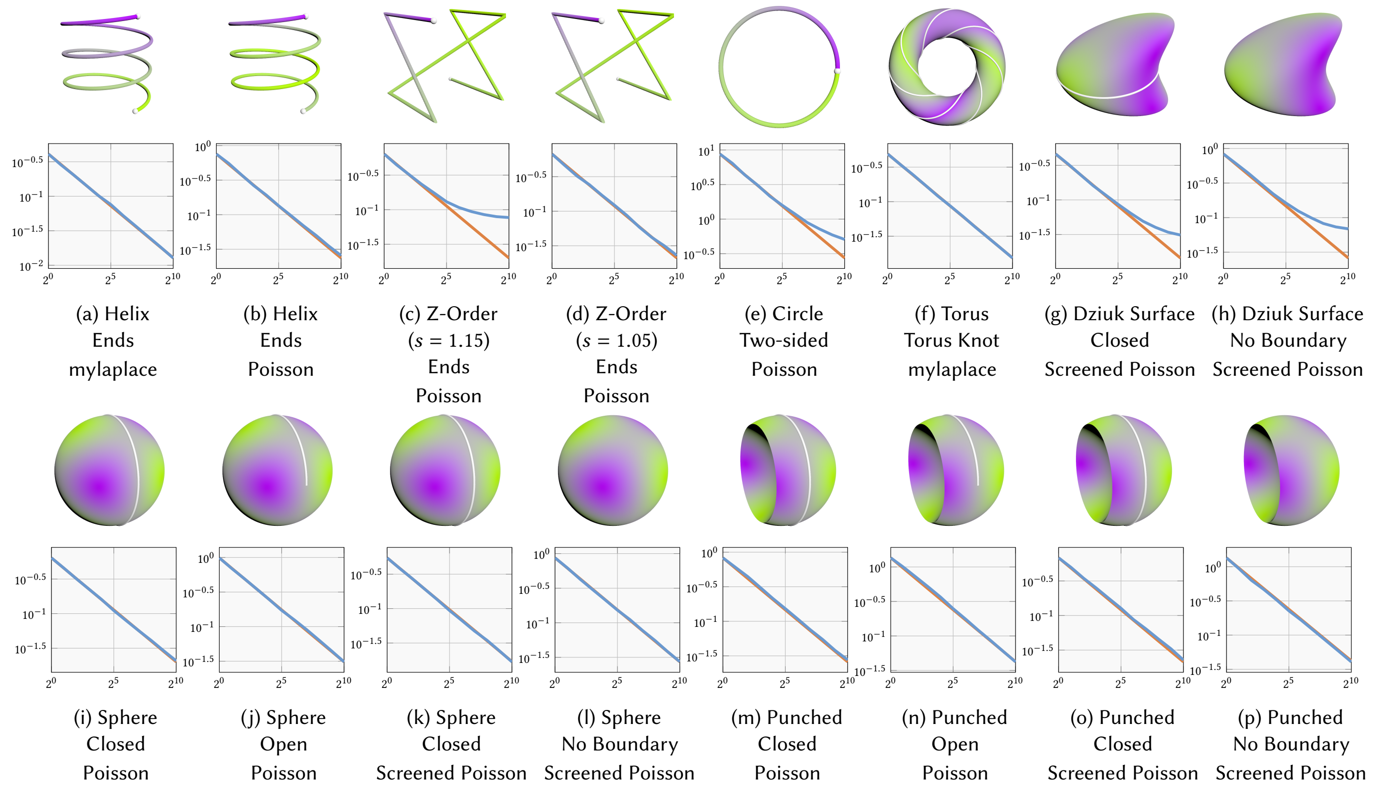


Fig. 4. Error convergence. In each of the examples, we show the visualization of the scene setup (top), error convergence plot (middle), and scene description (bottom). The scene visualizations show the analytical or reference solution of the problem by mapping the range of solution values to the green-to-purple color gradient and placing a white point or curve on the Dirichlet boundary. The vertical axis of the error plot shows the root mean squared error of the estimates at a few points on the surface in a logarithmic scale, and the horizontal axis shows the number of samples N_P in a logarithmic scale. The blue curves show the result of the experiment, and the orange curves show a line that corresponds to the desired $O(1/\sqrt{N_P})$ convergence rate. The scene description indicates the surface shape (top), boundary shape/type (middle), and problem type (bottom). While the expected $O(1/\sqrt{N_P})$ is achieved in most scenes, the bias remains high when an aggressive medial axis pruning parameter is used (c) and when the source term of the problem is relatively complex (e, g, and h). We describe the details of the scene configurations in the supplemental note.

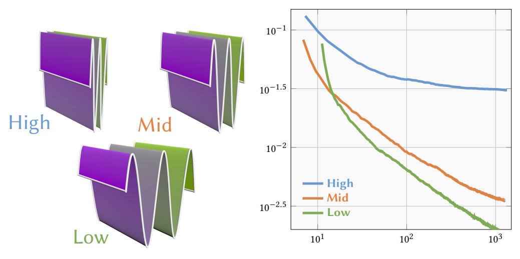


Fig. 5. Effect of local feature size on convergence speed. We bend a rectangular strip of size 10 by 2 units along sinusoidal curves with high, middle, and low frequencies (top left, top right, and bottom, respectively, in the visualization) and solve the Laplace equation. The analytical solution is defined as the distance along the longer edge of the strip from one of the shorter edges. The vertical axis of the convergence plot represents the root mean squared error (RMSE), while the horizontal axis shows the time in seconds measured on a MacBook Pro with an M1 Pro chip. We used 1000 sample evaluation points. The geometry with a larger local feature size allows for faster convergence with lower bias.

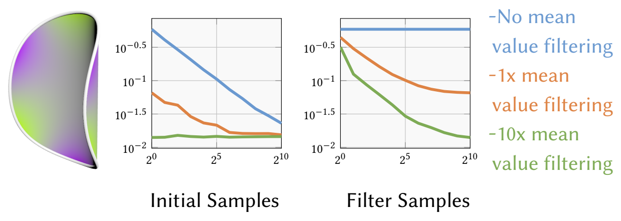


Fig. 6. Mean value filtering error. For the Laplace equation with solution $u = r^3 \sin(3\theta)$ in polar coordinates defined on a unit disk, we curve the disk and measure the error of PWoS with mean value filtering. In each of the two plots, we compare the error without mean value filtering against one application of filtering and ten applications of filtering. The left plot shows the root mean squared error when changing the number of sample paths used to generate the initial estimate with a fixed number of filter samples (1024). The right plot shows the root mean squared error when changing the number of samples used to construct the filter and fixing the number of sample paths used to generate the initial estimate to 1. We observe that applying the mean value filter constructed with a sufficiently large number of samples can reduce the error significantly, even when the initial estimate is constructed with a small sample path count.

5.1 Error Convergence

We conducted error convergence studies of our method using discretized surfaces. In each scene in Fig. 4, we change the number of sample paths N_P to plot the root mean squared error measured against the analytical or reference solution. We do not apply the mean value filtering method in these studies. The scene setup includes Laplace, Poisson, and screened Poisson equations, and both smooth surfaces and surfaces with sharp corners. In all scenes, we use $\epsilon = 0.001$ and $N_V = 32$ except for Laplace equations, for which we use $N_V = 0$. While the method shows the expected convergence rate of $O(1/\sqrt{N_P})$ in most examples, we observed a relatively large bias with problems having more complex source term functions, indicating the need for future work to investigate estimating $g(\mathbf{x})$ from Eq. 6 for such problems.

Fig. 5 compares the convergence of our method for problems with different local feature sizes. We observe that larger local feature size corresponds to faster convergence with lower bias.

Mean value filtering. We run the mean value filtering algorithm on a Laplace equation on a triangulated curved disk surface in Fig. 6. In this setup, we observe that applying the filter multiple times with a filter constructed with a sufficiently high sample count can significantly reduce the error, even when the initial estimate is computed with a small number of samples. As expected, the filter is more effective when constructed with more samples, but the error decreases slower than the rate $O(1/\sqrt{N_P})$, where N_P is the number of samples used to construct the filter. As no recursive estimation is required with the mean value filtering, it significantly improves the efficiency of PWoS.

5.2 Applications

5.2.1 Diffusion curves. Diffusion curves [Orzan et al. 2008] succinctly represent an image as a collection of curves with associated colors. The final image, exhibiting smooth color gradients, is recovered by solving a Laplace equation with the curves dictating boundary conditions. In our application, we solve the surface Laplace equation using PWoS. With our approach, the surface need not have a boundary curve conforming to the discretized mesh, which contrasts with the common approach [De Goes et al. 2022]. Fig. 7 shows the reconstruction of color at each point on the discretized surface, represented as a quadrilateral mesh and a combination of triangles, polylines, and point clouds. Our method naturally supports two-sided boundaries, with different colors specified on each side of a curve, and surface geometries with mixed-codimension. Additionally, the pointwise nature of PWoS allows it to be applied in a view-dependent manner. For example, given a camera configuration, for antialiasing, we sample points within each pixel to generate rays. We then generate PWoS samples at the ray-surface intersection points. No computational resources are wasted on surface points that are invisible to the camera (Fig. 1), and we can obtain clean results without relying on a fine discretization of the surface. Boundary integral-based approaches [Bang et al. 2023; Sun et al. 2012] would similarly allow domain discretization-free evaluation of diffusion curves, but they first require a global linear system solve. Moreover, such methods are not applicable to general curved surfaces, and would need to map the results in the 2D domain to the surface via UV coordinates, for example.

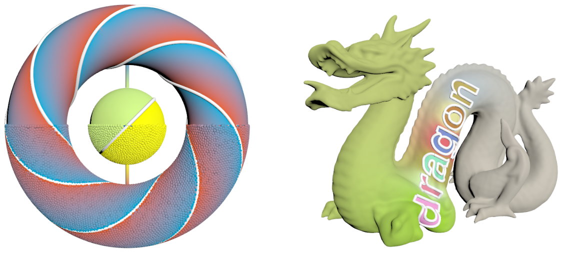


Fig. 7. Surface diffusion curves solved on various surface representations. The surface on the left is represented as a combination of triangles, polylines, and oriented points; the surface on the right is represented as a quadrilateral mesh. The scene on the left, featuring surface geometry of mixed codimension, was adapted from the work of King et al. [2023].

5.2.2 Geodesic distance. Crane et al. [2013] proposed the heat method, which solves two standard surface PDEs in series to compute the geodesic distance from the boundary C. The steps are summarized as

(1) $\Delta_{\mathcal{S}} u_{\mathcal{S}} - (1/t) u_{\mathcal{S}} = 0$ where $u_{\mathcal{S}} = 1$ on C,
(2) $\mathbf{X} = -(\nabla_{\mathcal{S}} u_{\mathcal{S}})/\|\nabla_{\mathcal{S}} u_{\mathcal{S}}\|_2$, and
(3) $\Delta_{\mathcal{S}} \phi_{\mathcal{S}} = \nabla_{\mathcal{S}} \cdot \mathbf{X}$ where $\phi_{\mathcal{S}} = 0$ on C,

where t is a small positive constant and ϕ is the geodesic distance. Step 2 uses the gradient of the solution to the screened Poisson equation found in Step 1. With a discretization-based method, a discrete gradient operator is used to estimate this gradient; in our method, we directly evaluate the gradient of u during Step 1 using the method in Section 3.3.2, without needing u itself. We evaluate

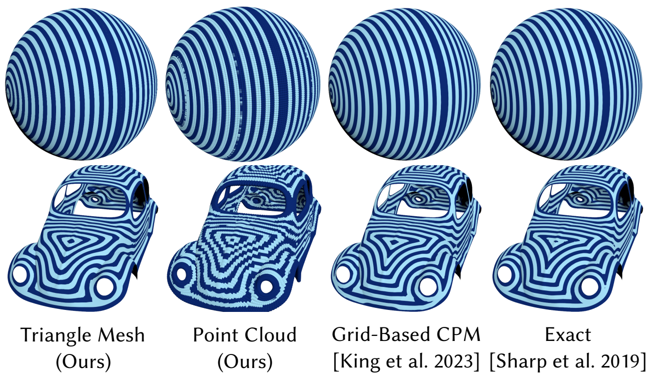


Fig. 8. Geodesic distance computation with the heat method. For each of the two scenes, we compare our algorithm on a polygonal mesh representation (leftmost) and oriented point cloud representation (middle-left) against a grid-based CPM counterpart [King et al. 2023] (middle-right) and the exact polyhedral distance computed with Geometry Central [Sharp et al. 2019] (rightmost). For the sphere surface (top), we compute the distance from the circle boundary curve in the center, and for the car surface (bottom), we compute the distance from the surface boundary edges. Note that the rendering of the point clouds assigns a UV coordinate per point, resulting in larger visual differences.

the gradient at mesh vertices and normalize it to get $\mathbf{X}$ at mesh vertices. In Step 3, again, we do not rely on a discrete divergence operator to solve the Poisson equation, but instead use the method described in Section 3.3.2. When our Poisson solver requires the evaluation of $\mathbf{X}$ at a point, we interpolate $\mathbf{X}$ from the mesh vertices and (re)normalize it. We can similarly compute the geodesic distance on a surface represented as a point cloud. Fig. 8 compares our PWoS-based heat method on surfaces represented as polygonal meshes or oriented point clouds against the heat method with grid-based CPM [King et al. 2023] and the exact geodesic distance computed with Geometry Central [Sharp et al. 2019]. Our results are consistent with the reference implementation, albeit with minor deviations.

5.2.3 Surface wave animation. Using our screened Poisson equation solver, we can solve some classes of time-dependent problems. We discretize the wave equation in time with implicit Euler to get a screened Poisson equation and solve it with time stepping (Fig. 9). At each time step, we store the solution at the vertices of the mesh and query the solution from previous frames by interpolating the values. In contrast to grid-based CPM [Auer et al. 2012], our method directly deals with surface geometry without defining an embedding grid.

5.3 Performance

The performance of our method depends on several factors; we report timings for two representative examples. We measured these timings using a workstation with two Intel(R) Xeon(R) Silver 4316 CPUs, each with 20 CPU cores. For the scene in Fig. 1 bottom left, the image resolution is 640 by 480 and the number of samples per pixel was 1024. The precomputation step, including medial axis computation, took less than 1 minute, and the rest of the main parts of PWoS took 2 hours and 11 minutes. We did not apply mean value filtering. For the scene in Fig. 7 left, we have 28,119 evaluation points. The medial axis point cloud extraction took 2.4 seconds, the initial solution estimate with 1 sample took 1.3 seconds, and the application of 10 mean value filtering steps with a filter constructed with 128 samples took 13 seconds. Optimizing the implementation with GPU acceleration may further improve performance.

6 CONCLUSION AND DISCUSSION

We have developed a Monte Carlo method for surface PDEs by augmenting the formulation of the walk on spheres method with a closest point projection step. Our algorithm is justified through its connection to the theory of closest point extensions drawn from the CPM literature. To accelerate its convergence, we have developed a practical mean value filtering method that utilizes a discrete basis defined over the surface. We have further analyzed the method's convergence on representative analytical tests and demonstrated its application to graphics problems.

PWoS currently supports only Dirichlet boundary conditions; efficient Neumann or Robin boundary handling similar to the walk on stars method for volumetric PDEs [Miller and Sawhney et al. 2024; Sawhney and Miller et al. 2023] would require the availability of a few more queries, such as a ray intersection query against the (extended) boundaries.

While we used a local feature estimation algorithm to allow walks with larger step sizes, the local feature size estimation itself imposes additional smoothness assumptions on the surface. To respect small-scale local features, the walk can require many iterations to reach a Dirichlet boundary. This effect is partly due to our algorithm (like WoS) being based on an integral equation that holds only locally inside a ball. Revisiting this choice using an integral equation based on a global relationship, such as the one underpinning the walk on *boundary* method [Sugimoto et al. 2023], could lead to a more efficient alternative for surface PDEs.

Lastly, our method relied on the assumption that the closest point extension compensation term (i.e., $g(\mathbf{x})$ in Eq. 6) in the embedding PDE is negligible. We empirically showed that the algorithm designed with this assumption works well when the source term has a relatively simple expression, but we do not yet have a complete understanding of when this assumption is strictly valid. However, since $g(\mathbf{x})$ tends to zero continuously as $\mathbf{x}$ approaches the surface, the influence of ignoring this term is expected to decrease as we shrink the embedding space (i.e., shrink the sphere size). One can always take a smaller sphere size, albeit at a higher computational cost, as we show in Fig. 10. Extending our method to consider the effect of the compensation term would further improve the reliability and broaden the applicability of our method.

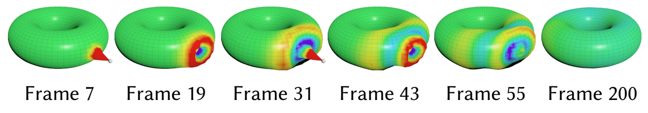


Fig. 9. Surface wave animation. We solve the wave equation on the surface and visualize the solution as a displacement applied on the surface in the normal directions. Combined with a time-stepping approach, our method can be applied to a few time-dependent problems. We set the solution to one point on the mesh as the boundary condition for the first 49 frames, and remove the boundary from the 50th frame. The waves damp out as the simulation continues, as expected.

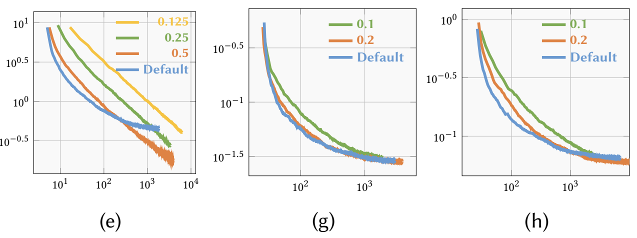


Fig. 10. Using bounded sphere size. For the Poisson and screened Poisson problems (e), (g), and (h) in Fig. 4, we compare the Default option of not constraining the sphere size (apart from the limit imposed by the local feature size estimate) against specified limits on the maximum sphere size as indicated in the legend. The vertical axis shows the root mean squared error, and the horizontal axis shows the time in seconds measured on a MacBook Pro with an M1 Pro chip. For (e), we had 1024 evaluation points, and for (g) and (h), we used 100 sample evaluation points. While limiting the sphere size may reduce the bias, as we can observe from the intersections of the curves for the default option and the curves for the sphere-size-constrained option, the computation may take longer, and it is difficult to get a practical advantage.

ACKNOWLEDGMENTS

This research was partially funded by NSERC Discovery Grants (RGPIN-2021-02524 & RGPIN-2020-03918), CFI-JELF (Grant 40132), and a grant from Autodesk. The first author was partially funded by the David R. Cheriton Graduate Scholarship. The second author was partially funded by the Ontario Graduate Scholarship.

REFERENCES

N. Amenta and M. Bern. 1999. Surface Reconstruction by Voronoi Filtering. *Discrete & Computational Geometry* 22, 4 (1999), 481–504. https://doi.org/10.1007/PL00009475

S. Auer, C. B. Macdonald, M. Treib, J. Schneider, and R. Westermann. 2012. Real-Time Fluid Effects on Surfaces using the Closest Point Method. *Computer Graphics Forum* 31, 6 (2012), 1909–1923. https://doi.org/10.1111/j.1467-8659.2012.03071.x

Ghada Bakbouk and Pieter Peers. 2023. Mean Value Caching for Walk on Spheres. In *Eurographics Symposium on Rendering*, Tobias Ritschel and Andrea Weidlich (Eds.). The Eurographics Association, Delft, Netherlands, 10 pages. https://doi.org/10.2312/sr.20231120

Seungbae Bang, Kirill Serkh, Oded Stein, and Alec Jacobson. 2023. An Adaptive Fast-Multipole-Accelerated Hybrid Boundary Integral Equation Method for Accurate Diffusion Curves. *ACM Trans. Graph.* 42, 6, Article 215 (dec 2023), 28 pages. https://doi.org/10.1145/3618374

Alec Bartsch, Colin Thompson, and Fernando de Goes. 2023. A Procedural Approach for Stylized Bark Shading. In *ACM SIGGRAPH 2023 Talks* (<conf-loc>, <city>Los Angeles</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 10, 2 pages. https://doi.org/10.1145/3587421.3595419

Mikhail Belkin, Jian Sun, and Yusu Wang. 2009. Constructing Laplace operator from point clouds in $\mathbb{R}^d$. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms* (New York, New York) *(SODA '09)*. Society for Industrial and Applied Mathematics, USA, 1031–1040. https://doi.org/10.1137/1.9781611973068.112

A. Bunge and M. Botsch. 2023. A Survey on Discrete Laplacians for General Polygonal Meshes. *Computer Graphics Forum* 42, 2 (2023), 521–544. https://doi.org/10.1111/cgf.14777

Yujia Chen and Colin B. Macdonald. 2015. The Closest Point Method and Multigrid Solvers for Elliptic Equations on Surfaces. *SIAM Journal on Scientific Computing* 37, 1 (2015), A134–A155. https://doi.org/10.1137/130929497

Ka Chun Cheung, Leevan Ling, and Steven J. Ruuth. 2015. A localized meshless method for diffusion on folded surfaces. *J. Comput. Phys.* 297 (2015), 194–206. https://doi.org/10.1016/j.jcp.2015.05.021

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.* 32, 5, Article 152 (oct 2013), 11 pages. https://doi.org/10.1145/2516971.2516977

Fernando De Goes, William Sheffler, and Kurt Fleischer. 2022. Character articulation through profile curves. *ACM Trans. Graph.* 41, 4, Article 139 (jul 2022), 14 pages. https://doi.org/10.1145/3528223.3530060

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 317–324. https://doi.org/10.1145/311535.311576

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graphical Models and Image Processing* 58, 5 (1996), 471–483. https://doi.org/10.1006/gmip.1996.0039

Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. 2009. The scale axis transform. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry* (Aarhus, Denmark) *(SCG '09)*. Association for Computing Machinery, New York, NY, USA, 106–115. https://doi.org/10.1145/1542362.1542388

Nathan King, Haozhe Su, Mridul Aanjaneya, Steven Ruuth, and Christopher Batty. 2023. A Closest Point Method for Surface PDEs with Interior Boundary Conditions for Geometry Processing. arXiv:2305.04711 [cs.GR]

Jaehwan Ma, Sang Won Bae, and Sunghee Choi. 2012. 3D medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer* 28, 1 (2012), 7–19. https://doi.org/10.1007/s00371-011-0594-7

Richard Henri MacNeal. 1949. *The solution of partial differential equations by means of electrical networks.* Ph. D. Dissertation. California Institute of Technology. https://doi.org/10.7907/PZ04-5290

Thomas März and Colin B. Macdonald. 2012. Calculus on Surfaces with General Closest Point Functions. *SIAM J. Numer. Anal.* 50, 6 (2012), 3303–3328. https://doi.org/10.1137/120865537

Balint Miklos, Joachim Giesen, and Mark Pauly. 2010. Discrete scale axis representations for 3D geometry. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 101, 10 pages. https://doi.org/10.1145/1833349.1778838

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4, Article 82 (jul 2023), 11 pages. https://doi.org/10.1145/3592400

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Trans. Graph.* 43, 4, Article 41 (jul 2024), 18 pages. https://doi.org/10.1145/3658153

Mervin E. Muller. 1956. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *The Annals of Mathematical Statistics* 27, 3 (1956), 569 – 589. https://doi.org/10.1214/aoms/1177728169

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 92, 8 pages. https://doi.org/10.1145/1399504.1360691

A. Petras and S.J. Ruuth. 2016. PDEs on moving surfaces via the closest point method and a modified grid based particle method. *J. Comput. Phys.* 312 (2016), 139–156. https://doi.org/10.1016/j.jcp.2016.02.024

Cécile Piret. 2012. The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *J. Comput. Phys.* 231, 14 (2012), 4662–4675. https://doi.org/10.1016/j.jcp.2012.03.007

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

Steven J. Ruuth and Barry Merriman. 2008. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.* 227, 3 (jan 2008), 1943–1961. https://doi.org/10.1016/j.jcp.2007.10.009

Karl K. Sabelfeld and Nikolai A. Simonov. 1994. *Random Walks on Boundary for Solving PDEs.* De Gruyter, Berlin. https://doi.org/10.1515/9783110942026

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, 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 (aug 2023), 22 pages. https://doi.org/10.1145/3592398

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

Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum* 39, 5 (2020), 69–80. https://doi.org/10.1111/cgf.14069

Nicholas Sharp, Keenan Crane, et al. 2019. *GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing.* https://geometry-central.net/

Side Effects Software, Inc. 2023. *Houdini.* https://www.sidefx.com/products/houdini/ Computer Software.

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. https://doi.org/10.1145/311535.311548

Jos Stam. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.* 22, 3 (jul 2003), 724–731. https://doi.org/10.1145/882262.882338

Ryusuke Sugimoto, Christopher Batty, and Toshiya Hachisuka. 2024. Velocity-Based Monte Carlo Fluids. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 8, 11 pages. https://doi.org/10.1145/3641519.3657405

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, Article 81 (jul 2023), 16 pages. https://doi.org/10.1145/3592109

Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. 2012. Diffusion curve textures for resolution independent texture mapping. *ACM Trans. Graph.* 31, 4, Article 74 (jul 2012), 9 pages. https://doi.org/10.1145/2185520.2185570

Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum* 35, 2 (2016), 573–597. https://doi.org/10.1111/cgf.12865

Greg Turk. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. *SIGGRAPH Comput. Graph.* 25, 4 (jul 1991), 289–298. https://doi.org/10.1145/127719.122749

Ingrid von Glehn, Thomas März, and Colin B. Macdonald. 2013. An embedded method-of-lines approach to solving partial differential equations on surfaces. arXiv:1307.5657 [math.NA]