Geometric Queries on Closed Implicit Surfaces for Walk on Stars

Tianyu Huang
tianyu@illumiart.net
Independent researcher
Beijing, China
Tsinghua University
Beijing, China

Abstract

Walk on stars (WoSt) is currently one of the most advanced Monte Carlo solvers for PDEs. Unfortunately, the lack of reliable geometric query approaches has hindered its applicability to boundaries defined by implicit surfaces. This work proposes a geometric query framework over closed implicit surfaces for WoSt, under the scope of *walkin' Robin* [Miller et al. 2024b]. Our key observation is that all WoSt queries can be formulated as constrained global optimization or constraint satisfaction problems. Based on our formulations, to solve the highly non-convex problems, we adopt a branch-and-bound approach based on interval analysis. To the best of our knowledge, our method is the first to study closest silhouette point queries and Robin radius bound queries on closed implicit surfaces. Our formulations and methods first enable mesh-free PDE solving via WoSt when boundaries are defined by closed implicit surfaces.

CCS Concepts

• **Computing methodologies** → **Shape analysis**.

ACM Reference Format:
Tianyu Huang. 2025. Geometric Queries on Closed Implicit Surfaces for Walk on Stars. In *SIGGRAPH Asia 2025 Technical Communications (SA Technical Communications '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3757376.3771378

1 Introduction

Partial differential equations (PDEs) are fundamental in modeling physical phenomena and widely used in graphics. Recently, Monte Carlo solvers for PDEs have gained increasing attention, with *walk on stars (WoSt)* [Miller et al. 2024b; Sawhney et al. 2023], an extension of *walk on spheres (WoS)* [Muller 1956], emerging as a promising approach. Unlike traditional methods, WoSt avoids discretization and thus bypasses the robustness and performance challenges of volumetric meshing, making it well-suited for increasingly complex geometries in graphics.

Despite these advantages, WoSt has not yet been demonstrated to operate entirely in domains defined by implicit surfaces (hereafter referred to as *implicit domains*). Implicit surfaces can represent intricate geometric details, support robust Boolean/blending operations, and are central to many modern scene representations. A volumetric PDE solver supporting implicit domains directly would inherit these benefits, but current WoSt implementations remain tied to boundary meshes or only support Dirichlet boundaries defined by implicit surfaces [Miller et al. 2024a], limiting its strengths in implicit domains. The difficulty in extending WoSt to implicit domains is that, while WoS only requires closest point queries, WoSt additionally requires geometric queries that have not been well studied on implicit surfaces. To facilitate explanation, we list all the geometric queries required by WoSt:

(1) Closest point query (CPQ).
(2) Ray intersection query.
(3) Closest silhouette point query (CSPQ).
(4) Robin radius bound query (RRBQ).
(5) Point sampling on *reflecting* (Neumann/Robin) boundaries.

While queries (1), (2), and (5) are well studied, no existing methods address (3) and (4) on implicit surfaces. For instance, silhouette detection has been studied in non-photorealistic [Bremer and Hughes 1998] and differentiable [Zhou et al. 2024] rendering, but methods for CSPQ remain unexplored on implicit surfaces; and RRBQ has only been studied for boundary meshes [Miller et al. 2024b]. Consequently, WoSt still falls back to intermediate boundary meshing to handle implicit domains, undermining its mesh-free appeal and degrading geometric fidelity.

In response, we propose a geometric query solution over closed (compact and boundary-free) implicit surfaces for WoSt, corresponding to PDE problems where the Dirichlet and reflecting boundaries

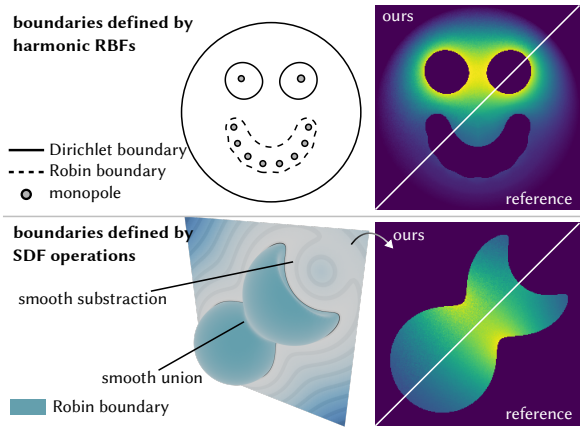

Figure 1: Walk on stars (WoSt) with boundaries defined by closed implicit surfaces. Our method can perform complete WoSt geometric queries on various closed implicit representations (here we demonstrate harmonic RBFs [Miller et al. 2024a] and SDF operations). WoSt based on our method yields correct solutions to the Laplace Dirichlet-Robin problems.

SA Technical Communications '25, Hong Kong, Hong Kong

are defined by closed implicit surfaces. Our approach builds on an observation: all the queries above can be formulated as constrained global optimization or constraint satisfaction problems. Based on our proposed formulations, to run optimization and satisfaction on the non-convex function landscape, we adopt branch-and-bound techniques based on interval analysis (IA) [Snyder 1992]. From a contributions perspective, we are the first to study the formulation and implementation of queries (3) and (4) on closed implicit surfaces, and the first to realize a complete WoSt when boundaries are defined by closed implicit surfaces.

2 Preliminary: Interval Analysis (IA) for Constrained Global Optimization and Constraint Satisfaction

The geometric queries exhibit non-convex landscapes, and WoSt often admits conservative queries. Consequently, we adopt IA as the tool for our queries. We recall two relevant methods introduced by Snyder [1992]: `MINIMIZE` for constrained global optimization, and `SOLVE` for constraint satisfaction. In the methods, *boxes*, *i.e.*, Cartesian product of intervals, are used to divide the space. For an *objective function* g, its corresponding *objective inclusion function* $\square g$ maps a box Y to an interval enclosing all $g(x), \forall x \in Y$. *Constraint inclusion function* $\square G$ that takes a box Y returns a three-value logic:

- `positive`: constraint G holds for all points in Y,
- `negative`: constraint G fails for all points in Y,
- `unknown`: both satisfying and unsatisfying points exist in Y.

Solution acceptance set constraint $\square A$ returns `true` if the search box is fine enough (that meets the specified tolerance). `X` is the initial box. The algorithms are as follows:

Listing 1: `MINIMIZE`: Constrained Global Optimization

```
def MINIMIZE( □g, □G, □A, X ):
    L ← PriorityQueue({X})
    UB ← +∞ # upper bound for pruning
    results ← ∅ # multiple solutions may arise
    while L ≠ ∅:
        Y ← L.pop_min_lower_bound() # best candidate
        if □A(Y) = true: # stricter tests may be applied
            results ← results ∪ {Y} # accept Y
        else:
            for Y_i in subdivide(Y):
                if □G(Y_i) = negative:
                    continue # prune constraint-violating boxes
                [a_i, b_i] ← □g(Y_i)
                if a_i > UB:
                    continue # prune suboptimal boxes
                L.insert(Y_i, priority=a_i)
                UB ← min(UB, b_i) # update upper bound
    return results
```

Listing 2: `SOLVE`: Constraint Satisfaction

```
def SOLVE( □G, □A, X ):
    L ← {X}, results ← ∅
    while L ≠ ∅:
        Y ← remove(L) # take a box from L
        if □G(Y) = positive ∧ □A(Y) = true:
            results ← results ∪ {Y} # accept fine &
                constraint-satisfying boxes
        elif □G(Y) = unknown:
            L ← L ∪ subdivide(Y) # further subdivide unknown
                boxes
    return results
```

In our paper, `SOLVE` is used for point sampling on reflecting boundaries (section 3.4), while the rest utilizes `MINIMIZE`.

3 Method

We consider a domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ whose boundary[1] is partitioned into two closed implicit surfaces: the Dirichlet boundary $\partial\Omega_{\mathrm{D}}$ and the reflecting boundary $\partial\Omega_{\mathrm{R}}$. They are defined as the zero level sets of smooth implicit functions $f_{\mathrm{D}}, f_{\mathrm{R}} : \mathbb{R}^d \to \mathbb{R}$, with the regularity condition $\nabla f_{\mathrm{D}}(x) \neq 0, \forall x \in \partial\Omega_{\mathrm{D}}$ and $\nabla f_{\mathrm{R}}(z) \neq 0, \forall z \in \partial\Omega_{\mathrm{R}}$. Under this condition, ∇f_{D} and ∇f_{R} are parallel to the corresponding surface normals. No Lipschitz assumption is required. We denote by $x \in \Omega$ a query point, by $y \in \partial\Omega_{\mathrm{D}}$ a Dirichlet boundary point, and by $z \in \partial\Omega_{\mathrm{R}}$ a reflecting boundary point. The closed and open balls centered at x with radius R are written as $B(x, R)$ and $\mathring{B}(x, R) = B(x, R) \setminus \partial B(x, R)$, *resp.* Some notations here differ from those in the WoSt paper for contextual consistency; readers unfamiliar with WoSt may refer to the supplementary material.

3.1 Closest Point Query (CPQ) & Ray Intersection Query

In WoSt, CPQ is used on $\partial\Omega_{\mathrm{D}}$ and ray intersection query is used on $\partial\Omega_{\mathrm{R}}$, these queries are well-studied as global optimization problems; for completeness, we briefly summarize them here. The CPQ can be formulated as:

$$y_{\mathrm{D}} = \arg\min_{y \in \mathbb{R}^d} \|y - x\|^2 \quad \text{subject to} \quad f_{\mathrm{D}}(y) = 0, \tag{1}$$

and ray intersection query can be formulated as:

$$z_{\mathrm{I}} = \arg\min_{z \in \mathbb{R}^d} \|z - x\|^2 \quad \text{subject to } f_{\mathrm{R}}(z) = 0,\ z \in \{x + tv \mid t > 0\}, \tag{2}$$

where $v \in \mathbb{R}^d$ is a unit direction vector. Apart from using `MINIMIZE`, many efficient methods exist, which we do not discuss further here. CPQ yields the *Dirichlet distance* $R_{\mathrm{D}} = \|y_{\mathrm{D}} - x\|$.

3.2 Closest Silhouette Point Query (CSPQ)

A silhouette point is a point whose normal is perpendicular to the viewing direction. In WoSt, CSPQ is used on $\partial\Omega_{\mathrm{R}}$ to produce the *silhouette bound* R_{S} for the radius of the star-shaped region such that no silhouette exists within the star-shaped region. We propose to formulate CSPQ as the following problem when $x \notin \partial\Omega_{\mathrm{R}}$:

$$z_{\mathrm{S}} = \arg\min_{z \in \mathbb{R}^d} \|z - x\|^2 \quad \text{subject to} \quad \begin{cases} f_{\mathrm{R}}(z) = 0 \\ \nabla f_{\mathrm{R}}(z) \cdot (z - x) = 0, \end{cases} \tag{3}$$

therefore we have $R_{\mathrm{S}} = \|z_{\mathrm{S}} - x\|$. In WoSt, R_{S} should satisfy $R_{\mathrm{S}} \leq R_{\mathrm{D}}$, thus we introduce the additional constraint $\|z - x\| \leq R_{\mathrm{D}}$. To narrow the search space, we initialize the `MINIMIZE` algorithm with a box centered at x and of side length $2R_{\mathrm{D}}$. When the problem involves Robin boundaries, we slightly shrink R_{S} to ensure $\cos\theta(z) = \frac{n_z \cdot (z-x)}{r} = \frac{|\nabla f_{\mathrm{R}}(z) \cdot (z-x)|}{\|\nabla f_{\mathrm{R}}(z)\| r} > 0, \forall z \in B(x, R_{\mathrm{S}}) \cap \partial\Omega_{\mathrm{R}}$[2], thereby avoiding singularities in *reflectance* ρ_μ [Miller et al. 2024b, eqs. 9 and 33]. The query strategy for $x \in \partial\Omega_{\mathrm{R}}$ is discussed in the supplementary material.

[1] In this section, "boundary" refers to the boundary of the PDE problems and should be distinguished from the boundaries of the implicit surfaces discussed in the context of open and closed surfaces.

[2] ∇f_{R} always points outward from the implicit surface, irrespective of x's location (inside or outside); thus we take the absolute value in the dot product.

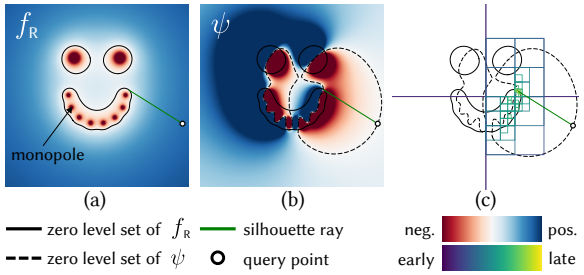

Figure 2: Closest silhouette point query (CSPQ). Figures (a) and (b) illustrate the landscapes of f_R and $\psi = \nabla f_R(z) \cdot (z - x)$, *resp.* Figure (c) visualizes the minimization process.

3.3 Robin Radius Bound Query (RRBQ)

In walkin' Robin, to ensure $\rho_\mu \in [0, 1]$ for a stable Monte Carlo estimator, Miller et al. [2024b] propose shrinking R_S to the *Robin radius bound* $R_R \le R_S$. We now discuss the query for R_R by dimension.

3.3.1 2D Case. In 2D, a desired R_R ensures, $\forall z \in B(x, R_R) \cap \partial\Omega_R \subset B(x, R_S) \cap \partial\Omega_R$, the following holds [Miller et al. 2024b, eq. 35]:

$$R_R \le r \exp\left(\frac{\cos\theta}{\mu(z)\, r}\right), \tag{4}$$

where $r = r(z) = \|z - x\|$, $\cos\theta > 0$ (section 3.2), and $\mu(z) > 0$ is the prescribed Robin coefficient defined on $\partial\Omega_R$. Since z appears entirely on the right-hand side of eq. (4), it is natural to consider taking the minimum of it over $B(x, R_S) \cap \partial\Omega_R$ as the upper bound for R_R, which corresponds to the following optimization problem:

$$R_R^* = \min_{z \in \mathbb{R}^2} r \exp\left(\frac{\cos\theta}{\mu(z)\, r}\right) \quad \text{subject to} \quad \begin{cases} f_R(z) = 0 \\ r \le R_S. \end{cases} \tag{5}$$

We further find that the bound R_R^* here is tight, *i.e.*, there exists no $\tilde{R}_R > R_R^*$ satisfying eq. (4), because $\forall z \in B(x, R_S) \cap \partial\Omega_R$ we have:

$$r \exp\left(\frac{\cos\theta}{\mu(z)\, r}\right) - r = r\left[\exp\left(\frac{\cos\theta}{\mu(z)\, r}\right) - 1\right] > 0. \tag{6}$$

Therefore, the minimizer z^* necessarily satisfies $r^*(z^*) < R_R^*$, or equivalently $z^* \in \mathring{B}(x, R_R^*) \cap \partial\Omega_R$, so $\tilde{R}_R > R_R^*$ would necessarily violate eq. (4) at z^*. Hence, R_R^* is guaranteed to be the tight bound.

Finally, we take the lower bound of the interval for R_R^* for numerical stability. This also applies to the 3D case.

3.3.2 3D Case. In 3D, eq. (4) takes a different form [Miller et al. 2024b, eq. 11]:

$$R_R \le \frac{r}{1 - \frac{\cos\theta}{\mu(z) r}} \quad \text{when} \quad r > \frac{\cos\theta}{\mu(z)}. \tag{7}$$

The definition of symbols follow the 2D case except that the points and vectors are defined in $\mathbb{R}^3$. The biggest difference lies in the condition $r > \frac{\cos\theta}{\mu(z)}$. When $0 < r \le \frac{\cos\theta}{\mu(z)}$, R_R is no longer subject to eq. (7), and it can be made arbitrarily large [Miller et al. 2024b, section 4.2.1]. We also observe that eq. (7) naturally guarantees:

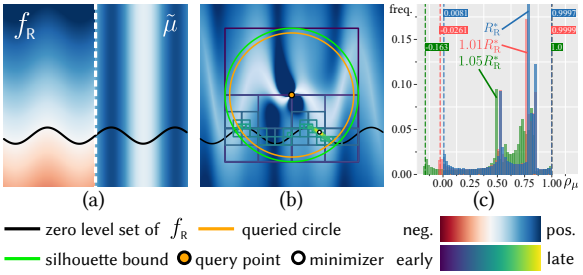

Figure 3: Robin radius bound query (RRBQ). Figure (a) shows the landscape of f_R and $\tilde{\mu}$ (section 3.3.3). Figure (b) shows the landscape of the objective, optimization process and result. In figure (c), we randomly sample points on $\partial\Omega_R$ inside the queried circle, and plot the distribution of ρ_μ [Miller et al. 2024b, eqs. 9 and 33]. As illustrated, $\rho_\mu \in [0, 1]$ when the radius equals to R_R^*, while increasing the radius slightly yields invalid ranges of ρ_μ.

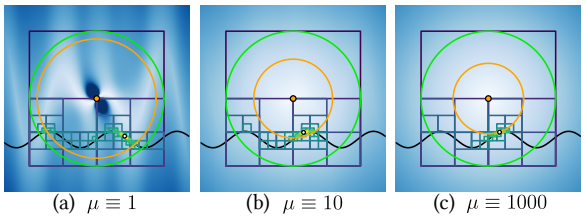

Figure 4: Sanity check on the effect of increasing μ on the radius. We set $\tilde{\mu}(z)$ to a constant *w.r.t.* z, increase it from (a) to (c), and perform RRBQ. The results show the query converges to CPQ on $\partial\Omega_R$ as μ increases, which agrees with the results reported by Miller et al. [2024b, fig. 7].

$$\frac{r}{1 - \frac{\cos\theta}{\mu(z) r}} - r = \frac{\frac{\cos\theta}{\mu(z)}}{1 - \frac{\cos\theta}{\mu(z) r}} > 0 \quad \text{when} \quad r > \frac{\cos\theta}{\mu(z)}. \tag{8}$$

Thus, the following optimization problem also yields a tight bound:

$$R_R^* = \min_{z \in \mathbb{R}^3} \frac{r}{1 - \frac{\cos\theta}{\mu(z)\, r}} \quad \text{subject to} \quad \begin{cases} f_R(z) = 0 \\ \frac{\cos\theta}{\mu(z)} < r \le R_S. \end{cases} \tag{9}$$

We denote the objective function in eq. (9) as $O(z; x)$. Using IA to impose $r > {\cos\theta}/{\mu(z)}$ may be overkill, as in condition-violating regions, we no longer need to evaluate its range. We instead adopt a simpler approach: based on the observation that $O(z; x) \to +\infty$ as $r \to ({\cos\theta}/{\mu(z)})^+$, we set a new optimization objective:

$$\tilde{O}(z; x) = \begin{cases} \frac{r}{1 - \frac{\cos\theta}{\mu(z)\, r}}, & r > \frac{\cos\theta}{\mu(z)} \\ +\infty, & o.w. \end{cases} \tag{10}$$

Since we are performing a minimization, the optimizer will automatically bypass the $+\infty$ region. This objective can be computed numerically by clamping the interval of $1 - \frac{\cos\theta(z)}{\mu(z)\, r}$ to $[0, +\infty)$ (we define division by 0 to yield $+\infty$). In the end we have:

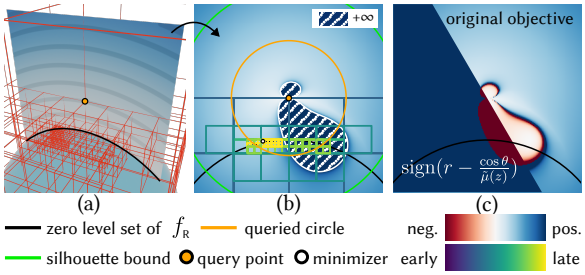

Figure 5: RRBQ in 3D. Figures (a) and (b) show the optimization process and results in $\mathbb{R}^3$ and on a slice, *resp.* Comparing (b) and (c), we see that regions where $r > \frac{\cos\theta}{\tilde{\mu}(z)}$ is not satisfied are set to $+\infty$ in eq. (10), then the optimizer skips these regions during the coarse stage. The sampled values (see fig. 3 caption) of ρ_μ fall within $[0.4891, 0.99996]$.

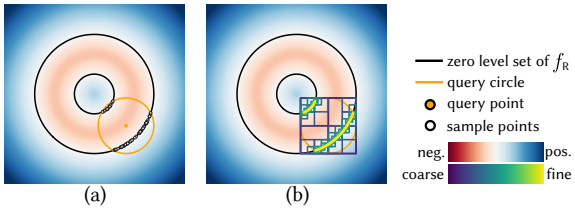

Figure 6: Point sampling on reflecting boundaries. Figure (a) shows the implicit function landscape, the query circle, and the sampled points. Figure (b) illustrates the search boxes.

$$R_R^* = \min_{z\in\mathbb{R}^3} \tilde{O}(z;x) \quad \text{subject to} \quad \begin{cases} f_R(z) = 0 \\ r \le R_S. \end{cases} \tag{11}$$

3.3.3 Discussion & Implementation. By definition, the domain of μ is $\partial\Omega_R$, whereas the IA operates in $\mathbb{R}^d$. Therefore, we extend μ to $\mathbb{R}^d$ by constructing $\tilde{\mu} : \mathbb{R}^d \to \mathbb{R}$ that satisfies $\tilde{\mu}(z) = \mu(z), \forall z \in \partial\Omega_R$. We initialize the `MINIMIZE` algorithm with a box centered at x with side length $2R_S$.

3.4 Point Sampling on Reflecting Boundaries

In WoSt, we need to sample points on $\Gamma = B(x, R_R) \cap \partial\Omega_R$ [Sawhney et al. 2023, section 5.2], which can be formalized as the following constraint satisfaction problem:

$$\Gamma := \{z \in \mathbb{R}^d \mid f_R(z) = 0,\ \|z - x\| \le R_R\}. \tag{12}$$

We adopt the `SOLVE` algorithm. The initial search domain is the box centered at x and of side length $2R_R$. In practice, *reservoir sampling* or a cached tree structure can be used to improve performance.

4 Results and Discussion

We implement the method in Julia [Bezanson et al. 2017] using the `IntervalArithmetic.jl` [Sanders and Benet 2014] library, extending it with forward-mode automatic differentiation via dual intervals to obtain gradients. We set the tolerance in section 2 to $1/10$ of the ϵ-shell. For each newly introduced query, we conduct verification experiments in figs. 2 to 6. Based on our method, we also use WoSt to solve Laplace equations, as shown in fig. 1. The results match the expected behavior. We provide detailed setup in the supplementary document.

5 Conclusion, Limitations and Future Work

By proposing formulations and adopting IA-based global optimization and constraint satisfaction methods, our present method first enables all necessary geometric queries on closed implicit surfaces for WoSt, enabling mesh-free PDE solving via WoSt in implicit domains. Our work comes with limitations: although our method is, in principle, applicable to neural SDFs [Sharp and Jacobson 2022], we have not implemented this extension; our current boundary sampling strategy is relatively inefficient—recent advances [Ling et al. 2025] may be combined to improve performance; our work primarily serves as a proof-of-concept, leaving high-performance implementations for future work. At present, optimizing implicit surfaces with reflecting boundaries remains an open inverse problem in the Monte Carlo for PDEs community, and we believe our method could serve as a tool toward this goal.

Acknowledgments

Tianyu Huang would like to thank Hao Pan, and Ryusuke Sugimoto for valuable discussions. Jingwang Ling helped with proofreading of this paper. This paper utilized resources from Feng Xu's lab. Tianyu Huang also acknowledges the travel funding support from the Tsinghua University *Spark* Program.

References

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A fresh approach to numerical computing. *SIAM Review* 59, 1 (2017), 65–98. doi:10.1137/141000671

David Bremer and John F. Hughes. 1998. Rapid Approximate Silhouette Rendering of Implicit Surfaces. In *Proceesings of Implicit Surfaces 98.*

Selena Ling, Abhishek Madan, Nicholas Sharp, and Alec Jacobson. 2025. Uniform Sampling of Surfaces by Casting Rays. 44, 5 (2025).

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024a. Differential Walk on Spheres. *ACM Trans. Graph.* 43, 6, Article 174 (Nov. 2024), 18 pages. doi:10.1145/3687913

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024b. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Trans. Graph.* 43, 4 (2024).

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

David P. Sanders and Luis Benet. 2014. *IntervalArithmetic.jl.* doi:10.5281/zenodo.3336308

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 (2023).

Nicholas Sharp and Alec Jacobson. 2022. Spelunking the deep: guaranteed queries on general neural implicit surfaces via range analysis. *ACM Trans. Graph.* 41, 4, Article 107 (July 2022), 16 pages. doi:10.1145/3528223.3530155

John M. Snyder. 1992. Interval analysis for computer graphics. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 121–130. doi:10.1145/142920.134024

Siwei Zhou, Youngha Chang, Nobuhiko Mukai, Hiroaki Santo, Fumio Okura, Yasuyuki Matsushita, and Shuang Zhao. 2024. Path-Space Differentiable Rendering of Implicit Surfaces. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24).* Association for Computing Machinery, New York, NY, USA, Article 19, 11 pages. doi:10.1145/3641519.3657473

Supplementary Document for Geometric Queries on Closed Implicit Surfaces for Walk on Stars

Tianyu Huang
tianyu@illumiart.net
Independent Researcher
Beijing, China
Tsinghua University
Beijing, China

A Notes on the Mathematical Notation in Walk on Stars (WoSt)

For contextual consistency, we use specific symbols and names for certain WoSt concepts. To help readers unfamiliar with WoSt relate this work to the WoSt paper, the following table maps our notation to that of the walkin' Robin paper [Miller et al. 2024]:

Table 1: Mapping of notation between our work and Miller et al. [2024]

Our work	Miller et al. [2024]
Dirichlet distance R_{D}	distance to the closest point $R_\infty = d_{\mathrm{Dirichlet}}(x)$
silhouette bound R_{S}	distance to the closest silhouette point $R_0 = d_{\mathrm{silhouette}}(x)$
Robin radius bound R_{R}	R_μ, referred to as R in Miller et al. [2024, section 4.2]

B Notes on R_{S} Shrinking in CSPQ

In CSPQ (section 3.2), we propose slightly shrinking R_{S} to avoid singularities in ρ_μ, primarily due to the use of the ρ_μ notation from Miller et al. [2024, eqs. 7, 9 and 33], namely:

$$\rho_\mu(x,z) = 1 - \mu(z)\frac{G^{\mathrm{B}}(x,z)}{P^{\mathrm{B}}(x,z)}, \tag{1}$$

where $P^{\mathrm{B}}(x,z)$, the Poisson kernel, is proportional to the projected solid angle; that is, when $\cos\theta(z) = 0$, $P^{\mathrm{B}}(x,z) = 0$, resulting in a singularity. We refer to Sawhney et al. [2023, appendix A] for detailed expression of P^{B} and the Green's function G^{B}. In the original walkin' Robin work [Miller et al. 2024], this issue is not discussed because they operate on meshes, which have discretely varying normals that automatically ensure $\cos\theta > 0$ within $B(x, R_{\mathrm{S}})$; while in our work, we consider smooth implicit surfaces, so silhouette points with $\cos\theta = 0$ are included in $B(x, R_{\mathrm{S}})$.

However, we note that our shrinking operation is primarily for the convenience of subsequent discussion in the main text. Theoretically, even without shrinking, the stability of the sampler is unaffected (though efficiency may be impacted). This is mainly because, in the walkin' Robin estimator [Miller et al. 2024, eq. 8], the recursive term performs *importance sampling over the Poisson kernel*, so silhouettes with $\cos\theta = 0$ have zero probability of being sampled. Even without using the walkin' Robin's default sampler, for a good sampler that attempts to satisfy the optimal sampling probability [Miller et al. 2024, eq. 8]:

$$p \propto \frac{\rho_\mu(x_k, x_{k+1})P^B(x_k, x_{k+1})u(x_{k+1})}{\alpha(x_k)}, \tag{2}$$

we observe that there is also a P^B factor in the numerator, which cancels the singularity. Furthermore, since $G^B = 0$ on ∂B, the contribution and ideal sampling probability at the silhouette points are also zero.

C CSPQ on $\partial\Omega_{\mathrm{R}}$

When $x \in \partial\Omega_{\mathrm{R}}$, the optimization problem in the main text would always yield $z_{\mathrm{S}} = x$, and thus $R_{\mathrm{S}} = 0$. This outcome is *acceptable* because, in WoSt, the radius is clamped to a small ϵ, and the walk is still able to proceed. In fact, in walkin' Robin this is a standard strategy to handle cases where $x \in \partial\Omega_{\mathrm{R}}$ but $\mu(x) \neq 0$ [Sawhney and Miller 2023]. There are ways to address this issue, but it is worth noting that simply discarding the current point is not feasible: when the local geometry of $\partial\Omega_{\mathrm{R}}$ at x is concave, the silhouette may indeed lie exactly at x. Fortunately, determining this geometric property is not difficult: in systems with second-order derivatives, the Hessian can be used, and otherwise one may resort to finite-difference methods. Inspired by common tricks in ray tracing, one may also consider applying a slight offset along the normal direction to query points that lie on the boundary.

In our implementation, we follow the principle of *Occam's razor* and apply no special handling, except for an ϵ-clamp. A more robust solution will be presented in our future full work.

D Discussion of the Case $\mu(z) = 0$

In section 3.3, we assume $\mu(z) > 0$, mainly to avoid singularities in eqs. (5) and (7) in the main text. However, when $\mu(z) = 0$, we have $\rho_\mu = 1 \in [0,1], \forall z$ [Miller et al. 2024, eqs. 9 and 33]. Thus, similar to the treatment in eq. (10) in the main text, we may regard R_{R} as arbitrarily large at points z where $\mu(z) = 0$.

E Experimental Setup

E.1 Forward Automatic Differentiation (AD) Implementation

For the proof-of-concept experiments, we did not implement a sophisticated AD system; instead, we used a simple forward-mode AD system based on dual intervals. We employed a `IntervalDual` structure to store both an interval and its gradient:

```
struct IntervalDual <: Real
    val::Interval
    der::Vector{Interval} # "dual" gradient
end
```

For each mathematical operator, we overload a `IntervalDual` version; for example, for division:

```
/(a::IntervalDual, b::IntervalDual) = IntervalDual(
    a.val / b.val,
    [(a_i * b.val - a.val * b_i) / (b.val^2) for (a_i,
        b_i) in zip(a.der, b.der)] # gradient arithmetic
)
```

This allows us to seamlessly switch to `IntervalDual` arithmetic just like ordinary mathematical operations. For example, in the following harmonic RBF boundary function, the implementation is essentially identical to standard mathematical notation:

```
function harmonic_rbf(v)
    # ...parameter definition...
    h = c
    for n in 1:N
        dist = sqrt((x - p[n][1])^2 + (y - p[n][2])^2)
        h += a[n] / dist
    end
    return h
end
```

A similar idea has appeared in recent interval analysis work; for instance, Sharp and Jacobson [2022] replace the standard floating point arithmetic in the forward pass of an MLP with affine arithmetic (a tighter form of interval arithmetic). In principle, our method could also be applied to forward or backward derivatives of MLPs, but this would require substantial engineering effort, which we leave for future work.

Using a non-vectorized system on the CPU results in relatively low performance in our implementation. However, we believe that building a high-performance implementation on top of our work would not pose significant challenges, and the branch-and-bound algorithm is inherently highly parallelizable.

E.2 PDE Solution Generation

For reference solution, we extract the implicit surface using the marching cubes algorithm [Lorensen and Cline 1987] with a resolution of 256^d ($d = 2, 3$), and compute the reference solution with 8192 wpp using `Zombie` [Sawhney and Miller 2023]. Our method is limited by the performance of the symbolic computation system, so we generate experimental results with less wpp. The 2D results were computed with 1024 wpp, while the 3D results used 256 wpp. When sampling the boundary, we follow the approach of Sharp and Jacobson [2022] by first constructing and caching a tree structure to improve performance. In one set of experiments, the evaluation points for different geometric representations (meshes and implicit surfaces) were generated using the same mask. This ensures that the solutions produced by different representations yield identical contours on the slice, eliminating contour inconsistencies caused by small floating-point errors and facilitating side-by-side comparison.

References

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 163–169. doi:10.1145/37401.37422

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Trans. Graph.* 43, 4 (2024).

Rohan Sawhney and Bailey Miller. 2023. *Zombie: Grid-Free Monte Carlo Solvers for Partial Differential Equations.*

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 (2023).

Nicholas Sharp and Alec Jacobson. 2022. Spelunking the deep: guaranteed queries on general neural implicit surfaces via range analysis. *ACM Trans. Graph.* 41, 4, Article 107 (July 2022), 16 pages. doi:10.1145/3528223.3530155