GEOMETRY-ADAPTED STEPPING FOR STOCHASTIC CONTROL VIA WALK-ON-SPHERES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) algorithms with a continuous action space rely on discrete (Euler-Maruyama) time-stepping to simulate continuous-time stochastic dynamics, introducing temporal discretization error and requiring ad hoc collision handling to respect geometric boundaries. These issues may be improved with smaller integration steps, but this increases overall simulation cost, creating a tradeoff between performance and speed. We introduce Walk-on-Spheres (WoS) Monte Carlo rollouts that perform geometry-aware stepping to traverse the state space more efficiently, are collision-free by construction, and do not require time discretization. In the setting of quadratic control costs, relevant to KL-regularized reinforcement learning, we utilize the Doob h-transform to learn an optimal control policy for fixed and goal-conditioned navigation problems with only terminal rewards. We evaluate our method on 2D point-maze and 3D HM3D indoor navigation scenes, demonstrating significant speedups in rollout time while exceeding baseline performance under equal compute budget.

1 INTRODUCTION

Standard reinforcement learning (RL) algorithms rely on discrete time-stepping to simulate environmental dynamics. While natural for discrete-time systems such as Atari games, language models, and strategy games, this introduces temporal discretization error in continuous-time physical systems (Figure 1). Reducing this error requires smaller integration steps and increased simulation cost. Adaptive time-stepping mitigates this issue (Lamba et al., 2006; Gaines & Lyons, 1997), but still requires temporal discretization and does not take geometry of the state space into account.

We introduce and explore the use of the "Walk-on-Spheres" (WoS) algorithm (Muller, 1956), recently revisited in computer graphics (Sawhney & Crane, 2020), to simulate stochastic motion for these problems. Rather than integrating a Brownian trajectory in time, WoS samples exact exits from maximally-inscribed spheres, maximizing the step within a safety radius, and capturing the many steps of an enclosed walk within a single step to the boundary. This explicitly avoids temporal discretization, respects geometric boundaries by construction, and speeds simulation by allowing traversal of the geometric domain in fewer steps. Quicker traversal is particularly helpful in the sparse rewards navigational setting that we explore empirically.

In this work, we apply WoS to design an RL method for continuous time stochastic control problems on Euclidean state spaces $\Omega \subset \mathbb{R}^D$ with terminal rewards and quadratic control costs. In this setting, the Hamiltonian-Jacobi-Bellman equation satisfied by the value function may be linearized via the Hopf-Cole transform to a Laplace equation solved by a harmonic function h (Appendix C). The optimal control policy is given by the Doob h-transform of the standard Brownian motion (Theorem 3.1). Rather than solve for h directly, our method parameterizes and learns the corresponding exit distributions over maximally-inscribed spheres via stochastic optimization with a tailored importance sampler, which can be naturally interpreted as an RL policy. Our contributions are:

- A WoS-style environment integrator that avoids temporal discretization of stochastic trajectories and guarantees boundary collision avoidance without an adhoc mechanism.
- An actor-only learning method for our WoS policy based on the Doob h-transform for fixed goal and goal-conditioned problem with terminal rewards.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- Superior performance of our algorithm in both training speed and average reward against standard RL baselines on 2D mazes and HM3D house scenes, under matched compute budget, due to increased rollout efficiency and better boundary avoidance.

2 Background and Related Works

Our approach connects reinforcement learning with stochastic trajectories to Walk-on-Spheres through conditional Brownian motion. Similar to path integral control (Kappen, 2005), we utilize the linearization of the underlying HJB Equation to define a transition density for our Walk-on-Spheres trajectories. We restrict ourselves to terminal rewards and leave state-dependent rewards to future work. Here we briefly review Walk-on-Spheres (Sawhney & Crane, 2020), stochastic optimal control (Fleming & Soner, 2005), the Doob h-transform (Doob, 1984), and KL Regularized Reinforcement Learning (Levine, 2018). Symbols are defined where they first appear, and Table 22 in Appendix K collects them for reference.

2.1 Walk-on-Spheres

Consider a bounded domain $\Omega \subset \mathbb{R}^D$, with boundary $\partial\Omega$. A function u is called *harmonic* if it satisfies $\Delta u = 0$ with $u = g$ as boundary values on $\partial\Omega$. This same boundary value problem also has a stochastic interpretation through the Feynman–Kac formula. Let W_t be the standard Wiener (Brownian) process. Then u admits the representation Kakutani (1944):

$$u(x) = \mathbb{E}\left[g(W_{\tau_W})\right]. \tag{1}$$

Here $\tau_W := \inf\{t \geq 0 : W_t \notin \Omega\}$ is the first exit time of the process from Ω. Intuitively, $u(x)$ is the first boundary value reached, $g(W_{\tau_W})$, averaged over all Brownian walks starting at x.

A standard way of evaluating this expectation uses Euler–Maruyama steps to simulate Brownian trajectories W_{τ_W}, with step size Δt until the process exits the domain.

$$W_{t+1} = W_t + \sqrt{\Delta t}\,\xi \qquad \text{where } \xi \sim \mathcal{N}(0, \mathbb{I}) \tag{2}$$

This introduces $O(\sqrt{\Delta t})$ discretization error, which can only be traded off with simulation speed. Walk-on-Spheres (WoS) (Muller, 1956; Sawhney & Crane, 2020) evaluates the expectation in Equation 1, using the *mean value theorem* of harmonic functions (Theorem A.1 in Appendix) to recurse (see Appendix B). Rather than simulating the intermediate trajectory steps, it samples the next location directly from the exit distribution of a maximally inscribed sphere (e.g., Appendix Figure 4). Starting from a location $x_k \in \Omega$, WoS constructs the maximally inscribed sphere $B(x_k, R_k)$, where $R_k = \operatorname{dist}(x_k, \partial\Omega)$, and jumps directly to its boundary, stopping once it reaches the ϵ-shell $\partial\Omega^\epsilon := \{x \in \Omega : \operatorname{dist}(x, \partial\Omega) < \epsilon\}$ for some small $\epsilon > 0$. A recursive one-sample estimator for $u(x_k)$ can be expressed as:

$$\widehat{u}(x_k) = \begin{cases} \dfrac{P_B(x_k, x_{k+1})}{q(x_k, x_{k+1})}\widehat{u}(x_{k+1}), & x_{k+1} \sim q(x_k, \cdot), \quad x_k \notin \partial\Omega^\epsilon, \\ g(\mathrm{Proj}_{\partial\Omega}(x_k)), & x_k \in \partial\Omega^\epsilon, \end{cases} \tag{3}$$

where $P_B := \frac{1}{|\partial B(x_k, R_k)|}$ is the uniform exit distribution of a Brownian process from the sphere starting at its center, and $q(x_k, \cdot)$ is some importance-sampling proposal distribution defined on $\partial B(x_k, R_k)$. Standard WoS takes $q(x_k, \cdot) = P_B$, and an informed proposal reduces variance (Huang et al., 2025). These recursive maximal sphere jumps bypass the need for simulating intermediate steps, thereby completely avoiding the need for temporal discretization of trajectories.

2.2 Stochastic Optimal Control

Stochastic optimal control (SOC) provides the continuous-time analogue of reinforcement learning for systems governed by stochastic dynamics (Doya, 2000; Wang et al., 2020). Consider an agent evolving in Ω following a stochastic trajectory X_t as given in Equation 4, with the goal of maximizing a cumulative reward: a control cost $\frac{1}{2}\|a(X_t)\|^2$ and terminal reward $F(X_\tau)$ where X_τ denotes the location when the agent leaves the domain Ω for the first time. Under these assumptions, the

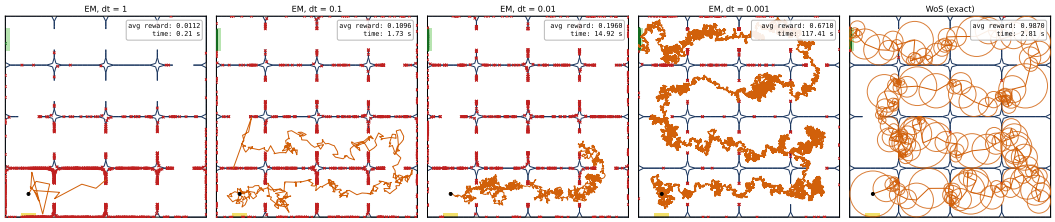


Figure 1: **Discretization error under a perfect control field.** Following Theorem 3.1, we generate the exact optimal control $\sigma^2 \nabla \log h$ for this challenging 2D maze with many narrow "doors" by solving $\Delta h = 0$ with $h|_{\partial\Omega} = \exp(F/\sigma^2)$ on a dense triangulation (FEM), and integrate 1000 walks from one start point in each panel (one most successful walk shown). Green marks the goal (reward $F = 1$), yellow a distractor ($F = 0.3$), and red a wall hit ($F = 0$). Euler–Maruyama (EM) in the first four panels grows more accurate as Δt falls and the number of wall hits decreases, but also requires significantly more computation time. In the rightmost panel, Walk-On-Spheres achieves a higher average reward in less time, 0.987 in $2.81\,\mathrm{s}$, when compared to the best EM result at $\Delta t = 10^{-3}$: 0.671 in $117.41\,\mathrm{s}$. A reward comparison on this maze between (imperfect) trained WoS and SAC policies at various Δt shows superior WoS performance even under equal iterations (Figure 11), demonstrating the detrimental effect of discretization error on actual learned policies.

maximal expected cumulative reward $V(x)$ that an agent can get starting from a state x, the value function, can be expressed as the following optimization problem:

$$V(x) = \sup_{a:\Omega\to\mathbb{R}^D} \mathbb{E}\left[F(X_\tau) - \int_0^\tau \frac{1}{2}\|a(X_t)\|^2\,\mathrm{d}t \middle| X_0 = x\right]; \qquad \mathrm{d}X_t = a(X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t \quad (4)$$

The value function $V(x)$ also admits a continuous version of the Bellman optimality condition, called the Hamilton–Jacobi–Bellman (HJB) equation Fleming & Soner (2005). Interestingly, restricting the HJB to the prior set of assumptions simplifies the HJB equation to the classic static Hamilton–Jacobi Equation (see Appendix C):

$$\frac{\sigma^2}{2}\Delta V(x) + \frac{1}{2}\|\nabla V(x)\|^2 = 0, \quad (5)$$

and the associated optimal control can be expressed as $a^* = \nabla V$.

Although inherently non-linear, a solution to Equation 5 can be obtained by linearizing the PDE using the *Hopf–Cole transform* (see Theorem 3.1 and Appendix C). This linearization technique is widely used in the *Path Integral Control* line of works (Kappen, 2005; Theodorou et al., 2010), where the optimal control is typically estimated from expectations under uncontrolled reference (Brownian) dynamics simulated via time-discretized Monte-Carlo trajectories. However, such uncontrolled rollouts rarely reach high-reward regions, introducing variance into the estimates, especially in sparse-reward problems. Notably, in classic chicken-and-egg fashion, it can be shown that the optimal policy itself can serve as the ideal proposal distribution for the importance sampler (i.e., achieving zero variance), giving rise to various methods that use time-discretized trajectories with a drift for the rollout in place of uncontrolled rollouts (Nüsken & Richter, 2021; Kappen & Ruiz, 2016; Williams et al., 2015; 2017). Since we are training an importance sampler to guide Brownian Walk-on-Spheres trajectories, our method can be classified alongside other importance sampling based methods where the importance sampler is trained to match the optimal policy (in our case, optimal exit distribution for each sphere jump).

There have also been several prior works which aim to build a solver for the HJB equation for a fixed time horizon (Nüsken & Richter, 2021; Domingo-Enrich et al., 2024; Hua et al., 2025; Kazim et al., 2023). In contrast, our work aligns more closely with those that consider the first exit time problem (Quer & Ribera Borrell, 2024; Zhou et al., 2021; Ribera Borrell et al., 2025; Du et al., 2026) except we target a *restricted* class of the HJB equation (i.e., Equation 5).

2.3 KL-Regularized Reinforcement Learning

The quadratic control cost $\frac{1}{2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2 \mathrm{d}t\right]$ in Equation 4 penalizes cumulative control effort. Since $a(X_t)$ represents the mean velocity of the stochastic process, this quadratic term can be interpreted as time-integrated kinetic energy for a trajectory (up to a mass factor). It is widely used in robotics to balance task reward against control effort producing smoother transitions, as simply performing reward maximization can produce bang-bang behavior (i.e., where policies favor extreme control inputs) (Bohez et al., 2019; Seyde et al., 2021).

From a path-measure perspective, it can be shown (see Appendix E) that the quadratic cost is equivalent to a KL divergence between the path measure π induced by the drifted walk $dX_t = a(X_t)\mathrm{d}t + \sigma\, \mathrm{d}W_t$ and the path measure $\mathbb{P}_W$ of Brownian motion $\sigma\, dW_t$:

$$\frac{1}{2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2 \mathrm{d}t\right] = \sigma^2 D_{\mathrm{KL}}(\pi\|\mathbb{P}_W) \tag{6}$$

This leads to a simple expression of the objective 4 in the path measure perspective: $\mathbb{E}_\pi\left[F(X_\tau)\right] - \sigma^2 D_{\mathrm{KL}}(\pi\|\mathbb{P}_W)$, and connects the quadratic control cost naturally to maximum-entropy reinforcement learning. In this setting, KL divergence of π against some reference measure is used to encourage policy exploration and prevent premature mode collapse (Levine, 2018; Hansen et al., 2024; Ziebart, 2018; Fox et al., 2016; Todorov, 2006; Kappen et al., 2013; Nüsken & Richter, 2021; Lv et al., 2026). Our work focuses on the case where the reference measure is $\mathbb{P}_W$ which corresponds to the quadratic control cost. In the following sections, we discuss how the optimal sphere exit distribution for this KL-regularized objective can be characterized by the Doob h-transform.

2.4 Conditional Brownian Motion and the Doob h-Transform

Arising in classical potential theory, a Markov process with transition measure $P(x, \mathrm{d}y)$ can be conditioned against a positive function h to preferentially drift toward regions where h is large (Doob, 1984). This spatial conditioning modifies the reference transition kernel $P(x, \mathrm{d}y)$ to incorporate the effect of a drift without explicitly integrating stochastic trajectories. Now consider the case where the transition measure is the harmonic sphere kernel P_B, which describes the Brownian exit distribution from a sphere $B(x, R)$ in Walk-on-Spheres. Theorem 2.1 gives the corresponding tilted kernel P_B^h, which directly samples sphere exits under the drift induced by h.

Theorem 2.1 (Doob-transformed Walk on Spheres)**.** *Let $h : \Omega \to \mathbb{R}_{>0}$ be harmonic and let $P_B(x, \cdot)$ denote the Brownian exit distribution from $B(x, R) \subset \Omega$, where $R = \mathrm{dist}(x, \partial\Omega)$. Tilting that exit distribution by h,*

$$P_B^h(x, y) = P_B(x, y)\frac{h(y)}{h(x)}, \tag{7}$$

or in direction space, writing the exit as $y = x + R\nu$ with $\nu \in \mathbb{S}^{D-1}$,

$$p_B^h(\nu \mid x) = \frac{1}{|\mathbb{S}^{D-1}|}\frac{h(x + R\nu)}{h(x)}, \tag{8}$$

induces a drift of $\sigma^2 \nabla \log h(x)$ in the corresponding Brownian process.

Proof. Proof in Appendix B □

3 Method

Our primary goal is to build an importance sampler (i.e., our policy) that samples Walk-on-Spheres trajectories according to optimal control a^* (Equation 5). To this end, we first use Theorem 2.1 to describe how the Doob h-transform induces a drift of $\sigma^2 \nabla \log h$ in Walk-on-Spheres trajectories. We then show that, under certain boundary conditions for the harmonic function h, $\sigma^2 \nabla \log h$ is the optimal control drift for Equation 5. (Theorem 3.1)

Theorem 3.1 (Hopf–Cole characterization of the optimal control)**.** *Let V solve the Hamilton–Jacobi equation in Equation 5 with boundary conditions $V = F$ on $\partial\Omega$, and define $h(x) := \exp(V(x)/\sigma^2)$. Then h is the positive harmonic solution of $\Delta h = 0$ on Ω with $h = \exp(F/\sigma^2)$ on $\partial\Omega$, and the optimal control for Equation 4 is*

$$a^*(x) = \sigma^2 \nabla \log h(x). \tag{9}$$

Proof. See Appendix C, which also gives the verification argument that makes a^* optimal. □

Corollary 3.1 (Optimal WoS exit distribution)**.** *With this h, Theorem 2.1 makes the optimal controlled diffusion for Equation 4 the Doob h-transform of the Brownian process, where every WoS sphere exit follows P_B^h of Equation 7.*

3.1 Learning an Importance Sampler to the Optimal Control Policy

Even if an ideal P_B^h from Corollary 3.1 were known accurately (which it is not), representing and sampling an arbitrary distribution on the surface of a sphere is nontrivial. Binning-based samplers such as CDF or alias tables (Walker, 1977) reintroduce the discretization problem and scale poorly with dimension. Thus, we instead learn a parameterized distribution over exit directions, which can be used to determine a sphere exit.

We aim to fit our parameterized model against the directional Doob kernel $p_B^h(\nu \mid x)$ (Theorem 2.1, Equation 8). We follow Huang et al. (2025) and approximate $p_B^h(\nu \mid x)$ with a von Mises–Fisher mixture model q_θ, whose parameters are learned using a neural network:

$$q_\theta(\nu \mid x) = \sum_{i=1}^{K} \lambda_{i,\theta}(x)\mathcal{V}\left(\nu; \omega_{i,\theta}(x), \kappa_{i,\theta}(x)\right), \qquad \nu \in \mathbb{S}^{D-1}, \tag{10}$$

where $\mathcal{V}(\nu; \omega, \kappa)$ is the von Mises–Fisher density, $\lambda_{i,\theta} \geq 0$ the mixture weights ($\sum_i \lambda_{i,\theta} = 1$), and $\omega_{i,\theta} \in \mathbb{S}^{D-1}$ and $\kappa_{i,\theta} \geq 0$ are the mean direction and concentration of mixture component i, all predicted by a network from the current state x. Appendix J covers a defensive variant that mixes q_θ with the uniform density.

Estimating the training target $\widehat{p}_B^h$. Since h is harmonic, a Monte Carlo estimate for p_B^h at a given location x can be built from Monte Carlo estimates of h at the exit point $x + R\nu$ and at the center x. However, building multi-sample estimators for both terms can lead to branching walks, whose count grows exponentially with depth. Therefore, to prevent this, we instead rely on a one sample unbiased estimate for the numerator, and reuse those same samples for the denominator (see Figure 5), since the mean value theorem (Theorem A.1 in Appendix) makes $h(x)$ the average of h over the sphere. The ratio of these two quantities can be used to build a Monte Carlo estimate for $\widehat{p}_B^h$ (Lemma 3.1).

Lemma 3.1 (MC estimate of training target)**.** *Given N WoS trajectories initialized at x, where trajectory n takes L_n steps through states $x_{n,k}$ along sampled directions $\nu_{n,k}$ and stops at $Y_n \in \partial\Omega^\epsilon$, with $F(Y_n)$ the terminal reward at the closest boundary point, define*

$$z_n := \sum_{k=0}^{L_n-1} \log \frac{p_B(\nu_{n,k} \mid x_{n,k})}{q_{\bar{\theta}}(\nu_{n,k} \mid x_{n,k})} + \frac{F(Y_n)}{\sigma^2}, \qquad m^\star := \max_{1 \leq j \leq N} z_j. \tag{11}$$

and the per-walk target for the directional Doob kernel of Equation 8,

$$\widehat{p}_{B,n}^h(\nu_{n,0} \mid x) = q_{\bar{\theta}}(\nu_{n,0} \mid x) \frac{\exp(z_n - m^\star)}{\frac{1}{N}\sum_{j=1}^{N} \exp(z_j - m^\star)}. \tag{12}$$

Then, for fixed $\bar{\theta}$ and any function g on $\mathbb{S}^{D-1}$ with finite expectation under $p_B^h(\cdot \mid x)$,

$$\frac{1}{N}\sum_{n=1}^{N} \frac{\widehat{p}_{B,n}^h(\nu_{n,0} \mid x)}{q_{\bar{\theta}}(\nu_{n,0} \mid x)} g(\nu_{n,0}) \longrightarrow \mathbb{E}_{\nu \sim p_B^h(\cdot|x)}\left[g(\nu)\right] \quad \textit{almost surely as } N \to \infty. \tag{13}$$

Proof. See Appendix D □

Intuitively, we propose a numerically stable way to estimate $\widehat{p}_B^h$. The Monte Carlo estimate $\widehat{p}_B^h$ relies on unbiased Monte Carlo estimates of h, which are then used to calculate the ratio in Theorem 2.1. From Equation 3, we see that for a non-trivial importance sampler q_θ, the h estimator collects the product of p_B/q_θ at each step along a trajectory (its *throughput*). This estimate can take extreme values in rare events. Hence for numerical stabilization, we instead collect the throughput and the boundary contribution for h in log space, and apply an $\exp$ to bring it back. Since the estimator of

Algorithm 1 Online learning of the importance sampler. Our algorithm is structurally identical to the standard RL loop for policy-gradient updates.

Require: Number of rollouts N, number of starting points M, defensive mixture parameter s
Ensure: Trained policy q_θ

```
Initialize θ
repeat
    θ̄ ← θ
    Sample M interior points from Ω                                  ▷ (i) sample initial states
    for each sampled point x do
        Sample ν_{n,0} ~ q_θ̄(· | x) for n ∈ [N]
        x_{n,1} ← x + R_{n,0} ν_{n,0} for n ∈ [N]
        Recursively continue each trajectory from x_{n,1} until Y_n ∈ ∂Ω^ε   ▷ (ii) roll out
        z_n ← F(Y_n)/σ^2 for n ∈ [N]
        for each transition in trajectory n ∈ [N] do
            z_n += log p_B(ν_{n,k} | x_{n,k}) − log q_θ̄(ν_{n,k} | x_{n,k})
        end for
        Compute p̂^h_{B,n}(ν_{n,0} | x) using Equation 12 for n ∈ [N]
        RolloutData ← {(x, p̂^h_{B,n}(ν_{n,0} | x), ν_{n,0})}_{n∈[N]}   ▷ (iii) state, action, reward
        Estimate ∇_θ L = (1/N) Σ_{n∈[N]} ∇̂_θ D_KL^(n) using Equation 14 and RolloutData
        Gradient Descent on θ using ∇_θ L                            ▷ (iv) update policy
    end for
until convergence
return q_θ
```

p_B^h is a ratio of these h estimates, it arises rise to a classic "exp-over-sum-of-exp" ratio, which can be further stabilized using softmax style offsetting by the group maximum $m^\star$ Goodfellow et al. (2016).

Online KL fitting. Since our goal is to approximate p_B^h using the vMF mixture model q_θ, we follow Huang et al. (2025) to fit the q_θ by minimizing its directional KL divergence from the Doob kernel p_B^h. As described in Algorithm 1, our training data $\hat{p}_B^h$ for q_θ comes only from the first sphere of each walk to avoid generating branching walks. Since for the first sphere, the first direction $\nu_{n,0}$ is sampled from $q_{\bar{\theta}}(\cdot \mid x)$, the importance weighted gradient estimate from walk n is

$$
\begin{aligned}
D_{\mathrm{KL}}\left(p_B^h \| q_\theta; x\right) &= \int_{\mathbb{S}^{D-1}} p_B^h(\nu \mid x) \log \frac{p_B^h(\nu \mid x)}{q_\theta(\nu \mid x)} \, \mathrm{d}\nu, \\
\widehat{\nabla_\theta D_{\mathrm{KL}}}^{(n)} &= -\frac{\widehat{p}_{B,n}^h(\nu_{n,0} \mid x)}{q_{\bar{\theta}}(\nu_{n,0} \mid x)} \nabla_\theta \log q_\theta(\nu_{n,0} \mid x).
\end{aligned}
\tag{14}
$$

The average of these estimates over the N walks converges to $\nabla_\theta D_{\mathrm{KL}}(p_B^h \| q_\theta; x)$, by Lemma 3.1 applied to $g = \nabla_\theta \log q_\theta$. Algorithm 1 collects these steps into the training loop.

4 Experimental Setup

We evaluate on point navigation, where a walk must reach a goal in cluttered geometry: four 2D mazes, and twelve indoor scenes drawn uniformly at random from the Habitat–Matterport 3D (HM3D) dataset (Ramakrishnan et al., 2021). On HM3D we consider a (i) *fixed goal*: spawn one goal location (as a sphere of fixed radius) and fit a policy to learn to navigate to that sphere. (ii) *conditioned goals*: a goal sphere (per starting point) is spawned uniformly at random in the domain and the policy learns how to navigate between two arbitrary locations in the domain. For conditioned goal problem, the policy also takes the goal location as an input. All methods share one environment, whose integrator is either Euler–Maruyama or Walk-on-Spheres, both implemented in a wavefront-style fashion using NVIDIA Warp Macklin (2022). Unlike WoS, an Euler–Maruyama step can cross a wall (Figure 13), so we perform a ray intersection test against $\partial\Omega$ and terminate walks on any wall hit. The geometric queries (distance function query, closest point query in WoS,

and ray intersection test) are done using NVIDIA Warp Macklin (2022). The WoS based policy is a vMF mixture (Equation 10) parameterized by a lightweight ReLU MLP outputting the raw parameters $\{\lambda'_i, \kappa'_i, \omega'_i\}_{i=1}^{K}$ (where K is the number of vMF components), which are then transformed according to Table 2. The inputs to the MLP are encoded using a multi-resolution hash grid (Müller et al., 2022; Müller, 2021). For *conditioned* goals, we concatenate the goal location to the spatial query points before passing it to the network. Appendix H describes the full setup.

Choosing Baselines. For standard RL algorithms, we use REINFORCE (Williams, 1992), SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017). SAC and PPO use the standard MLP policies of Stable-Baselines3 (Raffin et al., 2021). REINFORCE also uses an MLP policy, with an average return as a baseline for variance reduction. We also compare our method against PICE (Kappen & Ruiz, 2016) due to its mathematical similarities with our method (although it uses Euler–Maruyama trajectories). PICE is defined on a fixed horizon by default. To properly test against it, we instead run each iteration until a boundary is hit. We follow Holdijk et al. (2023) to parameterize the PICE policy with a neural network, and give it the same hash-grid encoder as our vMF model. Finally, we benchmark against MPPI (Williams et al., 2015; UM-ARM-Lab, 2019). MPPI is a planner and does not produce stochastic trajectories. We therefore use MPPI to plan trajectories from starting positions all over the domain, and report the average reward they collect. On the 2D mazes, we also report an FEM baseline, where we solve for the h function using FEM, and use CDF sampling with Walk on Spheres to get the average reward. This FEM strategy is not available on the 3D house scenes as they are incomplete, partially open geometries. We omit the HJB solver of Zhou et al. (2021), which did not make progress on our simplest maze.

A common control problem. For the comparison to be meaningful, every method must solve the same control problem. We therefore give the RL baselines the running reward $-\frac{1}{2}\int_0^\tau \|a(X_t)\|^2 \,\mathrm{d}t$ of Equation 4, where a is the mean velocity of the stochastic trajectories. We also fix their policy standard deviation at $\sigma/\sqrt{\Delta t}$, and disable entropy tuning (Appendix H). Appendix G gives the discrete penalty that imposes this running reward on the Euler–Maruyama methods. These settings are part of the problem definition, as tuning entropy and learning the standard deviation would make SAC and PPO solve a different equation from Equation 5, which would render them incomparable with our method.

Metrics. Each metric averages over starting points drawn uniformly from the domain (Appendix H). We report the objective (Equation 4), average terminal reward, success ratio, average length of successful walk, average number of steps taken by a successful walk, and time taken to unroll successful walks. For the objective 4, Euler-Maruyama methods report $E[F(X_\tau) - \frac{1}{2}\sum \|a(X_t)\|^2 \Delta t]$ along the trajectory, while WoS and FEM report $\mathbb{E}_\pi[F] - \sigma^2 D_{\mathrm{KL}}(\pi \| \mathbb{P}_W)$ (which are equivalent by Lemma F.1, Appendix F). Walk lengths are calculated by the sum of segment lengths $\sum_{k=0} \|X_{k+1} - X_k\|$, with a segment per Δt step for EM, and a segment per sphere exit for WoS; walk steps counts those segments. We don't compute the avg walk length and avg walk steps for MPPI as it produces deterministic trajectories not stochastic. Table 1 reports averages over scenes, Appendix I reports each scene individually.

5 Results

We designed our experiments to investigate the performance of our Walk-on-Spheres based policy against standard baselines using Euler-Maruyama rollouts under a fixed compute budget. We demonstrate superior performance across key metrics, including average reward, rollout time and training time of the algorithms. This is due to the fact that our method takes large jumps in the state-space, resulting in significantly fewer steps to reach the boundary and therefore fewer neural network queries compared to standard baselines.

5.1 Training Under a Fixed Compute Budget

All experiments were conducted on a NVIDIA Tesla V100 GPU capped at 11 hours or 20,000 training iterations (whichever is earlier). Our method returns the highest average reward and the highest objective (Equation 4) across all scenes and problem definitions (fixed goals and goal-conditioned)

Table 1: **Averages over scenes.** Objective is $\mathbb{E}[F(X_\tau)] - \frac{1}{2}\mathbb{E}\int_0^\tau \|a\|^2 \, \mathrm{d}t$; success is the fraction of walks that reach the goal; walk length is the distance covered and walk steps the number of steps taken, both averaged over successful walks; time is wall-clock milliseconds per successful walk. Bold marks the best entry in each column and underline the second best; the FEM reference is not ranked. *N/A* marks an average that is undefined because a method has no successful result in some scene or skipped for a different reason (Table 21). Per-scene values are in Tables 4–9 (2D), 16–21 (HM3D, fixed goal), and 10–15 (HM3D, conditioned goals).

Method	Objective ↑	Reward ↑	Success ↑	Walk length ↓	Walk steps ↓	Time (ms) ↓
2D mazes, fixed goal						
FEM reference	0.862	0.987	0.984	2.50	44.5	0.03
MPPI	-0.372	0.337	0.171	*N/A*	*N/A*	2.60
PICE	0.555	0.608	0.510	4.87	120.9	0.05
REINFORCE	0.717	0.858	0.823	4.04	95.3	0.02
SAC	0.755	0.918	0.897	4.29	102.0	0.02
PPO	0.759	0.912	0.890	5.20	124.6	0.02
WoS (ours)	**0.872**	**0.996**	**0.996**	**2.62**	**50.1**	**0.01**
HM3D, fixed goal						
MPPI	-6.890	0.203	0.003	*N/A*	*N/A*	19728.08
PICE	0.249	0.328	0.160	170.75	3378.0	43.39
REINFORCE	0.229	0.249	0.061	44.75	883.7	217.49
SAC	0.203	0.211	0.014	**28.36**	564.4	546.40
PPO	0.236	0.267	0.083	31.70	624.4	1013.20
WoS (ours)	**0.364**	**0.600**	**0.501**	36.64	**138.9**	**0.05**
HM3D, conditioned goals						
PICE	0.233	0.254	0.068	175.82	3481.5	23.47
REINFORCE	0.202	0.204	0.005	54.22	1073.5	718.33
SAC	0.197	0.201	0.002	23.11	461.8	381.21
PPO	0.200	0.201	0.002	79.59	1577.3	359.01
WoS (ours)	**0.300**	**0.391**	**0.239**	**22.63**	**88.3**	**0.15**

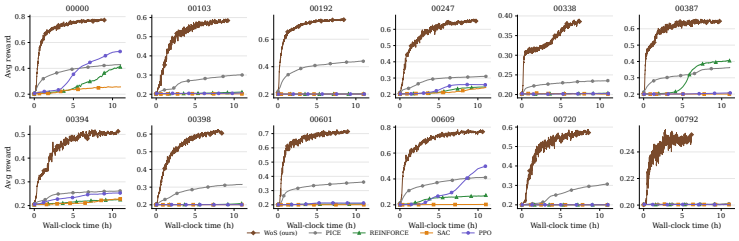


Figure 2: **HM3D, fixed goal.** Training reward against wall-clock time, one run per method, under the budget of Table 3. Our curves end early because our method reaches the iteration cap first; Figure 9 plots the same runs against iterations.

(Figures 2 and 3). This gain comes from the discretization free nature of the integrator together with training speed. Since our Monte Carlo rollouts are cheap, we train orders of magnitude faster than our baselines (which use a small Δt). In almost all scenes, 2D and 3D alike, we finish all 20,000 iterations within the 11-hour time cap (Table 3). The exceptions are a few goal conditioned scenes in HM3D, where our Walk-on-Spheres algorithm was cut off by the cap. Among the baselines, PPO did not finish the 20,000 iterations in any scene, 2D or 3D. This comes from the expensive Monte Carlo rollouts and multiple gradient steps taken per training iteration (10 steps for 2D scenes, and 2 steps for 3D scenes). On the 3D scenes, no baseline method reached 5,000 iterations before the time cap. See Figure 8, Figure 9, and Figure 10 for convergence plots.

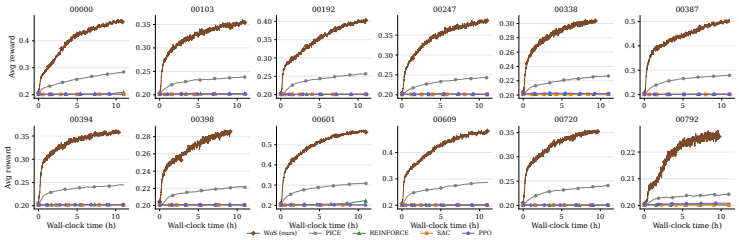


Figure 3: **HM3D, conditioned goals.** As in Figure 2; Figure 10 plots the same runs against iterations.

5.2 Cost of a Successful Walk

Time to roll out a successful (i.e., goal-reaching) trajectory separates the integrators most. Our method has the lowest time per successful walk in every scene (Tables 21, 15, 9). This is more apparent in the 3D scenes, where we achieve a $>$100x speedup compared to the next best baseline. Both Walk-on-Spheres and Euler Maruyama require geometric queries, which we accelerate with a BVH tree (Pharr & Jakob, 2023, Chapter 7). Collision tests are faster than closest point queries, but WoS takes much larger (and therefore fewer) steps than Euler-Maruyama, outweighing the benefit per query. Similarly, Walk on Spheres performs far fewer neural network queries (due to the fewer number of steps), further contributing to its superior performance.

6 Conclusion

To the best of our knowledge, we have formulated the first stochastic RL method that operates with Walk-on-Spheres (WoS) stepping, moving in a geometry-adaptive way through the state space. This avoids the tradeoff between performance and speed presented by temporal discretization and greatly decreases the number of boundary collisions. The geometry-aware stepping allows for quicker traversal of the state space, making rollouts and thus, training iterations, much cheaper, contributing to our superior performance on our 2D/3D navigation tasks when compared to standard baselines that use an Euler-Maruyama method for simulation.

The policy for our method takes the form of learned exit distributions on maximally-inscribed spheres, leveraging the machinery of the Hopf–Cole transformation and the Doob h-transform available to us in the setting of quadratic control costs. These control costs motivate smoother controls in robotics, avoiding sharply changing control inputs, and also arise in maximum-entropy reinforcement learning, where the term encourages policy exploration and prevents mode collapse. Our sphere exit distributions are parameterized by von Mises-Fisher mixtures, and are stochastically optimized via WoS sampling of trajectories under the current policy.

6.1 Limitations and Future Work

Our method provides a discretization-free way to simulate stochastic optimal control trajectories using Walk-on-Spheres, where an agent "teleports" directly to the boundary of maximally inscribed spheres within a bounded domain. However, physical systems cannot execute such instantaneous transitions, leaving the extraction of realizable control inputs from trained WoS policies as an interesting direction for future work. Second, our algorithm requires access to a distance-to-boundary function, which limits its direct applicability to arbitrary RL environments where typically only a collision checker is available. Third, our method is closer in spirit to standard policy-gradient algorithms such as REINFORCE: it uses neither a critic for variance reduction nor replay or rollout buffers to improve sample efficiency. Incorporating these components remains an enticing direction for future work that may even further boost the performance of our relatively simple learning method. Lastly, our algorithm currently addresses first exit time problems with terminal rewards modeled through Dirichlet boundary conditions, but one might naturally consider extensions to infinite-horizon problems, Neumann boundary conditions, and scenes with state-dependent rewards. We feel that all of the above limitations present clear opportunities to build upon our work as a first step in exploring WoS-based reinforcement learning.

AI USE STATEMENT

The core idea of the work was conceived by the authors with no help from generative models. However, we did occasionally use generative LLMs to brainstorm ideas for covering minor edge cases and to boost the performance of our algorithm. For writing code and experiment design, we verified the work by manually simplifying any generated code and designing extensive test cases. We also relied on open-source libraries as much as possible to minimize sources of AI-generated errors. Functions / code fragments that couldn't be imported were ported one-to-one using AI models (with exact line numbers cited in the code) and then manually verified by the authors for correctness.

Finally, for writing, generative models were used to clean up the draft and fix grammatical errors. This was done section by section, with extensive oversight to make sure that no factual or conceptual errors were introduced during the process. We also used LLMs to find related works and baselines, which the authors manually went through. All the citations in the references were extracted by the authors from paper websites and were not generated by LLMs. For proving mathematical claims, proofs were done manually and then cross-verified with generative AI to make sure nothing was missed and no wrong assumptions were made.

ETHICS STATEMENT

This work is simulation only, and involves no human subject, no user study and no personal data collection. For testing our algorithm, we use Habitat Matterport 3D dataset, which is publicly available under academic research license. Our algorithm only uses the geometric information of the dataset (i.e., distance to boundary queries and closest point queries), so no imagery of the real interior of the houses or it's occupants is used during the training.

REPRODUCIBILITY STATEMENT

All experiments are fully described in the submission, including hyperparameters, and training procedures. The codebase will be released with the camera ready paper deadline.

ACKNOWLEDGMENTS

TODO: To be filled out in the Camera Ready version.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018. URL https://arxiv.org/abs/1807.06757.

Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control, 2019. URL https://arxiv.org/abs/1902.04623.

Carles Domingo-Enrich, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky T. Q. Chen. Stochastic optimal control matching. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

J. L. Doob. *Conditional Brownian Motion*, pp. 668–702. Springer New York, New York, NY, 1984. ISBN 9781461252085. doi: 10.1007/978-1-4612-5208-5_29. URL http://dx.doi.org/10.1007/978-1-4612-5208-5_29.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1): 219–245, 2000. doi: 10.1162/089976600300015961.

Yuanqi Du, Jiajun He, Dinghuai Zhang, Eric Vanden-Eijnden, and Carles Domingo-Enrich. Rare event analysis via stochastic optimal control, 2026. URL https://arxiv.org/abs/2604.13213.

Lawrence C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1998. ISBN 0-8218-0772-2.

Wendell H Fleming and Halil Mete Soner. *Controlled Markov processes and viscosity solutions*. Stochastic Modelling and Applied Probability. Springer, New York, NY, 2 edition, November 2005.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, pp. 202–211, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115.

J. G. Gaines and T. J. Lyons. Variable step size control in the numerical solution of stochastic differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, October 1997. ISSN 1095-712X. doi: 10.1137/s0036139995286515. URL http://dx.doi.org/10.1137/S0036139995286515.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

Lars Holdijk, Yuanqi Du, Ferry Hooft, Priyank Jaini, Bernd Ensing, and Max Welling. Stochastic optimal control for collective variable free sampling of molecular transition paths, 2023. URL https://arxiv.org/abs/2207.02149.

Mengjian Hua, Mathieu Lauriere, and Eric Vanden-Eijnden. An efficient on-policy deep learning framework for stochastic optimal control. In *ICLR Workshop on Frontiers in Probabilistic Inference: Learning meets Sampling*, Singapore, 2025. ICLR. URL https://openreview.net/forum?id=sv5PiLZbUr.

Tianyu Huang, Jingwang Ling, Shuang Zhao, and Feng Xu. Guiding-based importance sampling for walk on stars. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, SIGGRAPH Conference Papers '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715402. doi: 10.1145/3721238.3730593. URL https://doi.org/10.1145/3721238.3730593.

Shizuo Kakutani. Two-dimensional brownian motion and harmonic functions. *Proceedings of the Japan Academy, Series A, Mathematical Sciences*, 20(10), January 1944. ISSN 0386-2194. doi: 10.3792/pia/1195572706. URL http://dx.doi.org/10.3792/pia/1195572706.

H J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11011–P11011, November 2005. ISSN 1742-5468. doi: 10.1088/1742-5468/2005/11/p11011. URL http://dx.doi.org/10.1088/1742-5468/2005/11/P11011.

H. J. Kappen and H. C. Ruiz. Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162(5):1244–1266, January 2016. ISSN 1572-9613. doi: 10.1007/s10955-016-1446-7. URL http://dx.doi.org/10.1007/s10955-016-1446-7.

Hilbert Kappen, Vicenç Gomez, and Manfred Opper. Optimal control as a graphical model inference problem. *Proceedings of the International Conference on Automated Planning and Scheduling*, 23:472–473, June 2013. ISSN 2334-0835. doi: 10.1609/icaps.v23i1.13573. URL http://dx.doi.org/10.1609/icaps.v23i1.13573.

Muhammad Kazim, JunGee Hong, Min-Gyeom Kim, and Kwang-Ki K. Kim. Recent advances in path integral control for trajectory optimization: An overview in theoretical and algorithmic perspectives, 2023. URL https://arxiv.org/abs/2309.12566.

Harbir Lamba, Jonathan C. Mattingly, and Andrew M. Stuart. An adaptive euler-maruyama scheme for sdes: convergence and stability. *IMA Journal of Numerical Analysis*, 27:479–506, 2006. URL https://api.semanticscholar.org/CorpusID:1280130.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL https://arxiv.org/abs/1805.00909.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, and Xiao Ma. Flac: Maximum entropy rl via kinetic energy regularized bridge matching, 2026. URL https://arxiv.org/abs/2602.12829.

Miles Macklin. Warp: A High-performance Python Framework for GPU Simulation and Graphics, March 2022. URL https://github.com/NVIDIA/warp. NVIDIA GPU Technology Conference (GTC).

Mervin E. Muller. Some continuous monte carlo methods for the dirichlet problem. *The Annals of Mathematical Statistics*, 27(3):569–589, September 1956. ISSN 0003-4851. doi: 10.1214/aoms/1177728169. URL http://dx.doi.org/10.1214/aoms/1177728169.

Thomas Müller. tiny-cuda-nn. NVIDIA, 2021. URL https://github.com/NVlabs/tiny-cuda-nn. Version 2.0.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

Nikolas Nüsken and Lorenz Richter. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *SN Partial Differ. Equ. Appl.*, 2(4), August 2021.

Matt Pharr and Wenzel Jakob. *Physically Based Rendering, fourth edition*. MIT Press, London, England, March 2023.

J. Quer and Enric Ribera Borrell. Connecting stochastic optimal control and reinforcement learning. *Journal of Mathematical Physics*, 65(8), August 2024. ISSN 1089-7658. doi: 10.1063/5.0140665. URL http://dx.doi.org/10.1063/5.0140665.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL https://jmlr.org/papers/v22/20-1364.html.

Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai, 2021. URL https://arxiv.org/abs/2109.08238.

Enric Ribera Borrell, Lorenz Richter, and Christof Schuette. Reinforcement learning with random time horizons. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5101–5123. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/borrell25a.html.

Rohan Sawhney and Keenan Crane. Monte carlo geometry processing: a grid-free approach to pde-based methods on volumetric domains. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392374. URL https://doi.org/10.1145/3386569.3392374.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies, 2021. URL https://arxiv.org/abs/2111.02552.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, December 2010. ISSN 1532-4435.

Emanuel Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/d806ca13ca3449af72a1ea5aedbed26a-Paper.pdf.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

UM-ARM-Lab. pytorch_mppi. University of Michigan Autonomous Robotics and Manipulation Laboratory, 2019. URL https://github.com/UM-ARM-Lab/pytorch_mppi.

Eric Veach. *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162.

Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, September 1977. ISSN 0098-3500. doi: 10.1145/355744.355749. URL https://doi.org/10.1145/355744.355749.

Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198): 1–34, 2020. URL http://jmlr.org/papers/v21/19-144.html.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015. URL https://arxiv.org/abs/1509.01149.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information theoretic model predictive control: Theory and applications to autonomous driving, 2017. URL https://arxiv.org/abs/1707.02342.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Mo Zhou, Jiequn Han, and Jianfeng Lu. Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066, 2021. doi: 10.1137/21M1402303. URL https://doi.org/10.1137/21M1402303.

Brian D. Ziebart. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. 7 2018. doi: 10.1184/R1/6720692.v1. URL https://kilthub.cmu.edu/articles/thesis/Modeling_Purposeful_Adaptive_Behavior_with_the_Principle_of_Maximum_Causal_Entropy/6720692.

Appendix Contents

A Walk on Spheres **14**

B Proof of Theorem 2.1 **15**

C Proof of Theorem 3.1 **16**

D Proof of Lemma 3.1 **17**

E Control Cost as a Path-Space KL Divergence **18**

F KL Divergence of the Walk on Spheres Policy **19**

G Running Cost for Euler–Maruyama Methods **19**

H Experimental Setup **20**

I More Results **23**

I.1 Convergence 23

I.2 Trajectories 26

I.3 Per-Scene Metrics 26

J Defensive Sampling of the Policy During Training **33**

K Notation **33**

A Walk on Spheres

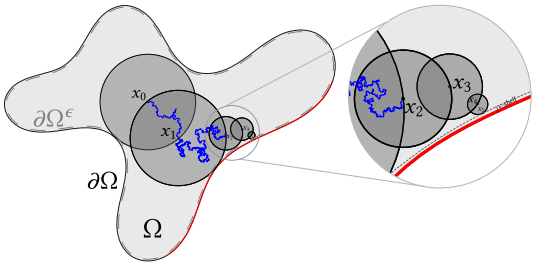


Figure 4: Walk on Spheres takes maximal leaps instead of simulating intermediate trajectory steps, thereby eliminating any error introduced by discretization.

Theorem A.1 (Mean value property)**.** *Let u be harmonic on Ω. Then for every sphere $B(x, R) \subset \Omega$, the value of u at the center equals its average over the surface (Evans, 1998),*

$$u(x) = \frac{1}{|\partial B(x, R)|} \int_{\partial B(x,R)} u(y)\, \mathrm{d}y = \mathbb{E}_{y \sim P_{B(x,\cdot)}} [u(y)]\,. \tag{15}$$

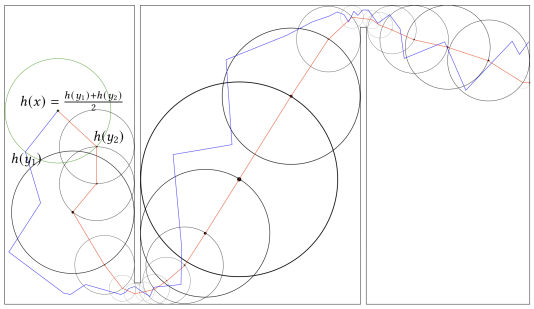


Figure 5: Without loss of generality, lets consider two Walk on Spheres trajectories Red and Blue. We are interested in finding the ratio $h(y_1)/h(x)$ and $h(y_2)/h(x)$. Hence we simulate two Walk on Spheres trajectories starting from $h(x)$. $h(y_1) = T_{y_1 \to Y_\tau^1} \cdot \exp(F/\sigma^2)$ and $h(y_2) = T_{y_2 \to Y_\tau^2} \cdot \exp(F/\sigma^2)$ Now since h is a harmonic function and satisfies the *mean value theorem* A.1, An estimate of $h(x)$ can be obtained by averaging $h(y_1)$ and $h(y_2)$. In our work, for x, we consider two data points $h(y_1)/h(x)$ and $h(y_2)/h(x)$, where the numerator is a one walk estimate, and the denominator uses walks. This estimate of the doob ratio, can be further simplified and numerically stabilized via using 3.1)

.

B PROOF OF THEOREM 2.1

Proof. We show that the tilted kernel is a normalized transition density, that its directional form is as claimed, and that the resulting process carries drift $\sigma^2 \nabla \log h$.

Let u be harmonic and write $u(x) = v(x)h(x)$, where $h > 0$ is harmonic. This implies:

$$\nabla u = h\nabla v + v\nabla h, \qquad \Delta u = \nabla \cdot (\nabla u) = h\Delta v + 2\,(\nabla v \cdot \nabla h) + v\Delta h$$

Substituting into $\frac{\sigma^2}{2}\Delta u = 0$ and using that $h > 0$ and $\Delta h = 0$ gives

$$0 = \frac{\sigma^2}{2}\Delta u(x) = \frac{\sigma^2}{2}h(x)\Delta v(x) + \sigma^2 \nabla v(x) \cdot \nabla h(x),$$

$$\implies 0 = \frac{\sigma^2}{2}\Delta v(x) + \sigma^2 \nabla v(x) \cdot \frac{\nabla h(x)}{h(x)} = \frac{\sigma^2}{2}\Delta v(x) + \sigma^2 \nabla \log h(x) \cdot \nabla v(x).$$

The final form of the PDE above is exactly the steady-state Kolmogorov backward equation associated with a Brownian process with drift $\sigma^2 \nabla \log h(x)$.

For WoS, $P(x, \mathrm{d}y) = P_B(x, y)\mathrm{d}y$. Applying the same substitution to the mean value theorem (Theorem A.1 in Appendix) gives

$$v(x)h(x) = \int_{\partial B(x,R)} v(y)h(y)P_B(x, y)\mathrm{d}y.$$

Dividing by $h(x)$ yields a weighted form of the mean value theorem (Theorem A.1 in Appendix)

$$v(x) = \int_{\partial B(x,R)} v(y)\left[P_B(x, y)\frac{h(y)}{h(x)}\right]\mathrm{d}y,$$

which identifies the conditioned transition in Equation 7. Since h is harmonic,

$$\int_{\partial B(x,R)} h(y)P_B(x, y)\,\mathrm{d}y = h(x),$$

and therefore $P_B^h(x, \cdot)$ integrates to one.

For the directional form, we use the mean value theorem (Theorem A.1 in Appendix) applied to h on $B(x, R)$,

$$\begin{aligned} h(x) &= \frac{1}{|\partial B(x,R)|} \int_{\partial B(x,R)} h(y)\, \mathrm{d}S_y \\ &= \frac{1}{R^{D-1}|\mathbb{S}^{D-1}|} \int_{\mathbb{S}^{D-1}} h(x + R\nu) R^{D-1}\, \mathrm{d}S_\nu \\ &= \frac{1}{|\mathbb{S}^{D-1}|} \int_{\mathbb{S}^{D-1}} h(x + R\nu)\, \mathrm{d}S_\nu. \end{aligned}$$

Thus the factors R^{D-1} from the sphere area and the change of variables cancel. Since the base transition is uniform in direction space,

$$p_B(\nu \mid x) = \frac{1}{|\mathbb{S}^{D-1}|},$$

and substituting $y = x + R\nu$ into the Doob transform gives

$$p_B^h(\nu \mid x) = p_B(\nu \mid x) \frac{h(x + R\nu)}{h(x)} = \frac{1}{|\mathbb{S}^{D-1}|} \frac{h(x + R\nu)}{h(x)}.$$

The mean value theorem (Theorem A.1 in Appendix) verifies that this density integrates to one. □

C Proof of Theorem 3.1

Proof. For a controlled drift f and running control cost r, the value function of the first-exit problem satisfies the Hamilton–Jacobi–Bellman equation (Fleming & Soner, 2005)

$$\frac{\sigma^2}{2} \Delta V(x) + \sup_{a \in \mathcal{A}} \{ f(x, a) \cdot \nabla V(x) - r(x, a) \} = 0, \quad x \in \Omega, \qquad V = F \text{ on } \partial\Omega. \tag{16}$$

This equation expresses the equilibrium that must be satisfied by the value function V, balancing control cost against expected change in the value function $f(x, a) \cdot \nabla V$ and the effect of stochastic noise via the diffusion term $\frac{\sigma^2}{2} \Delta V$.

If $f(x, a) = a$ is the direct control input and $r(x, a) = \frac{1}{2}\|a\|^2$ reflects quadratic control costs, then the second term sup can be obtained by differentiating the inner expression with respect to a and setting to 0 (as it is quadratic and concave in a):

$$\frac{\partial}{\partial a} \left(a \cdot \nabla V(x) - \frac{1}{2}\|a\|^2 \right) = \nabla V(x) - a$$

$$\implies a^*(x) = \arg\max_{a \in \mathbb{R}^D} \left\{ a \cdot \nabla V(x) - \frac{1}{2}\|a\|^2 \right\} = \nabla V(x).$$

Plugging the optimal control $a^*(x)$ back in the HJB equation, we arrive at the static Hamilton-Jacobi equation:

$$\frac{\sigma^2}{2} \Delta V(x) + \frac{1}{2} \|\nabla V(x)\|^2 = 0.$$

Now we consider the Hopf-Cole transformation by defining:

$$h(x) := \exp(V(x)/\sigma^2) \iff V(x) = \sigma^2 \log h(x)$$

$$\implies \nabla V(x) = \sigma^2 \frac{\nabla h(x)}{h}, \qquad \Delta V(x) = \sigma^2 \nabla \cdot \left(\frac{\nabla h(x)}{h} \right) = \sigma^2 \left(\frac{\Delta h(x)}{h} - \frac{\|h(x)\|^2}{h^2} \right).$$

Substituting into the Hamilton–Jacobi equation linearizes it into the Laplace equation:

$$0 = \frac{\sigma^2}{2} \Delta V(x) + \frac{1}{2} \|\nabla V(x)\|^2 = \frac{\sigma^4}{2} \left(\frac{\Delta h(x)}{h} - \frac{\|h(x)\|^2}{h^2} \right) + \frac{\sigma^4}{2} \left(\frac{\|\nabla h\|^2}{h^2} \right) = \frac{\sigma^4}{2h} \Delta h(x).$$

As $\sigma \neq 0$ and $h(x) > 0$, we get that h is harmonic in Ω, and as $V(x) = F(x)$ on $\partial\Omega$, h is subject to Dirichlet boundary conditions:

$$h(x) = \exp\left(\frac{F(x)}{\sigma^2}\right), \qquad x \in \partial\Omega.$$

Finally, following from $V(x) = \sigma^2 \log h(x)$, we get that the optimal drift is given by

$$a^*(x) = \nabla V(x) = \sigma^2 \nabla \log h(x).$$

By Theorem 2.1, this is exactly the drift induced by the Doob h-transform of the passive Brownian process. □

D Proof of Lemma 3.1

Proof. Starting from Equation 8 and writing the first exit of trajectory n as $x_{n,1} = x + R\nu_{n,0}$,

$$p_B^h(\nu_{n,0} \mid x) = p_B(\nu_{n,0} \mid x)\frac{h(x_{n,1})}{h(x)}. \tag{17}$$

For notational convenience within the proof, define the importance weight accumulated along a segment of trajectory n by

$$T_{x_{n,a}\to Y_n} := \prod_{k=a}^{L_n-1} \frac{p_B(\nu_{n,k} \mid x_{n,k})}{q_{\bar\theta}(\nu_{n,k} \mid x_{n,k})}. \tag{18}$$

Using the remaining segment of trajectory n to estimate $h(x_{n,1})$ and the sample mean over N trajectories to estimate $h(x)$ gives

$$\widehat{h}_n(x_{n,1}) = T_{x_{n,1}\to Y_n} \exp\left(\frac{F(Y_n)}{\sigma^2}\right), \qquad \widehat{h}_N(x) = \frac{1}{N}\sum_{j=1}^{N} T_{x_{j,0}\to Y_j} \exp\left(\frac{F(Y_j)}{\sigma^2}\right). \tag{19}$$

Substituting these estimates into Equation 8,

$$\widehat{p}_{B,n}^h(\nu_{n,0} \mid x) = p_B(\nu_{n,0} \mid x)\frac{T_{x_{n,1}\to Y_n} \exp(F(Y_n)/\sigma^2)}{\frac{1}{N}\sum_{j=1}^{N} T_{x_{j,0}\to Y_j} \exp(F(Y_j)/\sigma^2)} \tag{20}$$

$$= q_{\bar\theta}(\nu_{n,0} \mid x)\frac{T_{x_{n,0}\to Y_n} \exp(F(Y_n)/\sigma^2)}{\frac{1}{N}\sum_{j=1}^{N} T_{x_{j,0}\to Y_j} \exp(F(Y_j)/\sigma^2)}, \tag{21}$$

where the second equality follows from

$$p_B(\nu_{n,0} \mid x) T_{x_{n,1}\to Y_n} = q_{\bar\theta}(\nu_{n,0} \mid x) T_{x_{n,0}\to Y_n}. \tag{22}$$

By the definition of z_n in Equation 11,

$$T_{x_{n,0}\to Y_n} \exp\left(\frac{F(Y_n)}{\sigma^2}\right) = \exp(z_n), \tag{23}$$

and therefore

$$\widehat{p}_{B,n}^h(\nu_{n,0} \mid x) = q_{\bar\theta}(\nu_{n,0} \mid x)\frac{\exp(z_n)}{\frac{1}{N}\sum_{j=1}^{N} \exp(z_j)}. \tag{24}$$

Finally, following a softmax style stabilization, multiplying the numerator and denominator by $\exp(-m^\star)$ yields

$$\widehat{p}_{B,n}^h(\nu_{n,0} \mid x) = q_{\bar\theta}(\nu_{n,0} \mid x)\frac{\exp(z_n - m^\star)}{\frac{1}{N}\sum_{j=1}^{N} \exp(z_j - m^\star)}, \tag{25}$$

which is algebraically identical but numerically stable.

For Equation 13, write $w_n = \exp(z_n)$; the factors $\exp(-m^\star)$ cancel. The walks are independent and identically distributed given x, so by the strong law of large numbers, almost surely,

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\widehat{p}^{\,h}_{B,n}(\nu_{n,0}\mid x)}{q_{\bar\theta}(\nu_{n,0}\mid x)}\,g(\nu_{n,0}) = \frac{\frac{1}{N}\sum_n w_n\,g(\nu_{n,0})}{\frac{1}{N}\sum_j w_j} \xrightarrow[N\to\infty]{} \frac{\mathbb{E}\left[w_n\,g(\nu_{n,0})\right]}{\mathbb{E}\left[w_n\right]}.$$

Conditioning on $\nu_{n,0}$, the rest of the walk is the estimate $\widehat{h}_n(x_{n,1})$ of Equation 19, with $\mathbb{E}[\widehat{h}_n(x_{n,1}) \mid \nu_{n,0}] = h(x_{n,1})$. Hence

$$\begin{aligned}\mathbb{E}\left[w_n\,g(\nu_{n,0})\right] &= \mathbb{E}_{\nu\sim q_{\bar\theta}(\cdot|x)}\left[\frac{p_B(\nu\mid x)}{q_{\bar\theta}(\nu\mid x)}\,h(x+R\nu)\,g(\nu)\right]\\ &= \int_{\mathbb{S}^{D-1}} p_B(\nu\mid x)\,h(x+R\nu)\,g(\nu)\,\mathrm{d}\nu\\ &= h(x)\,\mathbb{E}_{\nu\sim p^h_B(\cdot|x)}\left[g(\nu)\right],\end{aligned}$$

where the last step is Equation 8. Setting $g = 1$ gives $\mathbb{E}[w_n] = h(x)$, and the ratio is $\mathbb{E}_{p^h_B}[g(\nu)]$. As with any Walk on Spheres estimate, the ϵ-shell adds a bias that vanishes as $\epsilon \to 0$. □

E Control Cost as a Path-Space KL Divergence

Lemma E.1 (Control cost as a path-measure KL)**.** *Let π be the path measure induced by $\mathrm{d}X_t = a(X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t^\pi$ and let $\mathbb{P}_W$ be the path measure induced by $\sigma\,\mathrm{d}W_t$, both stopped at τ. If $\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right] < \infty$, then*

$$\frac{1}{2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right] = \sigma^2 D_{\mathrm{KL}}\left(\pi\|\mathbb{P}_W\right). \tag{26}$$

Proof. By Girsanov's Theorem, the Radon–Nikodym derivative of π with respect to $\mathbb{P}_W$ is

$$\frac{\mathrm{d}\pi}{\mathrm{d}\mathbb{P}_W} = \exp\left(\frac{1}{\sigma^2}\int_0^\tau a(X_t)\cdot\mathrm{d}X_t - \frac{1}{2\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right). \tag{27}$$

Substituting the controlled dynamics $\mathrm{d}X_t = a(X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t^\pi$ into the first integral,

$$\begin{aligned}\frac{\mathrm{d}\pi}{\mathrm{d}\mathbb{P}_W} &= \exp\left(\frac{1}{\sigma^2}\int_0^\tau a(X_t)\cdot(a(X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t^\pi) - \frac{1}{2\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right) &(28)\\ &= \exp\left(\frac{1}{\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t + \frac{1}{\sigma}\int_0^\tau a(X_t)\cdot\mathrm{d}W_t^\pi - \frac{1}{2\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right) &(29)\\ &= \exp\left(\frac{1}{2\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t + \frac{1}{\sigma}\int_0^\tau a(X_t)\cdot\mathrm{d}W_t^\pi\right). &(30)\end{aligned}$$

Taking the π-expectation of the logarithm,

$$\begin{aligned}D_{\mathrm{KL}}\left(\pi\|\mathbb{P}_W\right) &= \mathbb{E}_\pi\left[\log\frac{\mathrm{d}\pi}{\mathrm{d}\mathbb{P}_W}\right] &(31)\\ &= \mathbb{E}_\pi\left[\frac{1}{2\sigma^2}\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t + \frac{1}{\sigma}\int_0^\tau a(X_t)\cdot\mathrm{d}W_t^\pi\right] &(32)\\ &= \frac{1}{2\sigma^2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right] + \underbrace{\mathbb{E}_\pi\left[\frac{1}{\sigma}\int_0^\tau a(X_t)\cdot\mathrm{d}W_t^\pi\right]}_{=0} &(33)\\ &= \frac{1}{2\sigma^2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right]. &(34)\end{aligned}$$

The stochastic integral $M_s := \int_0^s a(X_t)\cdot\mathrm{d}W_t^\pi$ is a π-martingale with $M_0 = 0$. Since the exit time τ is random, its expectation at τ is evaluated by Doob's Optional Stopping Theorem, which

gives $\mathbb{E}_\pi[M_\tau] = 0$ under the stated integrability condition $\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right] < \infty$. Multiplying through by σ^2 yields Equation 26.

Substituting Equation 26 into the control objective of Equation 4 gives

$$\mathbb{E}_\pi\left[F(X_\tau)\right] - \frac{1}{2}\mathbb{E}_\pi\left[\int_0^\tau \|a(X_t)\|^2\,\mathrm{d}t\right] = \mathbb{E}_\pi\left[F(X_\tau)\right] - \sigma^2 D_{\mathrm{KL}}\left(\pi\|\mathbb{P}_W\right), \tag{35}$$

which is the KL-regularized objective $J(\pi) = \mathbb{E}_\pi\left[F(X_\tau)\right] - \sigma^2 D_{\mathrm{KL}}(\pi\|\mathbb{P}_W)$. □

F KL Divergence of the Walk on Spheres Policy

Lemma F.1. *Let π be the path measure of a walk whose sphere exits $x_0, x_1, \ldots, x_L$ are drawn from our policy and whose path inside each sphere is Brownian motion conditioned on its exit point, where L is the number of steps. Then*

$$D_{\mathrm{KL}}\left(\pi\|\mathbb{P}_W\right) = \mathbb{E}_\pi\left[\sum_{k=0}^{L-1}\Big(\log q_\theta(\nu_k \mid x_k) + \log|\mathbb{S}^{D-1}|\Big)\right].$$

Proof. Given its sphere exits, a walk under π or $\mathbb{P}_W$ fills each sphere with Brownian motion conditioned on its exit point, so the factors for the paths inside the spheres cancel, and the Radon–Nikodym derivative is a product over the exits. The radius $R_k = \mathrm{dist}(x_k, \partial\Omega)$ is determined by x_k, so the exit $x_{k+1} = x_k + R_k\nu_k$ is determined by ν_k, which has density $q_\theta(\nu_k \mid x_k)$ under π and $1/|\mathbb{S}^{D-1}|$ under $\mathbb{P}_W$. Therefore

$$\begin{aligned}
D_{\mathrm{KL}}\left(\pi\|\mathbb{P}_W\right) &= \mathbb{E}_\pi\left[\log\frac{\mathrm{d}\pi}{\mathrm{d}\mathbb{P}_W}\right] \\
&= \mathbb{E}_\pi\left[\log\prod_{k=0}^{L-1}\frac{\mathrm{d}\pi(x_{k+1}\mid x_k)}{\mathrm{d}\mathbb{P}_W(x_{k+1}\mid x_k)}\right] \\
&= \mathbb{E}_\pi\left[\log\prod_{k=0}^{L-1}\frac{q_\theta(\nu_k\mid x_k)}{1/|\mathbb{S}^{D-1}|}\right] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{L-1}\Big(\log q_\theta(\nu_k\mid x_k) + \log|\mathbb{S}^{D-1}|\Big)\right].
\end{aligned}$$

□

G Running Cost for Euler–Maruyama Methods

Lemma G.1. *Let a_θ denote the mean of a learned Gaussian policy, the baselines' counterpart of the control a in Equation 4, and let $\tilde{a}_t \sim \mathcal{N}\left(a_\theta(X_t), \frac{\sigma^2}{\Delta t}\mathbb{I}\right)$ be the sampled action at step t of an Euler–Maruyama walk with exit step τ. Assume $\mathbb{E}[\tau] < \infty$. Then*

$$\mathbb{E}\left[\sum_{t<\tau}\tfrac{1}{2}\|a_\theta(X_t)\|^2\Delta t\right] = \mathbb{E}\left[\sum_{t<\tau}\left(\tfrac{1}{2}\|\tilde{a}_t\|^2\Delta t - \frac{D\sigma^2}{2}\right)\right].$$

Proof. Write $\tilde{a}_t = a_\theta(X_t) + \xi_t$, where $\xi_t \sim \mathcal{N}\left(0, \frac{\sigma^2}{\Delta t}\mathbb{I}\right)$ is independent of the walk up to step t. Expanding the square,

$$\mathbb{E}\Big[\sum_{t<\tau}\tfrac{1}{2}\|\tilde{a}_t\|^2\Delta t\Big] = \mathbb{E}\Big[\sum_{t<\tau}\tfrac{1}{2}\|a_\theta(X_t)\|^2\Delta t\Big] + \mathbb{E}\Big[\sum_{t<\tau}a_\theta(X_t)\cdot\xi_t\,\Delta t\Big] + \mathbb{E}\Big[\sum_{t<\tau}\tfrac{1}{2}\|\xi_t\|^2\Delta t\Big].$$

Since τ is a stopping time and ξ_t is independent of the walk before step t, Wald's identity lets us take the expectation of each summand separately. The cross term then vanishes, and each summand of the last term equals $\frac{1}{2}\cdot\frac{D\sigma^2}{\Delta t}\cdot\Delta t = \frac{D\sigma^2}{2}$. Rearranging gives the claim. □

H EXPERIMENTAL SETUP

Environment. All methods run in one Gymnasium (Towers et al., 2024) vector environment, shared by the 2D and 3D tasks, whose integrator is either Euler–Maruyama or Walk on Spheres. Both integrators are wavefront integrators (Pharr & Jakob, 2023) written in NVIDIA Warp (Macklin, 2022), in which walks advance in lock-step: by Δt for Euler–Maruyama and by a maximal sphere leap for Walk on Spheres (Figure 7). A walk ends when it reaches the ϵ-shell $\partial\Omega^\epsilon$.

Noise. The environment adds no noise. For every learned method, all stochasticity comes from sampling the policy: in the Euler–Maruyama environment the agent moves by $\tilde{a}_t\Delta t$ with $\tilde{a}_t \sim \mathcal{N}\left(a_\theta(X_t), \frac{\sigma^2}{\Delta t}\mathbb{I}\right)$, and since $\tilde{a}_t\Delta t = a_\theta(X_t)\Delta t + \sigma\sqrt{\Delta t}\,\xi_t$ with $\xi_t \sim \mathcal{N}(0, \mathbb{I})$, each move is one Euler–Maruyama step of the diffusion in Equation 4. In the Walk on Spheres environment, the noise is the sampled direction of each leap. MPPI is the exception. It samples noisy control sequences to plan but executes the planned mean action without noise, as the default code of UM-ARM-Lab (2019) does, so its walks are deterministic given its planning samples, and its success ratio, objective, and walk length describe noise-free walks.

Tasks. In 2D, the goal is the rightmost wall of each maze. On HM3D (Ramakrishnan et al., 2021), the goal is a sphere of fixed radius. A walk that ends on the sphere succeeds only if the center of the sphere is in line of sight (Figure 6). We consider two settings.

(i) **Fixed goal.** We take the PointNav episode of each scene with the longest geodesic path (Anderson et al., 2018) and place the goal at its goal position.

(ii) **Conditioned goals.** At each epoch, we sample a goal g uniformly from the interior of the domain, and every learned policy receives g as an additional input.

Evaluation protocol. Every reported quantity averages over the whole domain. With a fixed goal, each table metric comes from one rollout of M starting points, drawn uniformly from the interior with points inside the goal ball rejected, and N walks per starting point (Table 3). With conditioned goals, we draw M starting points and M goals uniformly, pair them, and simulate N walks from each starting point toward its goal. The training curves report the average reward of the training rollouts. The trajectory plots (Figure 12) start from one distant point per scene, the second endpoint of the longest geodesic path on HM3D and a hand-picked point in 2D; no table entry is computed from them.

Baselines. PICE (Kappen & Ruiz, 2016) is defined on a fixed horizon T. We port the code of Holdijk et al. (2023), set T large, and end each episode at the boundary, so that PICE and our method both stop at the exit time τ. MPPI (Williams et al., 2015; UM-ARM-Lab, 2019), a sampling-based planner built on the same path-integral formulation, learns no policy, so we run it at evaluation time only.

Our method treats σ as fixed. To make REINFORCE (Williams, 1992), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018) maximize Equation 4, we change three settings:

1. The running reward is $-\frac{1}{2}\int_0^\tau \|a_\theta(X_t)\|^2\,\mathrm{d}t$, the quadratic control cost of Equation 4.
2. Entropy tuning is disabled ($\alpha = 0$) for SAC and PPO, since the path-space KL term already acts as the entropy regularizer.
3. The policy standard deviation is fixed at $\frac{\sigma}{\sqrt{\Delta t}}\mathbb{I}$, so that all methods sample from the same diffusion.

Section 5 explains why these settings are part of the problem definition.

The quadratic penalty is $\mathbb{E}_\pi\left[\sum_{t<\tau}\frac{1}{2}\|a_\theta(X_t)\|^2\Delta t\right]$. Stable-Baselines3 (Raffin et al., 2021) computes every loss from the sampled action $\tilde{a}_t$, so a penalty on the mean would receive no gradient, and fixing that would mean editing the library. We leave it unchanged. We penalize the sampled action and subtract a constant,

$$\mathbb{E}_\pi\left[\sum_{t<\tau}\left(\frac{1}{2}\|\tilde{a}_t\|^2\Delta t - \frac{D\sigma^2}{2}\right)\right],$$

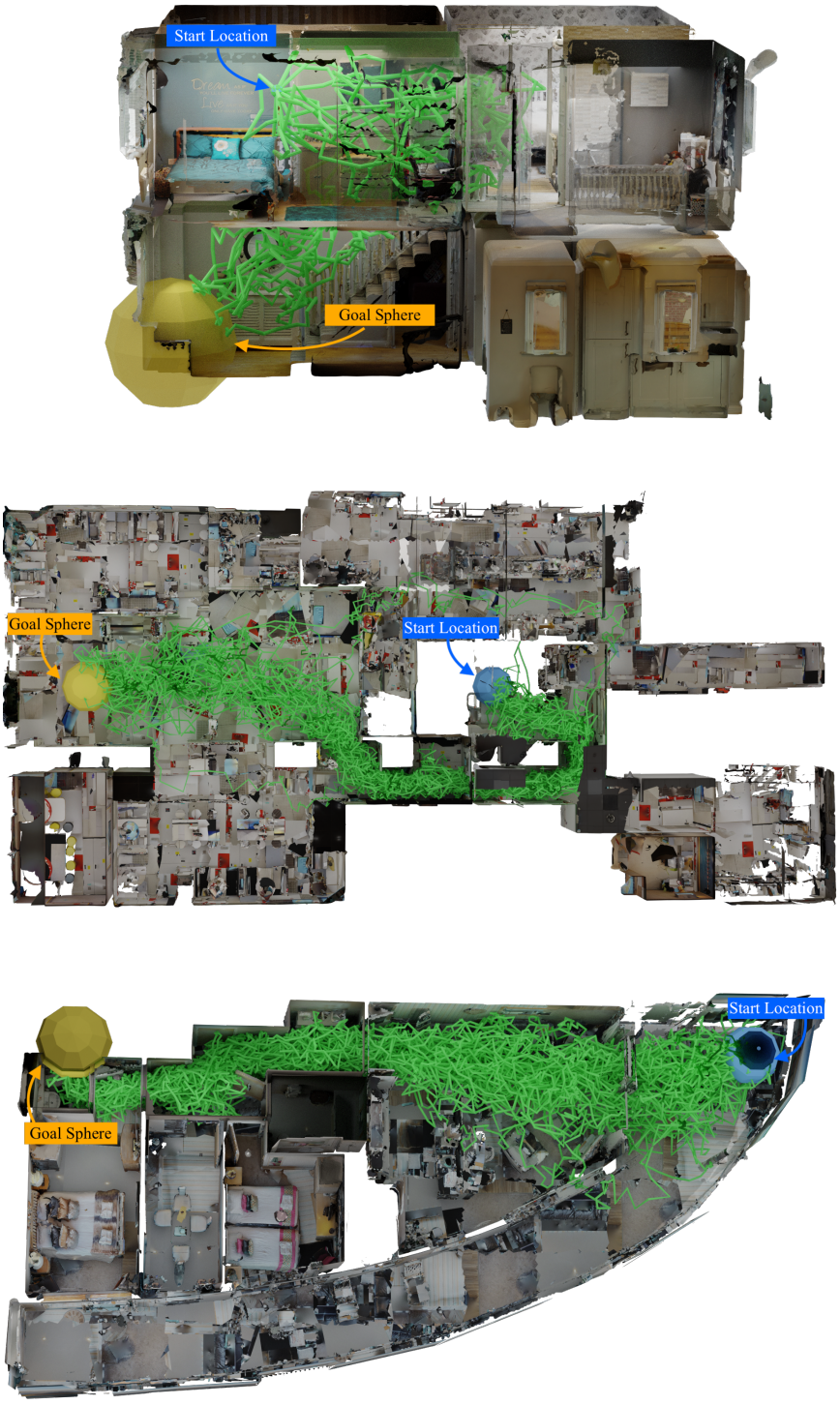


Figure 6: **HM3D scenes.** Walk on Spheres walks are shown in green. The yellow sphere is the goal, and a walk terminates when it hits that sphere. The blue sphere is used only to visually represent the starting point; all walks actually begin at its center.

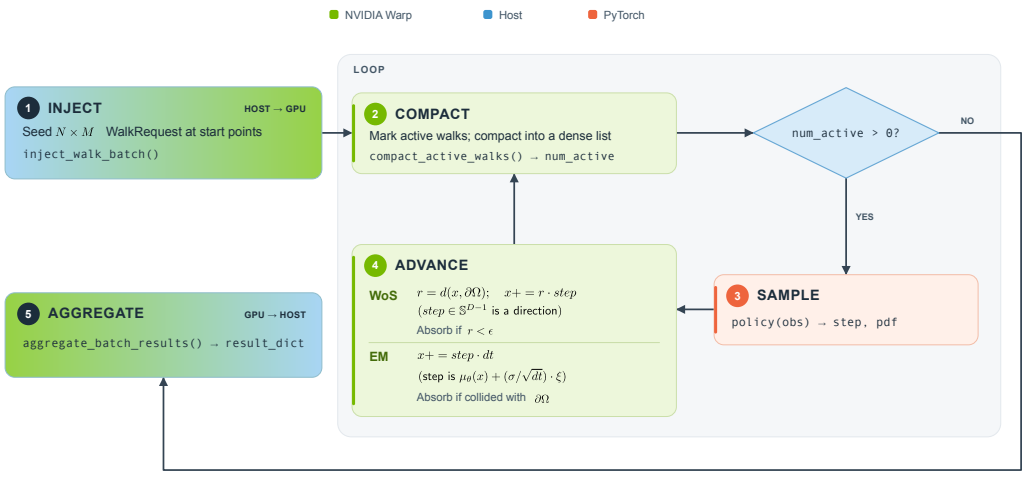


Figure 7: The wavefront environment loop shared by both integrators. After $N \times M$ walk requests are injected at the starting points, each iteration compacts the active walks into a dense list, queries the policy for every active walk in a single batch, and advances them in lock-step. Walk on Spheres advances by a leap of $R = \mathrm{dist}(x, \partial\Omega)$ along the sampled direction and absorbs at $R < \epsilon$, while Euler–Maruyama advances by Δt and absorbs on collision with $\partial\Omega$. The loop ends once no walk remains active.

which equals the required penalty in expectation (Lemma G.1).

On the 2D mazes, the FEM reference solves for h on a triangulation of the maze and drives Walk on Spheres with the resulting kernel P_B^h (Equation 7), sampled from a CDF table. Up to the discretization error of the triangulation and of the table, this is the optimal policy. We omit the HJB solver of Zhou et al. (2021), whose exit-time formulation is close to ours. It did not make progress on the simplest 2D maze.

Policy parameterization. Following Huang et al. (2025), our policy $q_\theta(\nu \mid x)$ is a K-component vMF mixture. A multi-resolution hash grid (Müller et al., 2022), which we run through `tiny-cuda-nn` (Müller, 2021), encodes the query point $x \in \Omega$, and a lightweight MLP with ReLU activations maps the features to raw parameters $\{\lambda_i', \kappa_i', \omega_i'\}_{i=1}^K$. Each raw parameter is mapped to its constrained form by Table 2, which keeps the mixture weights on the simplex, the concentrations positive, and the mean directions on the unit sphere. With conditioned goals, we concatenate the goal with the query point into a single vector and pass that vector through the hash grid. We use PyTorch for inference and gradients.

Table 2: **Policy outputs.** The network emits the raw parameters in the left column; the right column gives the transform applied to each, and the constraint it enforces for every mixture component $i \in [K]$.

Raw output	Transform	Parameter	Constraint
$\lambda_i' \in \mathbb{R}$	$\lambda_i = \exp(\lambda_i') / \sum_{j=1}^K \exp(\lambda_j')$	mixture weight	$\lambda_i \geq 0, \sum_i \lambda_i = 1$
$\kappa_i' \in \mathbb{R}$	$\kappa_i = \exp(\kappa_i')$	concentration	$\kappa_i > 0$
$\omega_i' \in \mathbb{R}^D$	$\omega_i = \omega_i' / \lVert\omega_i'\rVert$	mean direction	$\omega_i \in \mathbb{S}^{D-1}$

PICE uses the hash-grid encoder of our policy, SAC and PPO use the standard MLP policies of Stable-Baselines3, and REINFORCE uses an MLP policy with an average-return baseline. Every method samples M starting points uniformly and N walks from each, the protocol that Walk on Spheres requires.

Table 3: **Configuration.** Settings shared by all methods unless noted. M and N follow Algorithm 1: M starting points per iteration and N walks per starting point.

Setting	Value
Euler–Maruyama step size Δt	0.01 (2D mazes), 0.1 (HM3D)
Noise level σ	0.1
Shell width ϵ (Walk on Spheres)	$10^{-3}\times$ geometric mean of the bounding-box extents of Ω
Terminal reward F	1.0 on the goal, 0.2 on every other wall
Training: walks per starting point N	64
Training: starting points per iteration M	1024
vMF mixture components K	2 (2D Mazes); 3 (HM3D house scenes)
Defensive mixture weight s	1.0 (no uniform component)
Evaluation: starting points M	4096
Evaluation: walks per starting point N	16
Hardware	one Tesla V100 (training); one NVIDIA L40S (evaluation)
Training budget	11 hours or 20,000 iterations, whichever comes first
PICE horizon T	effectively ∞, since we terminate only when the boundary is reached
MPPI perturbations and horizon	32; 100

Training budget. Every method trains under the same wall-clock budget and iteration cap, whichever is more restrictive (Table 3). The two caps differ in how restrictive they are. On HM3D our method reaches the iteration cap within the budget, while the Euler–Maruyama baselines run out of time after fewer than 5,000 iterations, so the comparison is matched in compute and unmatched in iterations (Section 5).

Metrics. We report six metrics: (i) the average terminal reward, (ii) the time per successful walk, (iii) the average length of successful walks, (iv) the average number of steps taken by a successful walk, (v) the success ratio, and (vi) the objective of Equation 4. Walk length sums the Euclidean distances between consecutive recorded points, walk steps counts those points, and the success ratio is the fraction of walks that reach the goal. Walks advance in lock-step, so no walk has its own clock. Metric (ii) therefore divides the wall-clock time of the whole rollout by the number of successes,

$$t_{\mathrm{succ}} = \frac{t_{\mathrm{elapsed}}/NM}{n_{\mathrm{succ}}/NM} = \frac{t_{\mathrm{elapsed}}}{n_{\mathrm{succ}}},$$

the mean time per walk divided by the success ratio. Failed walks thus add to the cost of each success. For the Euler–Maruyama methods, the objective is $\mathbb{E}_\pi\left[F(X_\tau) - \sum_{t<\tau} \frac{1}{2}\|a_\theta(X_t)\|^2 \Delta t\right]$. Walk on Spheres produces no time-discretized controls, so for our method and the FEM reference we evaluate $\mathbb{E}_\pi[F(X_\tau)] - \sigma^2 D_{\mathrm{KL}}(\pi\|\mathbb{P}_W)$ (Lemma F.1). The recorded points of a walk are its Δt steps under Euler–Maruyama and its sphere exits under Walk on Spheres.

I More Results

I.1 Convergence

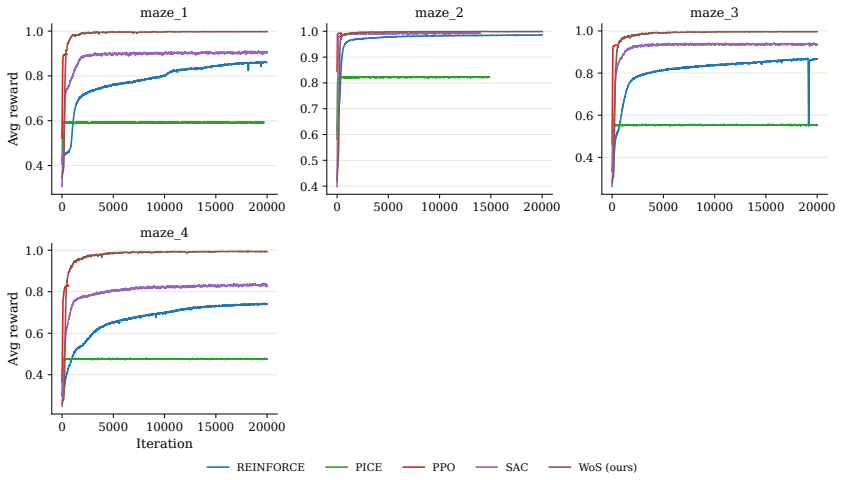


Figure 8: **2D mazes.** Average training reward against iterations, over starting points drawn uniformly from each maze. In 2D, PPO exhausts the 11-hour budget after a few hundred iterations, and every other method reaches the 20,000-iteration cap.

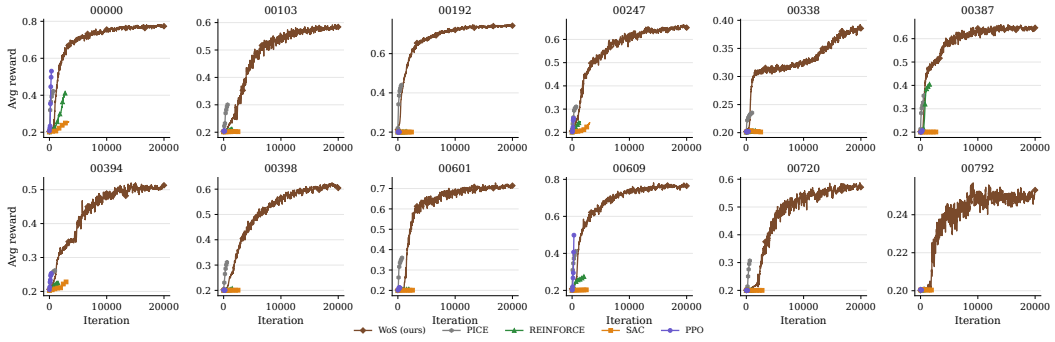


Figure 9: **HM3D, fixed goal.** Average training reward against iterations, for the runs of Figure 2. Within the 11-hour budget, the Euler–Maruyama baselines complete fewer than 5,000 iterations, so their curves end early.

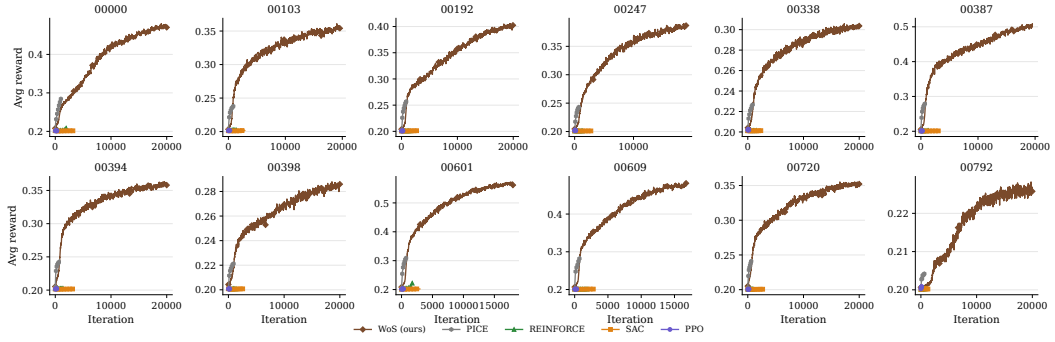


Figure 10: **HM3D, conditioned goals.** Average training reward against iterations, for the runs of Figure 3. Within the 11-hour budget, the Euler–Maruyama baselines complete fewer than 5,000 iterations, so their curves end early.

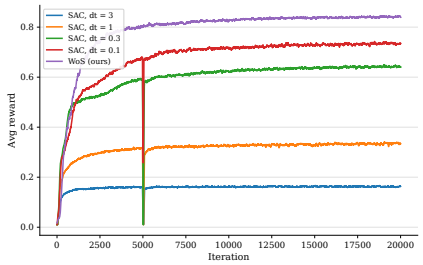


Figure 11: 4×4 **maze from Figure 1.** We trained SAC for various Δt values for an equal number of iterations. The Walk on Spheres policy converged to a higher average value.

I.2 TRAJECTORIES

Figure 12 shows walks of each trained policy on the 2D mazes.

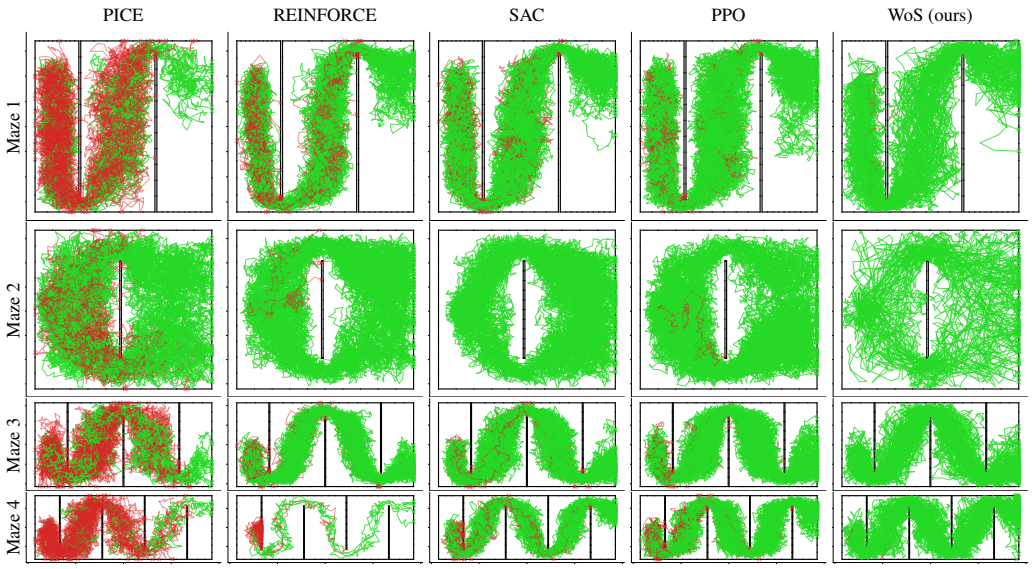


Figure 12: **2D mazes.** Walks of each trained policy from one distant starting point per maze; the goal is the rightmost wall. Green walks reach the goal, and red walks end elsewhere.

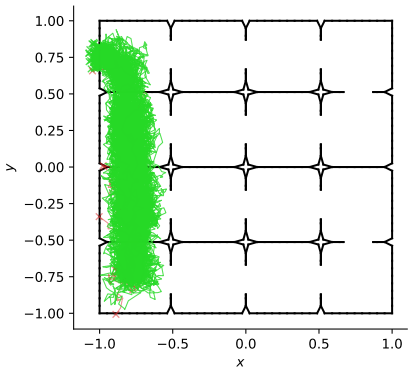


Figure 13: **Lack of Geometric awareness for timestepping integrators**: For the maze problem in Figure 1, when a policy (SAC) is trained using ray collision test is turned off, the geometric barriers are invisible to the learned policy, which in turns passes through walls to get to the goal. This is a problem with all timestepping algorithm as the underlying integrator itself is not geometry aware. For this training, we use $\Delta t = 1e - 2$, with the running cost term turned on.

I.3 PER-SCENE METRICS

Tables 4–21 report every metric for each maze and scene; Table 1 averages them.

Table 4: **Average reward** (↑). 2D mazes, fixed goal. $\mathbb{E}[F(X_\tau)]$. Entries are the mean $\pm$ its standard error over the start points, $s/\sqrt{M}$ with M the number of start points. The value under each column heading is the FEM reference solution (Appendix H).

Method	maze_1 FEM: 0.986	maze_2 FEM: 0.996	maze_3 FEM: 0.986	maze_4 FEM: 0.982
MPPI	0.355 ±0.004	0.442 ±0.004	0.283 ±0.003	0.268 ±0.003
PICE	0.592 ±0.004	0.820 ±0.002	0.549 ±0.003	0.471 ±0.004
REINFORCE	0.854 ±0.002	0.986 ±0.001	0.869 ±0.002	0.723 ±0.004
SAC	<u>0.910</u> ±0.002	0.993 ±4.9e-4	<u>0.934</u> ±0.001	<u>0.834</u> ±0.002
PPO	0.896 ±0.002	<u>0.994</u> ±3.0e-4	0.932 ±0.001	0.827 ±0.002
WoS (ours)	**0.998** ±1.7e-4	**0.999** ±1.1e-4	**0.996** ±2.2e-4	**0.993** ±2.9e-4

Table 5: **Reward minus control cost** (↑). 2D mazes, fixed goal. $\mathbb{E}[F(X_\tau)] - \frac{1}{2}\mathbb{E}\int_0^\tau \|a\|^2 dt$. Entries are the mean $\pm$ its standard error over the start points, $s/\sqrt{M}$ with M the number of start points. The value under each column heading is the FEM reference solution (Appendix H).

Method	maze_1 FEM: 0.858	maze_2 FEM: 0.950	maze_3 FEM: 0.844	maze_4 FEM: 0.795
MPPI	-0.321 ±0.006	-0.695 ±0.007	-0.309 ±0.005	-0.165 ±0.004
PICE	0.537 ±0.005	0.792 ±0.003	0.489 ±0.004	0.401 ±0.004
REINFORCE	0.700 ±0.004	0.923 ±0.001	0.691 ±0.003	0.553 ±0.004
SAC	0.734 ±0.003	0.930 ±0.001	0.743 ±0.003	<u>0.613</u> ±0.004
PPO	<u>0.739</u> ±0.003	<u>0.935</u> ±0.001	<u>0.751</u> ±0.003	0.613 ±0.004
WoS (ours)	**0.870** ±0.002	**0.953** ±0.001	**0.857** ±0.001	**0.809** ±0.002

Table 6: **Success ratio** (↑). 2D mazes, fixed goal. $\Pr[X_\tau \in \mathcal{G}]$. Entries are the mean $\pm$ its standard error over the start points, $s/\sqrt{M}$ with M the number of start points. The value under each column heading is the FEM reference solution (Appendix H).

Method	maze_1 FEM: 0.982	maze_2 FEM: 0.995	maze_3 FEM: 0.983	maze_4 FEM: 0.978
MPPI	0.193 ±0.005	0.303 ±0.005	0.104 ±0.004	0.085 ±0.003
PICE	0.490 ±0.005	0.775 ±0.003	0.436 ±0.004	0.339 ±0.004
REINFORCE	0.818 ±0.003	0.983 ±0.001	0.836 ±0.003	0.654 ±0.005
SAC	<u>0.887</u> ±0.002	0.992 ±0.001	<u>0.917</u> ±0.002	<u>0.793</u> ±0.003
PPO	0.870 ±0.002	<u>0.992</u> ±3.8e-4	0.915 ±0.001	0.784 ±0.003
WoS (ours)	**0.997** ±2.2e-4	**0.999** ±1.4e-4	**0.995** ±2.7e-4	**0.991** ±3.6e-4

Table 7: **Average walk length, successful walks** (↓). 2D mazes, fixed goal. $\mathbb{E}\sum_k \|X_{k+1} - X_k\|$ over walks with $X_\tau \in \mathcal{G}$. The value under each column heading is the FEM reference solution (Appendix H).

Method	maze_1 FEM: 2.59	maze_2 FEM: 1.90	maze_3 FEM: 2.71	maze_4 FEM: 2.80
PICE	4.57	6.68	4.79	3.43
REINFORCE	4.43	<u>5.03</u>	<u>4.13</u>	**2.57**
SAC	<u>4.18</u>	5.18	4.61	3.17
PPO	5.46	5.38	5.57	4.39
WoS (ours)	**2.73**	**1.95**	**2.88**	<u>2.94</u>

Table 8: **Average walk steps, successful walks** (↓). 2D mazes, fixed goal. steps taken over walks with $X_\tau \in \mathcal{G}$. Entries are the mean $\pm$ its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	maze_1 FEM: 44.6	maze_2 FEM: 30.3	maze_3 FEM: 46.6	maze_4 FEM: 56.3
PICE	113.6 $\pm$2.1	167.9 $\pm$1.5	118.3 $\pm$1.8	83.6 $\pm$1.4
REINFORCE	105.2 $\pm$1.3	<u>124.5</u> $\pm$1.0	<u>95.8</u> $\pm$1.0	**55.8** $\pm$0.7
SAC	<u>99.2</u> $\pm$1.1	129.3 $\pm$0.9	108.7 $\pm$1.1	70.7 $\pm$0.8
PPO	131.2 $\pm$1.5	133.4 $\pm$1.1	132.6 $\pm$1.3	101.2 $\pm$1.1
WoS (ours)	**50.1** $\pm$0.4	**31.8** $\pm$0.2	**53.8** $\pm$0.4	<u>64.5</u> $\pm$0.5

Table 9: **Time per successful walk** (↓). 2D mazes, fixed goal. ms, rollout wall-clock / number of successes. The value under each column heading is the FEM reference solution (Appendix H).

Method	maze_1 FEM: 0.03	maze_2 FEM: 0.02	maze_3 FEM: 0.03	maze_4 FEM: 0.04
MPPI	3.12	1.95	2.84	2.48
PICE	0.06	0.04	0.06	0.05
REINFORCE	<u>0.02</u>	<u>0.02</u>	<u>0.01</u>	<u>0.01</u>
SAC	0.02	0.03	0.02	0.01
PPO	0.02	0.03	0.02	0.02
WoS (ours)	**0.01**	**4.2e-3**	**0.01**	**0.01**

Table 10: **Average reward** (↑). HM3D, conditioned goals. $\mathbb{E}[F(X_\tau)]$. Entries are the mean $\pm$ its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	<u>0.288</u> ±0.002	<u>0.239</u> ±0.002	<u>0.260</u> ±0.002	<u>0.242</u> ±0.002	<u>0.226</u> ±0.001	<u>0.286</u> ±0.002	<u>0.243</u> ±0.002	<u>0.222</u> ±0.001	<u>0.316</u> ±0.002	<u>0.284</u> ±0.002	<u>0.243</u> ±0.002	<u>0.204</u> ±0.001
REINFORCE	0.209 ±0.001	0.203 ±4.5e-4	0.201 ±1.9e-4	0.201 ±1.9e-4	0.201 ±2.9e-4	0.202 ±2.6e-4	0.202 ±3.4e-4	0.201 ±1.6e-4	0.225 ±0.001	0.202 ±2.8e-4	0.201 ±1.2e-4	0.200 ±4.4e-5
SAC	0.202 ±2.6e-4	0.202 ±4.2e-4	0.201 ±2.2e-4	0.201 ±2.4e-4	0.201 ±2.4e-4	0.201 ±2.2e-4	0.203 ±4.8e-4	0.201 ±1.3e-4	0.202 ±2.9e-4	0.201 ±1.5e-4	0.201 ±2.2e-4	0.200 ±6.7e-5
PPO	0.202 ±2.6e-4	0.202 ±4.1e-4	0.201 ±1.8e-4	0.202 ±2.8e-4	0.201 ±2.8e-4	0.201 ±1.5e-4	0.201 ±2.5e-4	0.201 ±1.4e-4	0.202 ±3.0e-4	0.201 ±1.8e-4	0.201 ±1.4e-4	0.201 ±2.4e-4
WoS (ours)	**0.476** ±0.004	**0.357** ±0.004	**0.403** ±0.004	**0.386** ±0.004	**0.301** ±0.003	**0.506** ±0.004	**0.359** ±0.004	**0.288** ±0.003	**0.570** ±0.004	**0.484** ±0.005	**0.347** ±0.004	**0.222** ±0.002

Table 11: **Reward minus control cost** (↑). HM3D, conditioned goals. $\mathbb{E}[F(X_\tau)] - \frac{1}{2}\mathbb{E}\int_0^\tau \|a\|^2 dt$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	0.254 ±0.002	0.227 ±0.002	0.240 ±0.002	0.227 ±0.001	0.217 ±0.001	0.237 ±0.002	0.231 ±0.001	0.217 ±0.001	0.255 ±0.002	0.257 ±0.002	0.231 ±0.001	0.203 ±4.9e-4
REINFORCE	0.205 ±0.001	0.202 ±4.5e-4	0.201 ±1.9e-4	0.201 ±1.9e-4	0.201 ±2.9e-4	0.201 ±2.5e-4	0.202 ±3.4e-4	0.200 ±1.6e-4	0.214 ±0.001	0.201 ±2.8e-4	0.201 ±1.2e-4	0.200 ±4.5e-5
SAC	0.197 ±2.6e-4	0.201 ±4.2e-4	0.195 ±2.4e-4	0.199 ±2.4e-4	0.200 ±2.4e-4	0.199 ±2.2e-4	0.200 ±4.8e-4	0.200 ±1.3e-4	0.202 ±2.9e-4	0.200 ±1.5e-4	0.193 ±2.5e-4	0.181 ±4.3e-4
PPO	0.202 ±2.6e-4	0.202 ±4.1e-4	0.201 ±1.8e-4	0.197 ±3.2e-4	0.199 ±2.9e-4	0.201 ±1.5e-4	0.201 ±2.5e-4	0.201 ±1.4e-4	0.202 ±3.0e-4	0.201 ±1.8e-4	0.200 ±1.4e-4	0.191 ±3.8e-4
WoS (ours)	**0.335** ±0.004	**0.277** ±0.003	**0.287** ±0.004	**0.303** ±0.003	**0.262** ±0.003	**0.300** ±0.004	**0.304** ±0.003	**0.255** ±0.003	**0.407** ±0.004	**0.376** ±0.004	**0.284** ±0.003	**0.214** ±0.001

Table 12: **Success ratio** (↑). HM3D, conditioned goals. $\Pr[X_\tau \in \mathcal{G}]$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	0.110 ±0.003	0.048 ±0.002	0.074 ±0.003	0.053 ±0.002	0.033 ±0.002	0.108 ±0.003	0.054 ±0.002	0.027 ±0.002	0.145 ±0.003	0.105 ±0.003	0.053 ±0.002	0.005 ±0.001
REINFORCE	0.011 ±0.001	0.003 ±0.001	0.001 ±2.4e-4	0.001 ±2.4e-4	0.002 ±3.7e-4	0.002 ±3.2e-4	0.003 ±4.3e-4	0.001 ±1.9e-4	0.032 ±0.002	0.002 ±3.5e-4	0.001 ±1.5e-4	1.1e-4 ±5.5e-5
SAC	0.002 ±3.2e-4	0.002 ±0.001	0.001 ±2.7e-4	0.001 ±3.0e-4	0.001 ±3.0e-4	0.002 ±2.7e-4	0.003 ±0.001	0.001 ±1.6e-4	0.002 ±3.6e-4	0.001 ±1.9e-4	0.001 ±2.8e-4	2.1e-4 ±8.4e-5
PPO	0.002 ±3.2e-4	0.002 ±0.001	0.001 ±2.3e-4	0.002 ±3.5e-4	0.001 ±3.5e-4	0.001 ±1.9e-4	0.002 ±3.1e-4	0.001 ±1.8e-4	0.003 ±3.8e-4	0.001 ±2.2e-4	0.001 ±1.7e-4	0.001 ±3.0e-4
WoS (ours)	**0.344** ±0.005	**0.196** ±0.005	**0.253** ±0.005	**0.233** ±0.005	**0.126** ±0.004	**0.382** ±0.005	**0.199** ±0.005	**0.110** ±0.004	**0.463** ±0.005	**0.355** ±0.006	**0.184** ±0.005	**0.027** ±0.002

Table 13: **Average walk length, successful walks** (↓). HM3D, conditioned goals. $\mathbb{E}\sum_k \|X_{k+1} - X_k\|$ over walks with $X_\tau \in \mathcal{G}$.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	82.99	187.73	206.74	136.52	90.20	146.62	123.86	98.94	160.75	163.97	247.60	463.90
REINFORCE	35.92	51.06	41.46	35.53	18.36	55.47	31.84	25.80	84.93	32.24	65.72	172.26
SAC	16.84	**13.99**	31.03	**14.89**	**15.66**	**22.48**	**14.08**	32.84	25.54	49.12	20.39	**20.49**
PPO	25.86	20.69	30.45	182.36	31.87	36.26	27.57	32.81	25.43	34.74	46.39	460.68
WoS (ours)	**16.07**	28.68	**25.39**	22.67	16.68	25.93	15.39	**14.27**	**23.56**	**23.88**	**20.29**	38.72

Table 14: **Average walk steps, successful walks** (↓). HM3D, conditioned goals. steps taken over walks with $X_\tau \in \mathcal{G}$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	1641.6 ±23.2	3717.5 ±98.2	4093.8 ±81.0	2702.4 ±75.8	1785.5 ±63.5	2901.4 ±56.9	2452.7 ±54.6	1959.4 ±82.7	3181.6 ±49.3	3246.7 ±62.3	4903.9 ±106.0	9191.8 ±266.4
REINFORCE	712.2 ±31.0	1012.4 ±158.1	822.8 ±106.1	703.5 ±113.1	364.8 ±81.9	1100.3 ±102.0	630.3 ±107.7	511.2 ±70.5	1680.7 ±67.8	638.3 ±92.9	1299.8 ±188.5	3406.0 ±2194.4
SAC	336.2 ±46.1	279.1 ±74.3	619.2 ±85.3	298.1 ±59.3	313.1 ±77.0	449.4 ±56.1	280.9 ±54.1	657.9 ±99.9	509.9 ±67.8	981.9 ±120.4	405.3 ±68.4	410.8 ±96.6
PPO	512.5 ±62.3	410.1 ±84.5	604.0 ±110.8	3613.8 ±486.1	630.9 ±259.3	717.1 ±110.6	547.0 ±101.4	650.7 ±94.8	504.3 ±64.3	689.3 ±111.8	922.3 ±159.5	9125.5 ±747.9
WoS (ours)	**95.5** ±1.2	**90.9** ±1.6	**97.6** ±1.5	**93.4** ±1.5	**79.2** ±1.7	**109.4** ±1.3	**70.4** ±1.5	**69.0** ±1.9	**94.0** ±0.9	**92.6** ±1.4	**76.4** ±1.5	**91.7** ±5.0

Table 15: **Time per successful walk** (↓). HM3D, conditioned goals. ms, rollout wall-clock / number of successes.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	5.19	11.75	7.47	14.80	20.47	6.49	12.19	24.54	4.46	5.52	13.89	154.84
REINFORCE	17.15	88.13	199.98	181.11	134.88	118.55	108.26	273.36	9.53	145.89	670.06	6673.07
SAC	90.59	200.42	290.41	280.45	257.29	188.62	94.33	461.48	118.21	320.75	233.36	2038.67
PPO	117.20	188.07	375.59	322.22	533.37	363.44	293.41	465.22	130.79	225.48	730.26	563.07
WoS (ours)	**0.06**	**0.09**	**0.07**	**0.09**	**0.13**	**0.06**	**0.10**	**0.22**	**0.05**	**0.07**	**0.13**	**0.76**

Table 16: **Average reward** (↑). HM3D, fixed goal. $\mathbb{E}[F(X_\tau)]$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
MPPI	0.203 ±3.4e-4	0.204 ±0.001	0.204 ±3.5e-4	0.203 ±3.3e-4	0.201 ±2.6e-4	0.202 ±2.6e-4	0.206 ±0.001	0.202 ±2.8e-4	0.202 ±3.0e-4	0.204 ±3.8e-4	0.200 ±8.8e-5	0.201 ±2.8e-4
PICE	0.432 ±0.002	0.300 ±0.002	0.445 ±0.004	0.318 ±0.003	0.231 ±0.001	0.355 ±0.002	0.256 ±0.002	0.320 ±0.002	0.358 ±0.002	0.416 ±0.003	0.305 ±0.002	0.201 ±4.2e-4
REINFORCE	0.417 ±0.005	0.211 ±0.001	0.202 ±2.1e-4	0.244 ±0.002	0.203 ±0.001	0.408 ±0.005	0.221 ±0.002	0.208 ±0.001	0.209 ±0.001	0.266 ±0.003	0.200 ±9.4e-5	0.201 ±2.3e-4
SAC	0.262 ±0.003	0.202 ±4.3e-4	0.201 ±8.8e-5	0.241 ±0.003	0.201 ±3.3e-4	0.201 ±2.0e-4	0.223 ±0.002	0.201 ±1.2e-4	0.201 ±1.9e-4	0.205 ±4.7e-4	0.200 ±3.5e-5	0.200 ±2.1e-4
PPO	0.532 ±0.005	0.213 ±0.001	0.201 ±9.1e-5	0.259 ±0.003	0.202 ±4.2e-4	0.209 ±0.001	0.247 ±0.003	0.201 ±1.6e-4	0.216 ±0.002	0.519 ±0.005	0.200 ±3.0e-5	0.201 ±2.4e-4
WoS (ours)	**0.772** ±0.002	**0.586** ±0.004	**0.748** ±0.003	**0.655** ±0.004	**0.382** ±0.003	**0.642** ±0.003	**0.507** ±0.003	**0.607** ±0.003	**0.716** ±0.003	**0.766** ±0.003	**0.571** ±0.003	**0.254** ±0.002

Table 17: **Reward minus control cost** (↑). HM3D, fixed goal. $\mathbb{E}[F(X_\tau)] - \frac{1}{2}\mathbb{E}\int_0^\tau \|a\|^2 dt$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
MPPI	-4.469 ±0.049	-7.030 ±0.101	-7.632 ±0.096	-5.901 ±0.074	-5.094 ±0.061	-6.300 ±0.073	-5.653 ±0.084	-6.212 ±0.075	-5.807 ±0.062	-6.523 ±0.074	-9.571 ±0.143	-12.493 ±0.228
PICE	0.327 ±0.002	0.183 ±0.003	0.372 ±0.003	0.235 ±0.002	0.216 ±0.001	0.215 ±0.003	0.228 ±0.002	0.252 ±0.002	0.235 ±0.002	0.321 ±0.003	0.208 ±0.002	0.201 ±4.1e-4
REINFORCE	0.319 ±0.003	0.206 ±0.001	0.201 ±2.0e-4	0.232 ±0.002	0.202 ±0.001	0.318 ±0.003	0.215 ±0.001	0.205 ±0.001	0.207 ±0.001	0.248 ±0.003	0.200 ±9.1e-5	0.192 ±2.9e-4
SAC	0.237 ±0.002	0.202 ±4.3e-4	0.197 ±1.1e-4	0.218 ±0.002	0.199 ±3.2e-4	0.199 ±2.1e-4	0.205 ±0.001	0.199 ±1.2e-4	0.198 ±2.0e-4	0.199 ±4.6e-4	0.197 ±6.3e-5	0.189 ±3.0e-4
PPO	0.367 ±0.003	0.206 ±0.001	0.200 ±9.1e-5	0.240 ±0.002	0.201 ±4.1e-4	0.204 ±0.001	**0.229** ±0.002	0.201 ±1.6e-4	0.211 ±0.001	0.373 ±0.003	0.200 ±3.0e-5	0.200 ±2.5e-4
WoS (ours)	**0.555** ±0.003	**0.271** ±0.003	**0.544** ±0.005	**0.383** ±0.004	**0.225** ±0.003	**0.344** ±0.004	0.223 ±0.004	**0.343** ±0.004	**0.447** ±0.003	**0.507** ±0.004	**0.304** ±0.003	**0.226** ±0.002

Table 18: **Success ratio** (↑). HM3D, fixed goal. $\Pr[X_\tau \in \mathcal{G}]$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
MPPI	0.004 ±4.3e-4	0.005 ±0.001	0.005 ±4.4e-4	0.004 ±4.1e-4	0.002 ±3.2e-4	0.002 ±3.2e-4	0.007 ±0.001	0.003 ±3.5e-4	0.002 ±3.8e-4	0.005 ±4.8e-4	3.4e-4 ±1.1e-4	0.001 ±3.5e-4
PICE	0.289 ±0.003	0.125 ±0.003	0.306 ±0.005	0.147 ±0.003	0.038 ±0.002	0.194 ±0.003	0.071 ±0.002	0.150 ±0.003	0.198 ±0.002	0.270 ±0.004	0.131 ±0.002	0.002 ±0.001
REINFORCE	0.271 ±0.006	0.014 ±0.001	0.003 ±2.6e-4	0.055 ±0.003	0.004 ±0.001	0.259 ±0.006	0.027 ±0.002	0.010 ±0.001	0.011 ±0.001	0.083 ±0.004	3.8e-4 ±1.2e-4	0.001 ±2.9e-4
SAC	0.077 ±0.004	0.003 ±0.001	0.001 ±1.1e-4	0.052 ±0.003	0.001 ±4.2e-4	0.002 ±2.5e-4	0.028 ±0.002	0.001 ±1.5e-4	0.001 ±2.4e-4	0.006 ±0.001	9.2e-5 ±4.3e-5	0.001 ±2.6e-4
PPO	0.415 ±0.007	0.016 ±0.002	0.001 ±1.1e-4	0.073 ±0.004	0.002 ±0.001	0.012 ±0.001	0.059 ±0.003	0.001 ±2.0e-4	0.020 ±0.002	0.399 ±0.007	6.1e-5 ±3.7e-5	0.001 ±3.0e-4
WoS (ours)	**0.715** ±0.003	**0.482** ±0.005	**0.685** ±0.004	**0.568** ±0.005	**0.228** ±0.004	**0.553** ±0.004	**0.384** ±0.004	**0.509** ±0.004	**0.645** ±0.003	**0.708** ±0.003	**0.464** ±0.004	**0.067** ±0.003

Table 19: **Average walk length, successful walks** (↓). HM3D, fixed goal. $\mathbb{E}\sum_k \|X_{k+1} - X_k\|$ over walks with $X_\tau \in \mathcal{G}$.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	130.82	265.92	282.39	157.05	103.55	207.64	81.41	183.97	180.47	204.55	239.54	11.71
REINFORCE	70.01	28.83	129.50	70.65	7.54	87.02	43.44	26.24	**6.62**	40.26	18.67	**8.26**
SAC	**12.91**	**13.45**	149.92	**10.32**	7.55	29.83	**13.46**	**16.75**	18.43	41.44	17.20	9.06
PPO	60.15	14.29	77.57	35.77	**6.29**	**19.86**	15.57	20.52	10.62	98.63	**12.62**	8.54
WoS (ours)	21.33	56.34	**31.06**	32.16	29.23	38.87	36.64	32.70	31.59	**34.52**	37.02	58.27

Table 20: **Average walk steps, successful walks** (↓). HM3D, fixed goal. steps taken over walks with $X_\tau \in \mathcal{G}$. Entries are the mean ± its standard error over the start points, $s/\sqrt{M}$ with M the number of start points.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
PICE	2586.7 ±16.0	5260.0 ±61.3	5592.3 ±42.7	3106.5 ±48.6	2048.1 ±62.8	4107.3 ±45.0	1608.8 ±48.5	3639.5 ±48.3	3569.4 ±35.6	4048.6 ±45.8	4738.4 ±47.5	230.6 ±52.9
REINFORCE	1377.7 ±12.2	569.1 ±27.7	2566.2 ±172.6	1397.3 ±26.6	147.7 ±13.7	1714.4 ±25.5	857.8 ±40.8	518.8 ±34.2	**128.5** ±16.3	793.7 ±17.3	369.0 ±94.1	164.3 ±59.1
SAC	249.8 ±12.6	268.2 ±36.3	2994.1 ±453.4	196.7 ±7.1	149.6 ±28.3	595.9 ±82.9	259.4 ±16.8	334.7 ±89.3	368.5 ±52.1	827.6 ±43.2	346.7 ±193.7	181.8 ±77.2
PPO	1180.3 ±9.1	280.7 ±19.2	1538.4 ±195.6	703.8 ±12.8	**124.0** ±16.1	390.9 ±29.8	301.9 ±7.9	405.2 ±59.9	205.4 ±26.5	1943.6 ±22.7	247.8 ±64.9	170.7 ±49.3
WoS (ours)	**122.1** ±0.6	**179.8** ±1.2	**105.3** ±1.1	**143.2** ±1.1	134.6 ±1.7	**163.1** ±1.2	**165.8** ±1.5	**140.3** ±1.0	138.4 ±0.8	**130.1** ±1.1	**151.8** ±0.9	**92.9** ±1.7

Table 21: **Time per successful walk** (↓). HM3D, fixed goal. ms, rollout wall-clock / number of successes.

Method	00000	00103	00192	00247	00338	00387	00394	00398	00601	00609	00720	00792
MPPI	2561.74	3817.54	3396.88	4923.25	9766.35	6448.54	3542.58	7757.71	4231.82	2480.00	120253.75	67556.75
PICE	2.15	5.49	1.74	5.15	15.70	3.66	8.67	5.16	3.02	2.13	5.62	462.24
REINFORCE	0.61	15.82	100.59	4.31	64.71	0.86	10.50	23.80	17.26	2.10	1544.42	824.95
SAC	2.94	175.18	693.02	5.33	213.07	240.13	8.44	755.75	348.74	51.01	3667.41	395.72
PPO	0.58	21.42	579.93	8.32	166.61	33.08	7.52	445.73	13.07	0.87	9847.61	1033.71
WoS (ours)	**0.02**	**0.03**	**0.02**	**0.03**	**0.07**	**0.03**	**0.05**	**0.03**	**0.03**	**0.02**	**0.03**	**0.19**

J Defensive Sampling of the Policy During Training

Section 3.1 draws every walk from the vMF mixture $q_{\bar{\theta}}$, and each walk carries an importance weight (Appendix D) that divides by the density of the direction it sampled. A walk that wanders into a direction the policy gives low probability therefore returns a very large weight. The variance of our training targets can then explode. Huang et al. (2025) address this by drawing instead from a defensive mixture of the policy with the uniform density,

$$\tilde{p}_\theta(\nu \mid x) = s q_\theta(\nu \mid x) + (1 - s)\frac{1}{|\mathbb{S}^{D-1}|}, \qquad 0 < s \leq 1, \tag{36}$$

where the selection probability s is a hyper-parameter. The uniform component bounds the weights, at the cost of spending part of every rollout budget on undirected walks. Defensive sampling is common in rendering (Veach, 1998). Every estimator and gradient of Section 3.1 holds with $\tilde{p}_{\bar{\theta}}$ in place of $q_{\bar{\theta}}$.

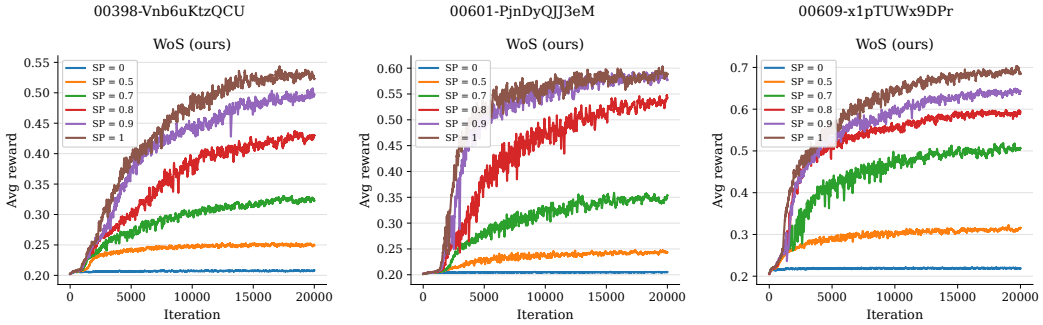


Figure 14: **Selection probability during training.** Average training reward against iterations on three HM3D scenes, one run per setting. Training draws each walk from the defensive mixture $\tilde{p}_{\bar{\theta}}$ of Equation 36; evaluation draws from the vMF mixture q_θ alone. SP in the panels is the selection probability s, and $s = 1$ removes the uniform component. Reward rises with s in all three scenes.

We swept s on three HM3D scenes (Figure 14). Training draws each walk from $\tilde{p}_{\bar{\theta}}$, and evaluation draws from q_θ alone. Average reward is ordered by s in all three scenes, and $s = 1$, the vMF mixture with no uniform component, is highest in each. We therefore set $s = 1$ in every experiment we report (Table 3).

We offer one hypothesis for this. The product of Huang et al. (2025) is the Monte Carlo estimate itself, so a single large weight lands in the answer. Our product is a trained policy, parameterized by a neural network and fitted over many walks, and a network averages the rare large weights away, the "fireflies" of rendering.

K Notation

Table 22: Symbols used throughout the paper, with the section that introduces each one.

Symbol	Meaning	Introduced
$\Omega, \partial\Omega, D$	bounded domain, its boundary, ambient dimension	Section 2
$\partial\Omega^\epsilon$	ϵ-shell $\{x \in \Omega : \mathrm{dist}(x, \partial\Omega) < \epsilon\}$	Section 2
$\mathrm{Proj}_{\partial\Omega}(Y)$	closest point to Y on $\partial\Omega$; $F(Y) := F(\mathrm{Proj}_{\partial\Omega}(Y))$	Section 2
F	terminal reward on $\partial\Omega$	Section 2.2
σ^2	policy noise, and the weight of the path-space KL term	Section 2.2
W_t	standard D-dimensional Wiener process	Section 2
W_t^σ	passive process σW_t, generator $\frac{\sigma^2}{2}\Delta$	Section 2
$\mathbb{P}_W, \tau_W$	path measure and exit time of the passive process	Section 2
X_t, π, τ	controlled diffusion, its path measure, its exit time	Section 2.2
V	value function of Equation 4	Section 2.2
h	Doob h-transform, $h = \exp(V/\sigma^2)$	Section 2.4
a^*	optimal control $\sigma^2 \nabla \log h$	Section 2.4
$B(x, R)$	sphere of radius R centered at x	Section 2
$P_B(x, y)$	sphere-exit kernel	Section 2
$p_B(\nu \mid x), p_B^h(\nu \mid x)$	directional densities on $\mathbb{S}^{D-1}$	Section 3.1
k, n	indices for walk steps and trajectories	Section 3.1
N, M	rollouts per starting point, starting points per iteration	Section 3.1
$x_{n,k}, R_{n,k}, \nu_{n,k}$	state, radius, direction at step k of trajectory n	Section 3.1
Y_n, L_n	exit point and number of steps of trajectory n	Section 3.1
q_θ	learned K-component vMF mixture	Section 3.1
$\tilde{p}_\theta$	defensive mixture of q_θ with the uniform density	Appendix J
$\bar{\theta}$	frozen parameters used for sampling	Section 3.1
s	defensive mixture weight	Appendix J
$\mathcal{V}(\nu; \omega, \kappa)$	von Mises–Fisher density, mean direction ω, concentration κ	Section 3.1
$z_n, m^\star$	log path weight of rollout n, and its maximum over rollouts	Section 3.1
Δt	time step of the Euler–Maruyama integrator	Section 4