LATENT SPACE PLANNING WITH SEQUENTIAL MONTE CARLO

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with learned world models often relies on trajectory optimization on latent rollouts of action sequences through differentiable dynamics. Stochastic variants inject noise into these updates to escape spurious basins, but unlike zero-order planners, they do not maintain a population. In contrast, sequential Monte Carlo (SMC) performs population-level inference, where importance weighting and resampling progressively steer a candidate population toward regions favored by the target distribution and can be combined with gradient frameworks. In this work, we propose latent-space planning as SMC, enabling inference through steering and optimization. Because the cost determines where the population is steered, we also study how costs influence planning, specifically, the (L_2) objective typically used by planners and a structured cost inspired by local progress measures from classical planners. Across five environments and two world models, we find that SMC improves goal-reaching success over zero-order, gradient-based, and stochastic gradient-based planners, with the effect of the cost structure becoming more pronounced in more complex settings. We find that SMC improves average goal-reaching success by upto 19, 23 and 33 percentage points over zero-order, gradient-based and stochastic gradient-based planners, respectively.

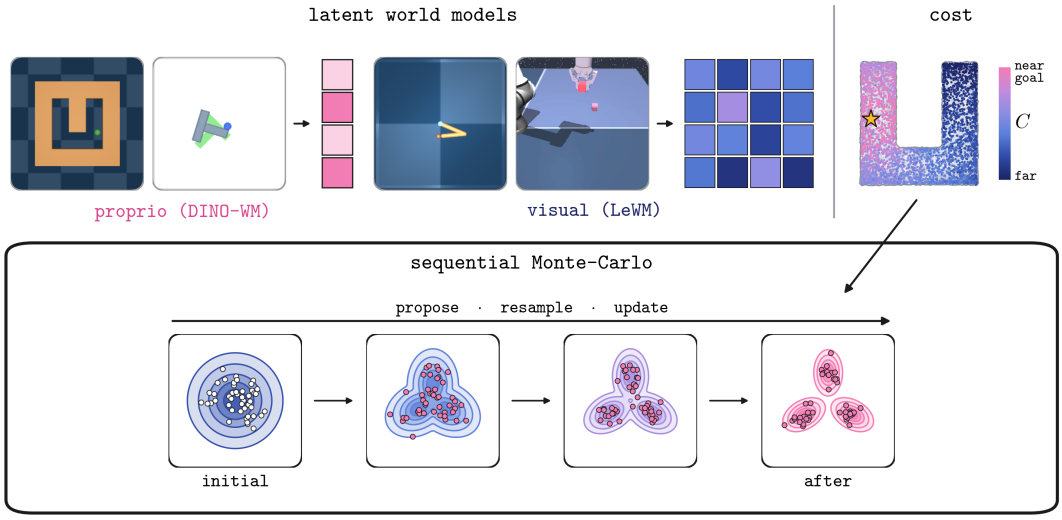

Figure 1: **Planning as sequential Monte Carlo over action sequences.** Top: Illustrating world model latent states in proprio and visual spaces and a cost formulation. Bottom: SMC progressively reallocates a population of action sequences toward low-cost regions through optimization, weighting, and resampling; the population concentrates on goal-reaching candidates.

1 INTRODUCTION

World models learn compact representations of environments from sensory observations and predict how those representations evolve under actions, enabling inexpensive simulation for tasks such as planning and navigation. Planning in these models is commonly performed by optimizing action

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.

sequences using zero-order methods such as the cross-entropy method (Zhou et al., 2025; Maes et al., 2026) or differentiable gradient-based optimization (Terver et al., 2025). Gradient-based approaches can optimize action sequences directly. However, while these methods leverage the differentiability of the models to optimize objectives defined over learned latent representations, they can become trapped in local minima. Recent parallelized stochastic variants such as GRASP (Psenka et al., 2026) introduce exploration by perturbing candidate trajectories, but unlike zero-order methods, these do not maintain a population, limiting the planner's ability to use information from the population as a whole: a population-based planner instead offers a natural mechanism for sharing this information, by selectively retaining and propagating candidates according to their progress toward the goal.

SMC (Schuster et al., 2017) provides a natural mechanism for such distributional steering of a population of candidate solutions, and has seen numerous applications in diffusion sampling (Chen et al., 2025; Neal, 1998; Millard et al., 2026; Wu et al., 2026; Young et al., 2026), reinforcement learning (Abdulsamad et al., 2026) and inference-time scaling (Yoon et al., 2026). Additionally, by viewing it within the lens of expectation-maximization, recent works show that it can be integrated into gradient-based frameworks(Cuin et al., 2026b;a). Thus, SMC may be viewed as a bridge for population-based steering and gradient optimization. In this work, we build on this idea for latent-space planning by treating action sequences as particles whose distributions can be progressively steered toward trajectories that are more favorable for reaching the goal.

SMC combines stochastic mutation with sequential importance weighting and resampling, progressively concentrating computational effort on candidates favored by the target distribution while retaining stochastic exploration. This target distribution is defined by a cost in the learned latent space, raising the question of what such a cost should represent. While latent-space planners define cost objectives over L_2 distances between latent states, we investigate whether a cost-to-go structure can provide a more informative signal for steering the population. In classical planning, cost-to-go functions are defined over the local topology of the state space and can be obtained by solving Partial Differential Equations (PDEs) that propagate goal information through this topology. A learned latent space, however, does not come with an explicit topology on which such a cost-to-go can be defined. We therefore construct a roadmap (Ni et al., 2026) from transitions in the offline dataset to recover local connectivity in the latent space, and define a PDE over this roadmap to obtain a cost-to-go structure. We use this cost-to-go as the potential defining the SMC target distribution, combining PDE-derived cost geometry with population-based stochastic exploration and optimization. We show that the proposed framework improves average goal-reaching success by up to 19, 23, and 33 percentage points over zero-order, gradient-based, and stochastic gradient-based planners, respectively. The PDE-derived cost provides additional gains on longer-horizon tasks and navigation environments, where capturing the structure of the planning space is particularly useful.

Thus, our main contributions are:

- We formulate inference-time planning in latent world models as a SMC problem, treating action sequences as particles and progressively steering their distribution toward trajectories favorable for goal reaching (fig. 1).
- We investigate how the choice of cost affects distributional steering in latent space by comparing the L_2 objective used by existing planners with a structured cost-to-go-like formulation inspired by classical planners.
- We evaluate both cost formulations within the SMC planner across five control environments and two learned world models, showing that SMC consistently improves goal-reaching success across a range of planning tasks.

2 Background

Latent world models. Planning from visual observations is often made tractable by learning a latent dynamics model and optimizing action sequences within the learned latent space. This paradigm underlies model-based reinforcement learning approaches such as Dreamer (Prasanna et al., 2024; Hafner et al., 2020; 2025) and TD-MPC (Hansen et al., 2024; 2022) and has more recently been extended to zero-shot settings and methods that plan entirely within learned latent representations. DINO-WM (Zhou et al., 2025), for example, predicts dynamics directly in the latent space, and similarly, joint-embedding predictive architectures (JEPAs) such as LeWM (Maes et al., 2026) represent each observation with a compact latent token and predict future representations in that space. These compact latent dynamics models make multi-step rollouts inexpensive and amenable to gradient-based action optimization. Consequently, the action sequence is typically optimized

against a latent-space objective, such as the L_2 distance between a predicted latent state and a goal representation. Recent methods further introduce stochasticity to improve exploration during this optimization process (Psenka et al., 2026). However, while these objectives help avoid local minima through stochastic exploration, they do not explicitly steer the candidate action sequences towards goal optimizing distributions.

Sequential Monte Carlo for planning, optimization, and inference-time sampling. SMC provides a general particle-based framework for maintaining and refining a population of candidate solutions through sequential weighting, resampling, and propagation (Doucet et al., 2001). This perspective has been adopted across a broad range of problems, including Bayesian and variational inference (Hirt & Dellaportas, 2019), stochastic optimization (Cuin et al., 2026b), and trajectory planning and control (Piché et al., 2018). More recent work has extended particle-based methods to inference-time sampling and steering (Yoon et al., 2026), including latent-variable models (Cuin et al., 2025) and discrete diffusion models (Ou et al., 2026), while interacting-particle methods have also been used for latent diffusion models (Wang et al., 2026). In parallel, Annealed Flow Transport (Arbel et al., 2021) augments SMC with learned transport maps between successive target distributions, and Sequential Controlled Langevin Diffusions (Chen et al., 2025) combines particle-based resampling with learned diffusion transport. These applications illustrate the flexibility of SMC as a population-based mechanism for steering candidate solutions toward a specified target. We adopt this perspective for latent-space planning, treating action sequences as particles and using the planning objective to steer their population toward trajectories that optimize over cost objectives.

Planning over cost objectives. Classical planning methods often derive cost structure from the geometry of the environment and characterize this structure through PDEs (Kimmel & Sethian, 2001; Mirebeau & Portegies, 2019). For example, minimum-time value functions are characterized by the Eikonal equation under isotropic dynamics. This formulation has been used in neural motion planning (Ni & Qureshi, 2022; 2023) and reinforcement learning (Giammarino & Qureshi, 2026; Giammarino et al., 2026; Viswanath et al., 2026a), where the Eikonal equation provides a physics-informed constraint for learning cost functions (Raissi et al., 2019; Viswanath et al., 2026c). Similarly, geodesic distance is connected to diffusion through Varadhan's formula (Crane et al., 2012), making it well suited to discrete graph- and mesh-based structures. In sparsely covered or long-horizon settings, roadmap-based methods construct graphs over known states to capture environment topology and use these structures to provide weak supervision (Viswanath et al., 2026b) for continuous value functions through PDE-based formulations (Ni et al., 2026). These approaches provide cost structures that account for how a state is situated within the broader environment. We extend this perspective to latent-space planning by investigating whether a roadmap constructed from encoded dataset states can provide a useful cost structure when the underlying environment is unavailable at inference time.

3 METHOD

Latent-space planning can be formulated as inference over distributions of action sequences. In the path-integral control framework (Williams et al., 2016; 2017), a cost function defines a target distribution that assigns greater probability to low-cost trajectories, and planning amounts to drawing action sequences from this distribution. This formulation can be viewed as a form of Expectation-Maximization that admits an iterative optimization procedure, alternating between estimating the action distribution under the current cost and updating the distribution toward lower-cost trajectories. Recent works have proposed SMC as a framework for the E-step with the M-step instantiated through first-order optimizers (Cuin et al., 2025; 2026a). We follow a similar approach and formulate planning as SMC with optimization.

3.1 PROBLEM SETUP

Let $\mathcal{Z} \subseteq \mathbb{R}^z$ be the latent space of a pretrained world model and $\mathcal{A} \subseteq \mathbb{R}^m$ the action space, with $\mathcal{U} \coloneqq \mathcal{A}^T$ the space of action sequences $\mathbf{a} = (a_0, \dots, a_{T-1})$. The world model $F_{\theta_{\mathrm{WM}}}$ is frozen and defines the rollout $\mathcal{F}^T_{\theta_{\mathrm{WM}}}(z_0, \mathbf{a}) = (z_1, \dots, z_T)$. Given an initial latent z_0 and a goal z_g, a cost $C(\mathbf{a}) \coloneqq C\big(\mathcal{F}^T_{\theta_{\mathrm{WM}}}(z_0, \mathbf{a})\big)$ scores a candidate by its rollout, and planning seeks a sequence of low cost. We define a target distribution $\pi(\mathbf{a})$ that can be denoted as,

$$\pi(\mathbf{a}) \propto q_0(\mathbf{a}) \exp\big(-C(\mathbf{a})\big), \tag{1}$$

where q_0 is an initial distribution over action sequences. Planning is then a problem of probabilistic inference where the desired trajectories are drawn from π, which concentrates on goal-reaching sequences.

3.2 Sequential Monte Carlo

To approximate this target, SMC constructs a sequence of intermediate proposals and unnormalized target distributions that progressively transforms the initial distribution into the target. Let (P_k) denote the target distribution and (Q_k) the proposal distribution at SMC step k, with $k = 0, \ldots, K$. We define a sequence of unnormalized intermediate target distributions $\{\gamma_k\}_{k=0}^K$ as

$$\gamma_k(\mathbf{a}) \propto q_0(\mathbf{a}) \exp\big(-C_k(\mathbf{a})\big), \tag{2}$$

such that $\gamma_0(\mathbf{a}) = q_0(\mathbf{a})$. The corresponding normalized target distributions are given by $P_k(\mathbf{a}) = \gamma_k(\mathbf{a}) / \int_{\mathcal{U}} \gamma_k(\mathbf{a}')\, d\mathbf{a}'$, with $P_0 = Q_0 = q_0$ and $P_K = \pi$. This is achieved by evolving a weighted population of N particles $\{\mathbf{a}^i, w^i\}_{i=1}^N$ over optimization steps $k = 0$ to K. The proposal path distribution is given by

$$Q_{0:K}(\mathbf{a}_{0:K}) = q_0(\mathbf{a}_0) \prod_{k=1}^{K} q_k(\mathbf{a}_k \mid \mathbf{a}_{k-1}), \tag{3}$$

while the corresponding target path distribution is

$$P_{0:K}(\mathbf{a}_{0:K}) \propto q_0(\mathbf{a}_0) \exp\big(-C_0(\mathbf{a}_0)\big) \prod_{k=1}^{K} q_k(\mathbf{a}_k \mid \mathbf{a}_{k-1}) \exp\big(-C_k(\mathbf{a}_k)\big). \tag{4}$$

The sequence is constructed such that $P_0 = Q_0$ and $P_K = \pi$. SMC initializes $\mathbf{a}_0^i \sim Q_0$ with weights $w_0^i \leftarrow \gamma_0(\mathbf{a}_0^i)/Q_0(\mathbf{a}_0^i)$ for $i = 1, \ldots, N$. At each subsequent step, particles are propagated according to the intermediate proposal Q_k and reweighted according to the change of measure between the intermediate target γ_k and proposal Q_k, followed by resampling according to the normalized weights.

3.3 Planning as Latent-Space SMC Optimization

An SMC sampler approximates $\{\pi_k\}_{k=0}^K$ with N weighted particles that are moved by Markov kernels, reweighted and resampled. Our particles $\mathbf{a}_k^{(i)} \in \mathcal{U}$, $i = 1, \ldots, N$, are complete action sequences, and the frozen world model maps each particle to its rollout. We draw $\mathbf{a}_0^{(i)} \sim q_0$ with unnormalized weights $W_0^{(i)} = 1$, the initial weight γ_0/q_0 of Cuin et al. (2026b, Sec. 3.2).

Analogous to the Langevin state updates that GRASP uses for exploration(Psenka et al., 2026), we propagate the particles $(q_k(\cdot \mid \cdot))$ through Langevin updates (Vempala & Wibisono, 2019). For particle i, with rollout $\mathbf{z}_{k-1}^{(i)} = \mathcal{F}_{\theta_{\mathrm{WM}}}^H(z_0, \mathbf{a}_{k-1}^{(i)})$ and gradient $\nabla\mathcal{L}_k^{(i)} := \nabla_{\mathbf{a}}\mathcal{L}\big(\mathbf{a}_{k-1}^{(i)}; \mathrm{sg}(\mathbf{z}_{k-1}^{(i)})\big)$,

$$\bar{\mathbf{a}}_k^{(i)} = \mathbf{a}_{k-1}^{(i)} - \eta\left(\nabla\mathcal{L}_k^{(i)}\right) + \sqrt{2\eta\Gamma}\,\varepsilon_k^{(i)}, \qquad \varepsilon_k^{(i)} \sim \mathcal{N}(0, I), \tag{5}$$

where $\mathrm{sg}(\cdot)$ is the stop-gradient operator of Psenka et al. (2026), which treats its argument as a constant during differentiation, and $\mathcal{L}$ is the planning objective: the L_2 distance between the final rolled-out latent and z_g, plus an action regularizer (weight 0.1, Table 2), with its gradient estimated by randomized smoothing. We take $q_0 = \mathcal{N}(0, I)$.

where η is the step size, Γ the Langevin temperature. The particle weights are updated as:

$$W_k^{(i)} = W_{k-1}^{(i)} \exp\big(-C(\mathbf{a}_{k-1}^{(i)})/\alpha\big), \tag{6}$$

and normalized as, $w_k^{(i)} = W_k^{(i)} / \sum_{j=1}^N W_k^{(j)}$. We track weight degeneracy with the effective sample size (ESS) (Cuin et al., 2026b, Eq. (12)),

$$\mathrm{ESS}_k = \frac{\big(\sum_{i=1}^N W_k^{(i)}\big)^2}{\sum_{i=1}^N \big(W_k^{(i)}\big)^2} = \frac{1}{\sum_{i=1}^N \big(w_k^{(i)}\big)^2}. \tag{7}$$

Algorithm 1 Latent-space planning as an SMC sampler; symbols as in Section 3.3. Lines with superscript (i) run for $i = 1, \dots, N$.

Require: initial latent z_0, goal latent z_g, frozen world model $F_{\theta_{\mathrm{WM}}}$, cost C, planning objective $\mathcal{L}$, initial distribution q_0, particles N, iterations K, step size η, Langevin temperature Γ, SMC temperature α, ESS fraction τ

$\mathbf{a}_0^{(i)} \sim q_0$; $W_0^{(i)} \leftarrow 1$;
for $k = 1, \dots, K$ **do**
  $\mathbf{z}_{k-1}^{(i)} \leftarrow \mathcal{F}^H_{\theta_{\mathrm{WM}}}(z_0, \mathbf{a}_{k-1}^{(i)})$ ▷ rollout
  $\nabla\mathcal{L}_k^{(i)} \leftarrow \nabla_{\mathbf{a}}\mathcal{L}\big(\mathbf{a}_{k-1}^{(i)}; \mathrm{sg}(\mathbf{z}_{k-1}^{(i)})\big)$ ▷ states stop-gradiented
  $\varepsilon_k^{(i)} \sim \mathcal{N}(0, I)$
  $\bar{\mathbf{a}}_k^{(i)} \leftarrow \mathbf{a}_{k-1}^{(i)} - \eta\,\big(\nabla\mathcal{L}_k^{(i)}\big) + \sqrt{2\eta\Gamma}\,\varepsilon_k^{(i)}$ ▷ move equation 5
  $W_k^{(i)} \leftarrow W_{k-1}^{(i)} \exp\big(-C(\mathbf{a}_{k-1}^{(i)})/\alpha\big)$ ▷ cost of $\mathbf{z}_{k-1}^{(i)}$; equation 6
  $w_k^{(i)} \leftarrow W_k^{(i)} / \sum_{j=1}^N W_k^{(j)}$; $\mathrm{ESS}_k \leftarrow 1/\sum_{j=1}^N \big(w_k^{(j)}\big)^2$ ▷ equation 7
  if $\mathrm{ESS}_k < \tau N$ **then**
    draw $I_k^{(1:N)}$ by systematic resampling with probabilities $w_k^{(1:N)}$
    $\bar{\mathbf{a}}_k^{(i)} \leftarrow \bar{\mathbf{a}}_k^{(I_k^{(i)})}$; $W_k^{(i)} \leftarrow 1$; $w_k^{(i)} \leftarrow 1/N$
  end if
  $\mathbf{a}_k^{(i)} \leftarrow \bar{\mathbf{a}}_k^{(i)}$
  $g_k \leftarrow \sum_{i=1}^N w_k^{(i)} \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{a}_k^{(i)}; \cdot)$ ▷ GD update
end for

When $\mathrm{ESS}_k < \tau N$, we draw ancestor indices $I_k^{(i)}$ by systematic resampling with probabilities $w_k^{(1:N)}$. This reallocates the finite planning population toward action sequences favored by the current target while maintaining multiple particles for further exploration.

In the optimization phase, we compute the weighted gradient $g_k = \sum_{i=1}^N w_k^{(i)} \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{a}_k^{(i)}; \cdot)$ and pass it to the optimizer for a gradient descent update. In the following subsection, we discuss cost formulations for C.

3.4 Defining Cost in Latent Spaces

We consider two notions of cost for planning in a learned latent space. The first is the direct L_2 objective, which provides the simplest measure of progress by comparing a candidate state to the goal in latent space. The second derives a structured cost from the geometry and connectivity of the latent planning space using a PDE. This formulation is motivated by classical planning, where cost functions incorporate the topology of the feasible state space.

A key challenge in constructing such a PDE-based cost is defining the domain over which the underlying structure is represented. In motion planning, this domain is typically determined by the feasible states of the environment and their connectivity. A learned world model, however, operates fully in a latent space and does not have direct access to the geometry or topology of the underlying environment. We therefore construct a *roadmap* from encoded dataset states, using their connectivity to provide the latent-space domain required by the PDE.

The valid offline dataset state $x \in \mathcal{S}$ is mapped to its corresponding latent representation $\xi = E_\theta(x)$, and $\xi \in \mathcal{Z}$, where E_θ denotes the world model's state encoder. We sample latent states from the offline dataset to serve as roadmap nodes. The state-action transitions in the dataset then provide the connectivity between these nodes, with an edge connecting latent states corresponding to observed valid transitions. The resulting roadmap therefore provides a discrete representation of the environment's topology within the learned latent space, allowing the PDE to be defined over the feasible regions induced by the offline data.

Definition 1 (Latent Roadmap)**.** *The latent roadmap is a graph $\mathcal{R} = (\mathcal{V}, \mathcal{E})$, where each vertex $\xi_i \in \mathcal{V}$ represents the latent encoding of an observed state, $\mathcal{V} = \{\xi_i\}_{i=1}^N$, $\xi_i = E_\theta(x_i)$, $x_i \in \mathcal{S}$. An edge $(\xi_i, \xi_j) \in \mathcal{E}$ represents local connectivity between two latent states induced by the observed transitions in the offline dataset. For each pair of connected vertices, we assign an affinity*

weight w_{ij} according to their local proximity, yielding the affinity matrix $W = [w_{ij}]$. We define the corresponding degree matrix as $D_{ii} = \sum_j w_{ij}$, and the Laplacian as $L = D - W$.

Let $\mathcal{M}$ denote the discrete manifold induced by the known latent states and their connectivity in the latent roadmap. Following recent work on PDEs over discrete manifolds, we define a PDE to characterize the reachability structure of the known latent states. Specifically, given a goal state $z_g \in \mathcal{M}$, we define

$$(\mathrm{Id} - t\Delta_{\mathcal{M}})\, u = f, \qquad u(z_g) = -1, \tag{8}$$

where $\Delta_{\mathcal{M}}$ denotes the Laplace–Beltrami operator on the latent manifold and $t > 0$ controls the extent of propagation over the manifold. In the discrete roadmap, the Laplace–Beltrami operator is approximated by the graph Laplacian $\Delta_{\mathcal{M}} \approx -L$, where $L = D - W$. The solution $u : \mathcal{M} \to \mathbb{R}$ therefore defines a scalar reachability field over the latent states, where $u(z)$ quantifies the cost of reaching the goal from state z, with $u(z_g) = -1$. Here $f = \delta_{z_g}$ is a unit source at the goal node. In practice, we solve for u using conjugate gradient. Leveraging the solution u, we can define C as follows:

$$C(z_t) = \begin{cases} 0, & z_t = z_g, \\ u(z_{t+1}) - u(z_t), & z_t \neq z_g. \end{cases} \tag{9}$$

The cost of a sequence is $C(\mathbf{a}) = \sum_{t=0}^{T-1} C(z_t)$ over its rollout.

4 Experiments

In this section, we evaluate our proposed planner by focusing on the following questions.

- (Q1) Does SMC planning plan better than standard Zero-order and gradient based planners (Section 4.2)?
- (Q2) How do the failure modes vary across trials (Section 4.3)?
- (Q3) Does the roadmap cost structure contribute anything beyond the L_2 objectives defined over learned latent states (Section 4.4)?

4.1 Setup

Environments and world models. We evaluate on five environments - Point maze, Push-T, Wall, Reacher and Cube. Reacher and Cube use LeWM/JEPA latents, in which each frame is encoded as a single 192-d token; these models expose no proprioceptive information and planning operates on the visual latent directly. Point maze, Push-T and Wall use DINO-WM, whose encoder produces 196 patch tokens per frame but which also exposes a 10-d proprioceptive latent, and it is on that latent that planning operates. The two settings therefore differ by roughly an order of magnitude in the dimension of the space being searched. The world model is frozen throughout; no component of any planner requires learning. We compare against 3 planners - cross-entropy method (CEM), gradient descent (GD) and GRASP. Hyperparameters are listed in Table 2, baseline configurations in Appendix C, and ablations over the particle count, resampling threshold and budget in Appendix D.

Protocol. Unless otherwise stated, all planners are evaluated on 150 optimization iterations. Each configuration runs 100 independent trials (seeds), with one episode executed per trial. Success is declared per the environment criteria for the environments, and roadmaps are built by sampling transition pairs from the offline dataset. Baselines use their published configurations.

Horizons. We measure the planning horizon in world-model steps, denoted by H. We evaluate horizons of $H \in \{5, 10, 15\}$, corresponding to the number of latent transitions predicted by the world model.

4.2 Quantitative Results

We begin by considering the tradeoffs of our proposed approach against other planners. Table 1 reports success rates across the shared grid, together with the mean number of optimizer iterations

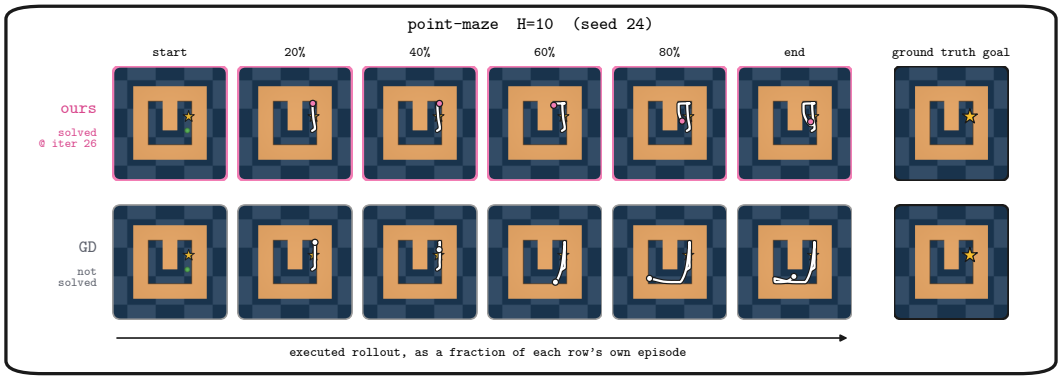

Figure 2: Point maze rollouts (seed 24). Ours solves in 26 iterations; GD reaches the cap. The goal frame is shown at the right.

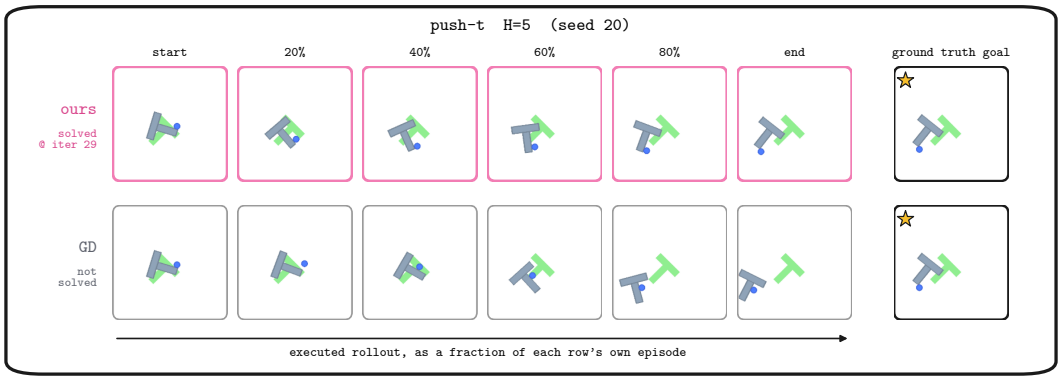

Figure 3: Push-T rollouts (seed 20). Ours solves in 29 iterations; GD reaches the cap.

required by successful runs. Across the environments, we find that our method generally trades optimization speed for a higher probability of finding a successful trajectory. We highlight this in fig. 4 and fig. 10. On Reacher, for example, our method reaches 100%, 92%, and 50% success at $H = 5, 10, 15$ compared with best-baseline values of 62%, 38%, and 32%. The corresponding successful runs require 27, 59, and 61 iterations on average, whereas CEM reaches its successful solutions in only 1–2 iterations. Thus, although successful CEM trajectories are found substantially earlier, its success rate saturates quickly; our population continues to improve over more optimization iterations, resulting in substantially higher final success. A similar pattern appears in the DINO-WM environments. On Point maze, our method achieves 75% and 81% success at $H = 5$ and $H = 10$, compared with 54% and 67% for the strongest baselines, while requiring 18 and 19 iterations on average. At $H = 15$, however, CEM achieves higher success (69% vs. 61%) and reaches successful trajectories in only 2 iterations. On Push-T, our method achieves the highest success at all three horizons (68%, 27%, and 5%), although success decreases sharply with horizon. Here again, the iteration counts increase with the difficulty of the problem, reaching 80 iterations at $H = 15$. We evaluate OGBench Cube (Park et al., 2025) using the evaluation protocol of its reference implementation: goals are drawn from held-out expert episodes, with 50 seeds, and success is defined as placing the block within 0.04m of the target. Our method achieves the highest success rate under both evaluation settings, reaching 88% across all seeds and 79% on the non-trivial subset, compared with 82/69% for gradient descent, 76/59% for the reference CEM solver, and 50/14% for GRASP. In this setting, successful trajectories are also found in relatively few iterations: both CEM and our method require 3 iterations on average, compared with 12 for GD and 18 for GRASP. We qualitatively visualize the trajectories for Point maze and Push-T in fig. 2, fig. 3 respectively.

Table 1: Planning success rate (%) with the mean number of optimiser iterations to success in parentheses, over 100 seeds per cell at a matched 150-iteration budget. Iteration counts average over successful seeds only. Best success rate per column in bold.

	Reacher			Point maze			Wall			Push-T		
Planner	5	10	15	5	10	15	5	10	15	5	10	15
GD	57 (10)	38 (14)	16 (18)	54 (22)	51 (12)	50 (22)	28 (29)	41 (34)	25 (21)	62 (22)	16 (32)	2 (24)
GRASP	27 (90)	33 (56)	32 (51)	44 (62)	49 (36)	62 (33)	21 (102)	9 (109)	5 (107)	37 (105)	4 (100)	0
CEM	62 (2)	38 (1)	28 (2)	51 (1)	67 (1)	**69** (2)	33 (3)	40 (3)	**49** (3)	45 (3)	4 (4)	2 (3)
Ours	**100** (27)	**92** (59)	**50** (61)	**75** (18)	**81** (19)	61 (26)	**56** (31)	**64** (23)	41 (26)	**68** (20)	**27** (46)	**5** (80)

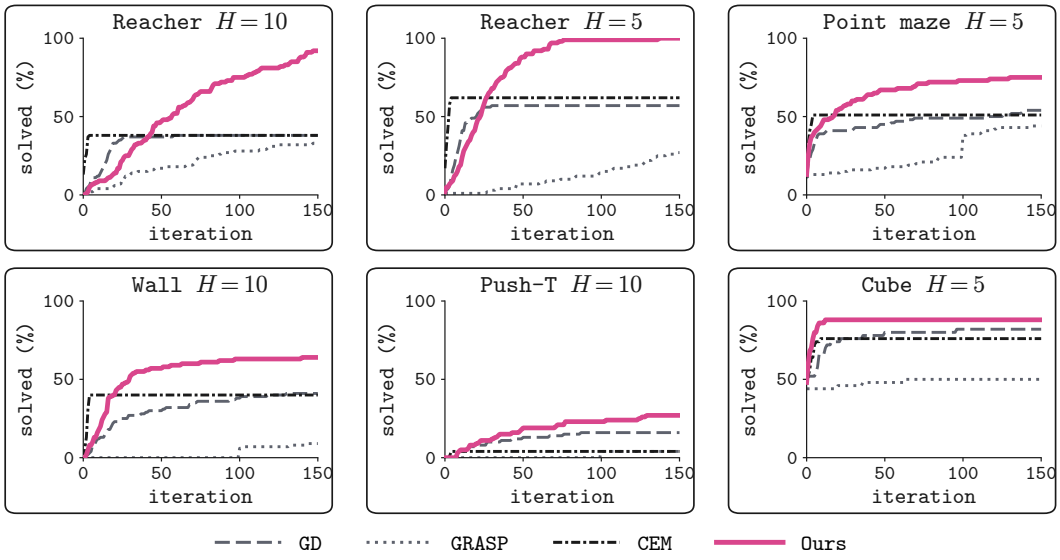

Figure 4: Fraction of seeds solved at or before each planner iteration, one panel per environment and horizon. Curves are monotone and their right-hand endpoints are the success rates of Table 1. We present additional settings in Appendix F. All are computed on latent cost-to-go.

4.3 Failure Modes

A success rate measures how often a planner succeeds, but not which instances it solves. Two planners can both achieve 50% success while agreeing on every seed or solving completely different seeds. Pairing the evaluations by seed distinguishes these cases. Figure 5 counts, for each baseline, the seeds that our method solves while the baseline does not, the reverse, aggregated over the four environments at each horizon; Figure 9 in Appendix E resolves the same counts by environment. Aggregated over the grid, our method wins on 350 seeds and loses on 70 against gradient descent, 444 and 47 against GRASP, and 344 and 112 against CEM. The losses are concentrated at the longest horizon: three cells have more baseline-only successes than ours, all at $H{=}15$ for Point maze against CEM (17 vs. 25) and GRASP (19 vs. 20), and Wall against CEM (16 vs. 24). These are the same cells in which CEM or GRASP achieves higher success in Table 1.

4.4 Impact of Latent Cost-to-go Objective

We quantitatively analyze the success rates against the two cost structures - L_2 and PDE-driven latent cost-to-go in Figure 6. At $H{=}5$, the two are nearly indistinguishable: four of the five environments differ by at most four percentage points, with a mean difference of -1.8 points. At $H{=}10$, the latent cost structure performs better in most environments, with an average advantage of 12 points, and the difference remains at $H{=}15$. However, at $H{=}5$, Push-T favors the L_2 objective by 16 points.

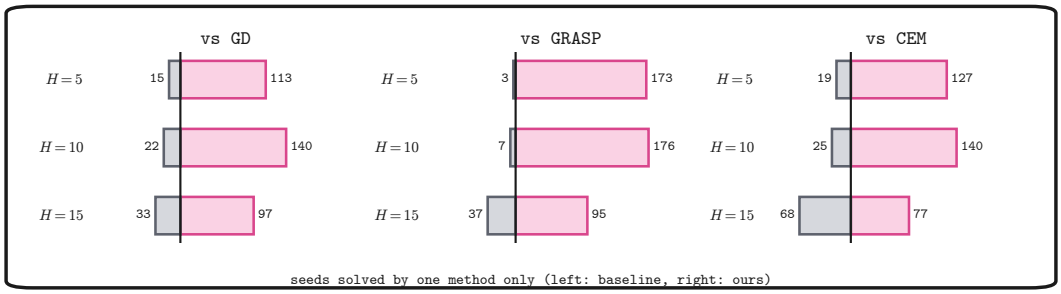

Figure 5: Paired outcomes against each baseline, summed over the four environments. Bars to the right count seeds our planner solves and the baseline does not; bars to the left count the reverse. Each bar aggregates 400 paired seeds. Per-environment panels are in Appendix E. All are computed on latent cost-to-go.

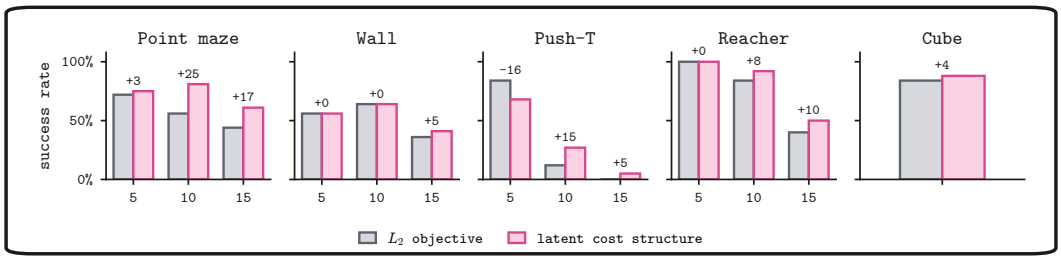

Figure 6: Contrasting the impact of the two cost structures: One panel per environment, horizons on the horizontal axis, with the difference above each pair.

5 Conclusion

We formulated planning with a learned world model as sequential Monte Carlo over action sequences, operating entirely within the model's latent space. By maintaining a population of candidate action sequences, the planner uses importance weighting, resampling, and stochastic propagation to progressively concentrate computation on trajectories favorable for goal reaching while retaining exploration. Across five environments and two learned world models, this population-based formulation improves goal-reaching success over zero-order, gradient-based, and stochastic gradient-based planners.

A central component of this formulation is the cost that defines the target distribution. We investigated two notions of progress toward a goal: the L_2 objective commonly used for latent-space planning and a structured cost-to-go-like formulation inspired by classical planning. For the latter, we construct a roadmap from transitions in the offline dataset to provide local connectivity in the learned latent space, and impose a PDE-based cost-to-go over this structure. This provides the SMC planner with a notion of progress that reflects the connectivity of the latent space rather than only the distance between individual latent states.

Our results show that SMC provides an effective framework for inference-time planning in learned world models, and that the structure used to define its target distribution is an important part of the planner. In particular, the PDE-based cost-to-go improves goal-reaching performance in maze navigation and longer-horizon settings, while our comparison with the L_2 objective shows that the benefits of population-based steering are not tied to a single choice of latent-space cost. More broadly, these results suggest that combining population-level distributional steering with structured cost geometry provides a useful approach to inference-time planning when the underlying environment is accessible only through a learned latent representation.

DISCLOSURE OF GENERATIVE AI USE

Generative AI tools (ChatGPT and Claude) assisted with manuscript drafting and revision, mathematical exposition and proof writing, interpretation of existing results, and LaTeX formatting, figure formatting and consistency checks. The authors are responsible for the final manuscript, including its mathematical claims, empirical reporting, and AI-assisted content.

REFERENCES

Hany Abdulsamad, Sahel Mohammad Iqbal, and Simo Sarkka. Sequential monte carlo for policy optimization in continuous pomdps. *Advances in Neural Information Processing Systems*, 38: 41241–41269, 2026.

Michael Arbel, Alexander G. D. G. Matthews, and Arnaud Doucet. Annealed flow transport monte carlo, 2021. URL https://arxiv.org/abs/2102.07501.

Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled langevin diffusions, 2025. URL https://arxiv.org/abs/2412.07081.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat, 2012. URL https://arxiv.org/abs/1204.6216.

James Cuin, Davide Carbone, and O. Deniz Akyildiz. Learning latent variable models via jarzynski-adjusted langevin algorithm, 2025. URL https://arxiv.org/abs/2505.18427.

James Cuin, Davide Carbone, and O Deniz Akyildiz. Learning latent variable models via jarzynski-adjusted langevin algorithm. *Advances in Neural Information Processing Systems*, 38:106837–106876, 2026a.

James Cuin, Davide Carbone, Yanbo Tang, and O. Deniz Akyildiz. Efficient stochastic optimisation via sequential monte carlo, 2026b. URL https://arxiv.org/abs/2601.22003.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Vittorio Giammarino and Ahmed Hussain Qureshi. Goal reaching with eikonal-constrained hierarchical quasimetric reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 59145–59182, 2026.

Vittorio Giammarino, Ruiqi Ni, and Ahmed Qureshi. Physics-informed value learner for offline goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 38:137769–137796, 2026.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Marcel Hirt and Petros Dellaportas. Scalable bayesian learning for state space models using variational inference with smc samplers. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 76–86. PMLR, 2019.

Kyurae Kim, Zuheng Xu, Jacob R. Gardner, and Trevor Campbell. Tuning sequential monte carlo samplers via greedy incremental divergence minimization, 2025. URL https://arxiv.org/abs/2503.15704.

Ron Kimmel and James A Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, 2001.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Andrew Millard, Joshua Murphy, Simon Maskell, and Zheng Zhao. Utilising gradient-based proposals within sequential monte carlo samplers for training of partial bayesian neural networks, 2026. URL https://arxiv.org/abs/2505.03797.

Jean-Marie Mirebeau and Jorg Portegies. Hamiltonian fast marching: A numerical solver for anisotropic and non-holonomic eikonal pdes. *Image processing on line*, 9:47–93, 2019.

Radford M. Neal. Annealed importance sampling, 1998. URL https://arxiv.org/abs/physics/9803008.

Ruiqi Ni and Ahmed H Qureshi. Ntfields: Neural time fields for physics-informed robot motion planning. *arXiv preprint arXiv:2210.00120*, 2022.

Ruiqi Ni and Ahmed H Qureshi. Progressive learning for physics-informed neural motion planning. *arXiv preprint arXiv:2306.00616*, 2023.

Ruiqi Ni, Yuchen Liu, and Ahmed H Qureshi. Weakly-supervised learning for physics-informed neural motion planning via sparse roadmap. *arXiv preprint arXiv:2604.13204*, 2026.

Zijing Ou, Chinmay Pani, and Yingzhen Li. Inference-time scaling of discrete diffusion models via importance weighting and optimal proposal design, 2026. URL https://arxiv.org/abs/2505.22524.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2018.

Sai Prasanna, Karim Farid, Raghu Rajan, and André Biedenkapp. Dreaming of many worlds: Learning contextual world models aids zero-shot generalization. *arXiv preprint arXiv:2403.10967*, 2024.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models, 2026. URL https://arxiv.org/abs/2602.00475.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ingmar Schuster, Heiko Strathmann, Brooks Paige, and Dino Sejdinovic. Kernel sequential monte carlo, 2017. URL https://arxiv.org/abs/1510.03105.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32, 2019.

Hrishikesh Viswanath, Juanwu Lu, S Talha Bukhari, Damon Conover, Ziran Wang, and Aniket Bera. Physics informed viscous value representations. *arXiv e-prints*, pp. arXiv–2602, 2026a.

Hrishikesh Viswanath, Hong Chul Nam, Xi Deng, Julius Berner, Anima Anandkumar, and Aniket Bera. Operator learning using weak supervision from walk-on-spheres. *arXiv preprint arXiv:2603.01193*, 2026b.

Hrishikesh Viswanath, Md Ashiqur Rahman, Abhijeet Vyas, Andrey Shor, Beatriz Medeiros, Stephanie Hernandez, Suhas Eswarappa Prameela, and Aniket Bera. Neural operator: Is data all you need to model the world? an insight into the paradigm of data-driven scientific ml. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026c.

Tim Y. J. Wang, Juan Kuntz, and O. Deniz Akyildiz. Training latent diffusion models with interacting particle algorithms, 2026. URL https://arxiv.org/abs/2505.12412.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 1433–1440. IEEE, 2016.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017.

Luhuan Wu, Yi Han, Christian Andersson Naesseth, and John Cunningham. Reverse diffusion sequential monte carlo samplers. *Advances in Neural Information Processing Systems*, 38:69445–69459, 2026.

Taehoon Yoon, Yunhong Min, Kyeongmin Yeo, and Minhyuk Sung. ψ-sampler: Initial particle sampling for smc-based inference-time reward alignment in score models. *Advances in Neural Information Processing Systems*, 38:104745–104781, 2026.

James Matthew Young, Paula Cordero-Encinar, Sebastian Reich, Andrew Duncan, and O Deniz Akyildiz. Diffusion path samplers via sequential monte carlo. *arXiv preprint arXiv:2601.21951*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

A Theoretical background

A.1 Target distribution and tempering

KL-regularized target From Yoon et al. (2026) Let p_0 be a reference distribution, r a reward and $\alpha \in \mathbb{R}_+$ the strength of the regularization. The distribution

$$p_0^* = \arg\max_q \ \mathbb{E}_{\mathbf{x}_0 \sim q}\big[r(\mathbf{x}_0)\big] - \alpha\, D_{\mathrm{KL}}\big(q \,\|\, p_0\big) \tag{10}$$

has the closed form $p_0^*(\mathbf{x}_0) = \frac{1}{Z_0}\, p_0(\mathbf{x}_0) \exp\big(r(\mathbf{x}_0)/\alpha\big)$, where Z_0 is the normalizing constant.

A.2 Feynman–Kac models and SMC samplers

Definition 2 (Feynman–Kac model; from Kim et al. (2025))**.** *Let $\mathcal{X}$ be a space with a σ-finite base measure, q the reference or initial proposal distribution, $(\mathsf{M}_k)_{k\in[K]}$ Markov kernels and $G_0 : \mathcal{X} \to \mathbb{R}_{\geq 0}$ and $(G_k)_{k\in[K]}$ non-negative Q-measurable functions, called potentials, where $[K] \coloneqq \{1, \dots, K\}$. The Feynman–Kac model describes a change of measure between the target and proposal path distributions*

$$\begin{aligned} P_{0:K}(\mathrm{d}x_{0:K}) &\coloneqq \frac{1}{Z_K}\Big\{G_0(x_0)\prod_{k=1}^{K} G_k(x_{k-1}, x_k)\Big\} Q_{0:K}(\mathrm{d}x_{0:K}), \\ Q_{0:K}(\mathrm{d}x_{0:K}) &\coloneqq q(\mathrm{d}x_0)\prod_{k=1}^{K} \mathsf{M}_k(x_{k-1}, \mathrm{d}x_k), \end{aligned} \tag{11}$$

with intermediate normalizing constants $Z_k = \int_{\mathcal{X}^{k+1}} G_0(x_0)\prod_{s=1}^{k} G_s(x_{s-1}, x_s)\, Q_{0:k}(\mathrm{d}x_{0:k})$, $k \in [K]$. Its SMC sampler draws N particles from q, each with weight 1; at each $k \in [K]$ it transports them with M_k, reweights them with G_k and optionally resamples them.

Definition 3 (Incremental importance weights; From Arbel et al. (2021))**.** *Let $\pi_k = \gamma_k/Z_k$, $k = 0, \dots, K$, with $Z_0 = 1$. Sample $X_0 \sim \pi_0$ and, for $k \geq 1$, $X_k \sim \mathsf{M}_k(X_{k-1}, \cdot)$ with a Markov kernel M_k, so that $X_{0:k}$ has density $\bar\eta_k(x_{0:k}) = \pi_0(x_0)\prod_{l=1}^{k} \mathsf{M}_l(x_{l-1}, x_l)$. For backward Markov kernels L_l, the extended target $\bar\pi_k(x_{0:k}) = \pi_k(x_k)\prod_{l=0}^{k-1} \mathsf{L}_l(x_{l+1}, x_l)$ has marginal π_k at time k by construction. The L_{k-1} are chosen so that the incremental importance weights*

$$G_k(x_{k-1}, x_k) = \frac{\gamma_k(x_k)\, \mathsf{L}_{k-1}(x_k, x_{k-1})}{\gamma_{k-1}(x_{k-1})\, \mathsf{M}_k(x_{k-1}, x_k)} \tag{12}$$

are well defined, and then $W_k(x_{0:k}) \coloneqq \bar\gamma_k(x_{0:k})/\bar\eta_k(x_{0:k}) = \prod_{l=1}^{k} G_l(x_{l-1}, x_l)$, where $\bar\gamma_k = Z_k\bar\pi_k$ is the unnormalized joint target.

Path measure targeted by an SMC sampler Let $\pi = \gamma/Z$ with $\gamma : \mathcal{X} \to \mathbb{R}_{>0}$, and let $(\pi_k)_{k=0}^K$, $\pi_k \propto \gamma_k$, be a path with $\pi_0 = q$, the initial proposal, and $\pi_K = \pi$. Let $(\mathsf{M}_k)_{k\in[K]}$ be forward and $(\mathsf{L}_{k-1})_{k\in[K]}$ backward Markov kernels such that

$$\pi_k \otimes \mathsf{L}_{k-1} \ll \pi_{k-1} \otimes \mathsf{M}_k \tag{13}$$

holds for every k, and set, for $k \geq 1$,

$$G_k(x_{k-1}, x_k) = \frac{Z_k}{Z_{k-1}} \frac{\mathrm{d}\big(\pi_k \otimes \mathsf{L}_{k-1}\big)}{\mathrm{d}\big(\pi_{k-1} \otimes \mathsf{M}_k\big)}(x_{k-1}, x_k), \tag{14}$$

which equals equation 12 when the Radon–Nikodym derivative can be evaluated pointwise. Then the Feynman–Kac model of Definition 2 with these kernels and potentials targets the path measure $P_{0:K}(\mathrm{d}x_{0:K}) = \pi(\mathrm{d}x_K) \prod_{k=1}^K \mathsf{L}_{k-1}(x_k, \mathrm{d}x_{k-1})$: the marginal of x_K is π, and for $k \in [K]$ the intermediate normalizing constant is $Z_k = \int_{\mathcal{X}} \gamma_k(x)\,\mathrm{d}x$.

A.3 Incremental weight of Algorithm 1

Proposition 1 (Algorithm 1 as a Feynman–Kac model)**.** *Let $q = q_0$, $G_0 \equiv 1$, M_k the move equation 5, and $G_k(x_{k-1}, x_k) = \exp\big(-C(x_{k-1})/\alpha\big)$ for $k \in [K]$. Then Algorithm 1 is the SMC sampler of the Feynman–Kac model of Definition 2 with these kernels and potentials, whose path measure is*

$$P_{0:K}(\mathrm{d}x_{0:K}) = \frac{1}{Z_K} \exp\Big(-\frac{1}{\alpha}\sum_{k=0}^{K-1} C(x_k)\Big)\, Q_{0:K}(\mathrm{d}x_{0:K}). \tag{15}$$

Proof. With $G_0 \equiv 1$, $\prod_{k=1}^K G_k(x_{k-1}, x_k) = \exp\big(-\frac{1}{\alpha}\sum_{k=1}^K C(x_{k-1})\big)$; substituting into equation 11 gives equation 15. Algorithm 1 draws $\mathbf{a}_0^{(i)} \sim q_0$ with unit weights, moves with M_k, multiplies by G_k and resamples when $\mathrm{ESS}_k < \tau N$, which is the SMC sampler of Definition 2. □

B Implementation details

The kernel block is held fixed across environments; only the particle count, the temperature and the budget vary. Values below are the ones used for every number in the paper.

Table 2: Planner settings. The kernel block is identical across environments by construction, so a per-environment row differs only in the roadmap graph, the particle count and α.

Group	Parameter	Value	Note
Kernel	step size η	0.1	,
	temperature Γ	10^{-3}	Langevin noise $\sqrt{2\eta\Gamma}$
	action regulariser	0.1	
	smoothing samples	8 / 1	LeWM / DINO-WM environments
	smoothing σ	1.0	annealed to 0
SMC	ESS fraction τ	0.5	resample threshold τN
	particles N	32 / 8 / 3 (2 at H=15)	Cube / Reacher / DINO-WM
	temperature α	0.1 / 1	Cube / all others
Roadmap	solver	heat	
	k-NN	64	
	graph nodes	3.8k / 12.6k / 55k	Point maze / Wall / others
	full-dimensional latents	yes	
Budget	iterations	150	matched across all planners

C Baseline configurations

All planners run at 150 iterations per environment step with evaluation after every iteration, and every arm sees the same world model, the same start and goal pairs and the same seeds.

Gradient descent (GD). Adam on the action sequence at step size 0.1, initialised from $\mathcal{N}(0, 1)$, backpropagating through the full T-step rollout.

GRASP. The published configuration: joint optimisation over actions and intermediate latent states with a quadratic penalty, state step size 0.1, action step size 10^{-3}, and the periodic full-gradient synchronisation its authors describe (`gd_interval`$=100$, `gd_opt_steps`$=25$).

CEM. Population 30 with 3 elites re-fitted for 5 outer iterations, which is 150 world-model rollouts per environment step and therefore matches the budget the gradient-based planners receive.

D Ablations

D.1 Particle count

Figure 7 sweeps $N \in \{1, 2, 4, 8, 16, 32\}$ on Reacher H=10 over the 20-seed stratified subset, with every other setting at its default. The sweep runs on Reacher rather than Point maze because DINO-WM encodes each frame as 196 ViT patch tokens and the population is capped at three particles there, so N=16 and N=32 are not expressible; Reacher is the LeWM path, one 192-d token per frame, and is also the environment on which the kernel was tuned.

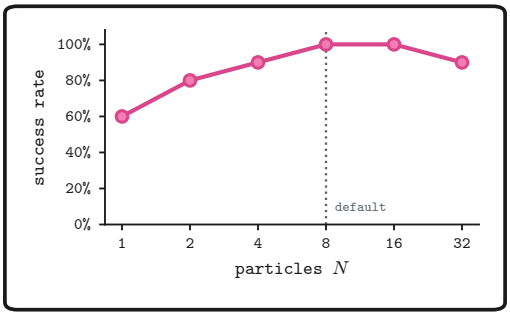

Figure 7: Success against population size on Reacher H=10, 20 stratified seeds, all other settings at their defaults. The dotted line marks the shipped N=8. Mean iterations to success over the same runs shows no trend in N (59, 75, 55, 79, 58, 64 for the six settings) and is not plotted.

D.2 Resampling threshold and budget

Figure 8 sweeps two SMC knobs on Point maze H=10, over the 20-seed stratified subset whose success rate reproduces the full 100-seed cell to within two points. Each sweep moves one knob and holds the rest at their defaults.

Budget. Success rises steeply out of a starved regime and then flattens: 45% at 25 iterations, 80% at 50, and 85% at 100 against the 80% of the 150 used throughout. The budget the paper runs at is thus on the flat part of the curve rather than at a peak. We do not sweep past 150 here, so this shows the comparison is not run at a starved budget, not that it is insensitive to a larger one.

E Per-seed comparisons

F Success against iteration, remaining panels

Figure 4 in the main text shows one panel per environment plus the largest remaining margin. The panels omitted there are collected here, so the full grid is available without selecting on outcome.

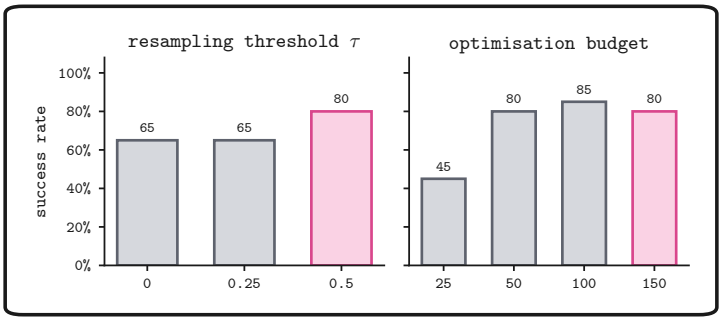

Figure 8: SMC knobs on Point maze $H{=}10$, 20 stratified seeds, one knob varied at a time.

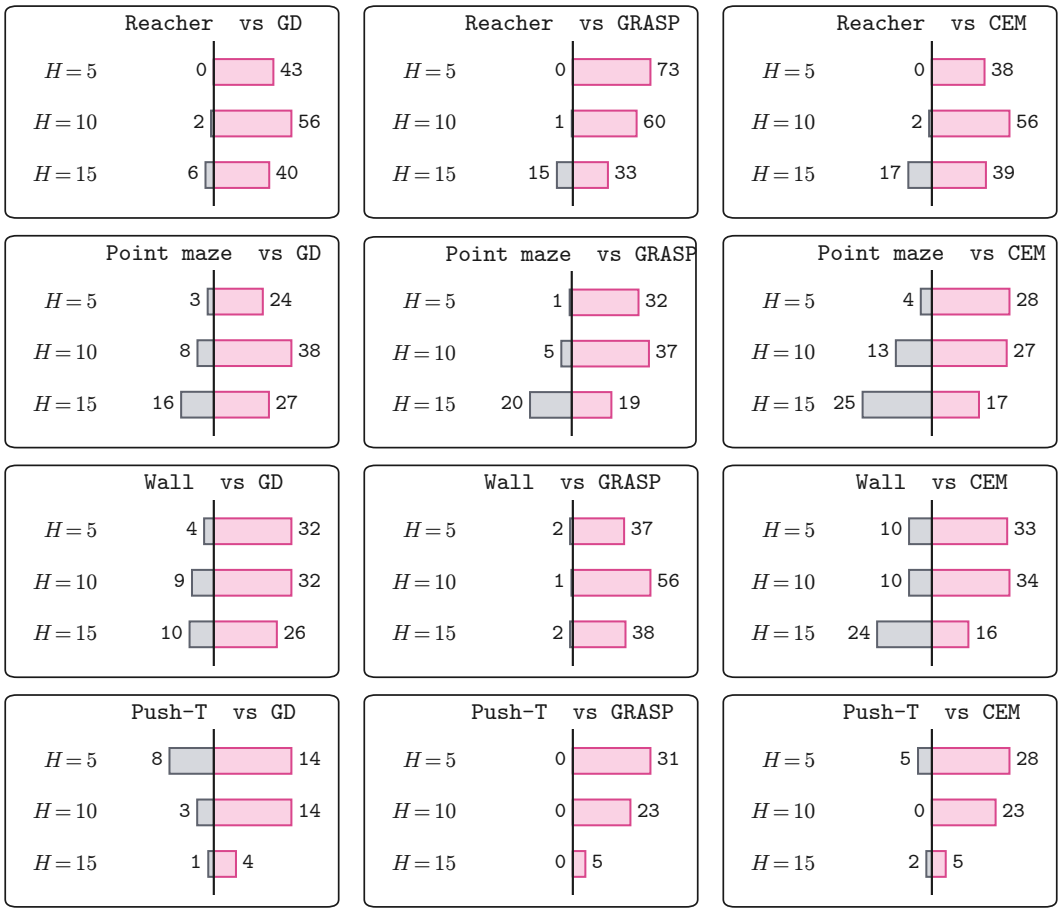

Figure 9: Per-seed paired outcomes, 100 seeds per cell, one panel per environment and baseline. Bars to the right count seeds we solve and the baseline does not; bars to the left count the reverse. Mirrored bars would mean the two methods trade instances; one-sided bars mean one solves a superset of the other.

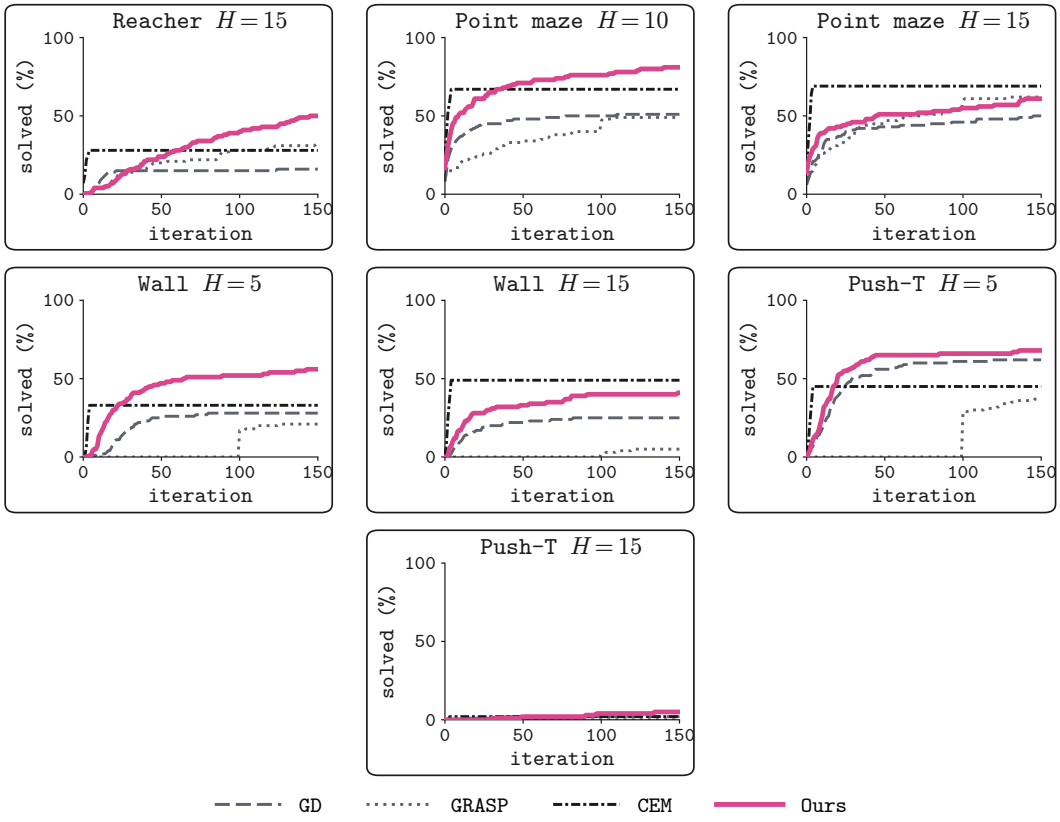

Figure 10: Fraction of seeds solved at or before each planner iteration, for the environment and horizon combinations not shown in Figure 4.