FAST WORLD PLANNER: ADVANTAGE-AWARE ACTION-CHUNK POLICY FOR ONLINE PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model Predictive Path Integral (MPPI) control is central to online planning for model-based reinforcement learning (MBRL), but its iterative refinement incurs planning latency. Reducing refinement iterations often degrades performance, motivating stronger policy guidance learned from the planner. However, existing methods typically learn from the planner through action-level alignment, which does not directly capture the complete H-step planner proposal. Moreover, planner imitation leaves further target improvement to online refinement. To address these limitations, we propose Fa**S**t **W**orld Pl**AN**ner (**SWAN**), an advantage-aware MPPI-based MBRL framework that explicitly represents and improves the complete H-step planner proposal to improve policy guidance for efficient online planning. For proposal representation, SWAN combines k-MPPI, a multimodal MPPI extension, with a mixture of factor analyzers (MFA) action-chunk policy to capture multiple modes and correlations across time and action dimensions. For proposal optimization, we introduce Advantage-Guided Proposal Optimization (APO), which reuses MPPI evaluations to construct an improved target using advantage estimates and trains the policy toward this target while preserving planner alignment. Experiments across 31 high-dimensional continuous-control tasks from DMControl, HumanoidBench, and MyoSuite show that SWAN substantially improves control performance and sample efficiency while using 50% fewer MPPI refinement iterations than state-of-the-art planning-based MBRL methods. Finally, SWAN demonstrates zero-shot sim-to-real transfer on a physical Unitree Go2, achieving stable velocity-tracking locomotion with high-frequency control. (see our demo at `https://swanfastworldplanner.github.io/`).

1 INTRODUCTION

Recent model-based reinforcement learning (MBRL) methods such as TD-MPC2 (Hansen et al., 2024) and BMPC (Wang et al., 2025a) combine learned world models and policies with Model Predictive Path Integral (MPPI) control, a sampling-based model predictive control (MPC) algorithm, to optimize action sequences online (Williams et al., 2017; Hansen et al., 2022; 2024; Wang et al., 2025a; Zhan et al., 2026). In these MPPI-based methods, the policy guides candidate generation and MPPI improves action sequences through iterative refinement. However, repeated trajectory evaluation introduces planning latency, while reducing refinement iterations leaves less opportunity to correct poor candidates (Williams et al., 2017; Pinneri et al., 2021). Prior work (Anthony et al., 2017; Silver et al., 2017) on expert iteration shows that policies learned from a planner can improve subsequent search. Thus, we seek better policy guidance through learning from the planner to improve control performance with fewer MPPI refinement iterations (Sacks & Boots, 2023).

A central obstacle to improving policy guidance is policy-planner mismatch between learned policies and planner-derived action distributions. Specifically, in prior critic-guided methods, the critic learns from planner-generated data but bootstraps with actions from the learned policy, while the resulting value estimates guide policy updates and provide terminal values for MPPI. Thus, policy-planner mismatch can lead to inaccurate value estimates, affecting both policy improvement and planning (Wang et al., 2025a; Serra-Gomez et al., 2026). To reduce the mismatch, recent MPPI-based methods either learn the policy through planner imitation (Wang et al., 2025a) or retain critic-guided updates with planner-derived regularization (Zhan et al., 2026; Serra-Gomez et al., 2026). However, two gaps remain in representing and improving the complete planner proposal. First, the *proposal*

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.

representation gap. Existing alignment objectives operate on individual actions or the planner's first-action distribution, rather than the complete H-step action-chunk distribution. Consequently, the learned policy does not directly learn the temporal correlations and multiple modes represented by the weighted action chunks. To examine the gap, we extend BMPC with our action-chunk proposal representation while retaining planner imitation, yielding BMPC-Chunk. As shown in Fig. 1 (left), BMPC-Chunk outperforms BMPC under the same MPPI refinement budget on *Dog-Run*, **motivating** direct representation of the complete planner proposal. Second, the *proposal optimization gap*. Even with an action-chunk representation, direct critic-guided policy updates remain susceptible to policy-planner mismatch. To reduce the mismatch, planner-derived regularization encourages planner alignment in critic-guided updates, but the trade-off with value maximization can leave residual mismatch. In contrast, planner imitation directly uses the current planner proposal as the learning target, leaving further target improvement to MPPI refinement. As shown in Fig. 1 (right), BMPC-Chunk remains sensitive to the number of refinement iterations, **motivating** additional proposal improvement in policy learning while preserving planner alignment.

Overcoming both gaps poses two technical **challenges**: i) efficiently represent the multimodal action-chunk planner proposal, and ii) optimize the proposal using advantage estimates while preserving planner alignment during policy learning. To address both challenges, we introduce Fa**S**t **W**orld Pl**AN**ner (SWAN), an advantage-aware MPPI-based MBRL framework that improves policy guidance through explicit representation and improvement of the complete H-step planner proposal. For proposal representation, SWAN combines k-MPPI, a multimodal extension of MPPI, with a **M**ixture of **F**actor **A**nalyzers (MFA) action-chunk policy. k-MPPI maintains multiple search modes through standard MPPI refinement within each mode, while the MFA policy models correlations across time and action dimensions and supports direct action-chunk sampling. For proposal optimization, SWAN introduces **A**dvantage-Guided **P**roposal **O**ptimization (APO). APO uses advantage estimates from MPPI trajectory evaluations to construct an improved target under a KL constraint relative to the current planner proposal. We then train the policy toward the improved target under a separate constraint that preserves planner alignment. Advantage information therefore enters policy learning through target construction rather than direct critic gradients through policy actions. Experiments on 31 high-dimensional continuous-control tasks across DMControl (Tunyasuvunakool et al., 2020), HumanoidBench (Sferrazza et al., 2024), and MyoSuite (Caggiano et al., 2022) show that SWAN improves control performance and sample efficiency with 50% fewer MPPI refinement iterations than the MPPI-based baselines. We further confirm the gains in Unitree G1 and Go2 simulations (Zakka et al., 2026), then demonstrate zero-shot sim-to-real transfer to a physical Go2 with stable velocity tracking at an average 308.16 Hz.

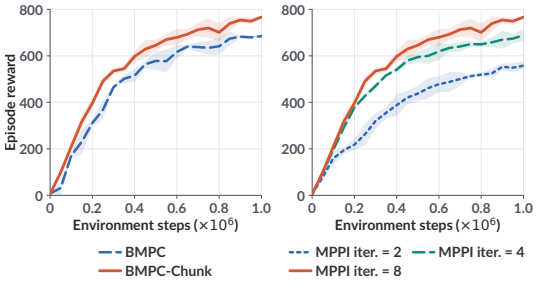


Figure 1: Motivation on *Dog-Run* task. BMPC-Chunk uses our action-chunk proposal representation with planner imitation. ***Left:*** Comparison between BMPC and BMPC-Chunk under 8 MPPI refinement iterations. ***Right:*** Comparison of BMPC-Chunk across 2, 4, and 8 MPPI refinement iterations. Candidate trajectory counts per iteration are fixed across settings. Curves show mean and 95% CIs over 3 seeds.

Our contributions are as follows. **1).** We propose SWAN, an advantage-aware MPPI-based MBRL framework that improves online planning efficiency through better policy guidance. **2).** We propose a novel approach that combines k-MPPI with an MFA policy to explicitly represent multiple proposal modes and correlations across time and action dimensions. **3).** We introduce APO to improve the planner proposal using advantage estimates while preserving planner alignment. Theoretically, we prove nonnegative expected H-step advantage of the exact APO target over the current planner proposal and derive a projection-aware sufficient condition for improvement over ideal planner imitation. **4).** We validate SWAN's control performance, sample efficiency, and online planning efficiency through extensive benchmark comparisons, controlled ablations, and robot locomotion simulations, and demonstrate zero-shot sim-to-real transfer to a physical Go2.

2 Related Work

Model-based Reinforcement Learning (MBRL). Recent MBRL learns world models to predict the consequences of actions for policy learning or planning (M. Moerland et al., 2023). Based on

how the learned world model is used, existing methods can be grouped into two categories: 1) imagination-based methods optimize policies from model-generated trajectories (Hafner et al., 2019a; 2020; 2023; Janner et al., 2019), 2) planning-based methods optimize action sequences online (Hafner et al., 2019b; Hansen et al., 2022; 2024). Within planning-based MBRL, TD-MPC and TD-MPC2 use MPPI as an online planner with learned policies guiding candidate generation (Hansen et al., 2022; 2024). To improve policy learning, subsequent MPPI-based methods reduce the mismatch between learned policies and planner-derived action distributions. BMPC (Wang et al., 2025a) learns a policy through imitation of the planner's first-action distribution. BOOM (Zhan et al., 2026) combines critic-guided policy updates with soft value-weighted alignment to planner actions stored in replay. PO-MPC (Serra-Gomez et al., 2026) learns an adaptive prior from the planner's first-action distribution to regularize policy optimization. However, these methods incorporate planner information through one-step action policy learning. In contrast, SWAN *directly* represents the complete H-step planner proposal and uses advantage estimates from MPPI trajectory evaluations to construct an improved target while preserving planner alignment.

Multimodal Action-Chunk Policies. Recent robot policies use diffusion and flow matching to *implicitly* model high-dimensional, multimodal action-chunk distributions (Chi et al., 2025; Chisari et al., 2024; Jiang et al., 2025). D-MPC (Zhou et al., 2025) also learns diffusion action proposals from offline trajectories for online MPC. Furthermore, for MPPI-based MBRL, nonlinear dynamics and non-convex control objectives can produce separated high-value regions in action-chunk space (Lambert et al., 2021; Honda et al., 2024). Return-weighted MPPI candidates can therefore contain multimodal structure, motivating diffusion or flow matching for learning the planner proposal. However, online proposal learning requires fast sampling and adaptation to changing planner proposals. Iterative generation introduces control latency (Wang et al., 2025b), while direct policy-gradient backpropagation through diffusion sampling can be *computationally expensive and unstable* (Ma et al., 2025). We thus combine k-MPPI with an MFA action-chunk policy to *explicitly* model the planner proposal, supporting direct sampling and capturing correlations across time and action dimensions.

Advantage-Guided Policy Learning. Prior model-free RL methods optimize policies using estimated advantages (Schulman et al., 2015; 2016; 2017). Subsequent work incorporates estimated advantages into policy learning by weighting actions from historical, replay, or offline data (Wang et al., 2018; Peng et al., 2019; Kostrikov et al., 2022). In this paper, we propose APO, which applies advantage estimates from MPPI trajectory evaluations to improve the current complete planner proposal, and then trains the policy toward the improved target under a separate planner-alignment constraint.

3 PRELIMINARIES

Problem Formulation. Reinforcement learning (RL) problems can be considered as an infinite-horizon discounted Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ denote the state and continuous action spaces, $\mathcal{P}$ is the transition kernel, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each discrete time step t, the agent observes s_t, samples action a_t from policy $\pi(\cdot \mid s_t)$, receives $r_t = \mathcal{R}(s_t, a_t)$, and transitions to s_{t+1} according to $\mathcal{P}$. The objective is to learn a policy π that maximizes $J(\pi) = \mathbb{E}_{\pi,\mathcal{P}}[\sum_{t=0}^{\infty} \gamma^t r_t]$. **Table 2 in Appendix A summarizes the notations used in this paper.**

MPPI-Based MBRL. MPPI-based MBRL combines learned world models and policies with MPPI to optimize action sequences online (Wang et al., 2025a). Following TD-MPC2 and BMPC (Hansen et al., 2024; Wang et al., 2025a), SWAN uses a latent world model comprising an encoder E_ψ, a dynamics model F_ψ, a reward model R_ψ, and a value model V_ϕ for terminal bootstrapping. The encoder gives $z_t = E_\psi(s_t) \in \mathbb{R}^{d_z}$, from which MPPI evaluates candidate action chunks.

At control step t, MPPI performs M refinement iterations over H-step action chunks $\tau = a_{t:t+H-1} \in \mathbb{R}^D$, where $D = Hd_a$. In prior methods, iteration $m = 1, \ldots, M$ forms a batch $\mathcal{T}^{(m)}$ of candidate chunks by combining samples from a diagonal Gaussian with chunks generated by rolling out the one-step action policy $\pi_{\theta_a}(\cdot \mid z_t)$ in the latent world model. For each candidate, MPPI rolls out $\widehat{z}_{h+1} = F_\psi(\widehat{z}_h, a_{t+h})$ from $\widehat{z}_0 = z_t$ and combines predicted rewards with the terminal value to estimate the H-step return

$$\widehat{G}_H(z_t, \tau) = \sum_{h=0}^{H-1} \gamma^h R_\psi(\widehat{z}_h, a_{t+h}) + \gamma^H V_\phi(\widehat{z}_H). \tag{1}$$

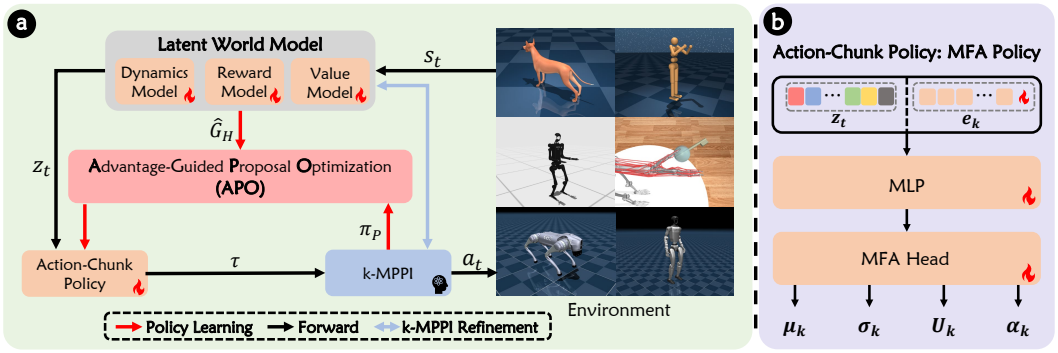


Figure 2: **Overview of the Proposed SWAN.** (a). The action-chunk policy initializes k-MPPI, which refines the proposal using predicted rewards and terminal value from the latent world model. APO constructs an improved target using advantage estimates and trains the policy toward the target while preserving planner alignment, providing initialization for subsequent planning. (b). The MFA policy models a multimodal distribution with correlations across time and action dimensions, enabling direct sampling of complete action chunks.

Based on the estimated returns, MPPI assigns each candidate a normalized weight $w^{(m)}(\tau) = \mathrm{softmax}_{\mathcal{T}^{(m)}}\big(\widehat{G}_H(z_t,\tau)/\eta_{\mathrm{MPPI}}\big)$, where $\eta_{\mathrm{MPPI}} > 0$ is the temperature and normalization is over $\mathcal{T}^{(m)}$. To continue refinement, MPPI projects the weighted empirical distribution onto a diagonal Gaussian by matching the weighted mean and elementwise variance

$$\mu^{(m+1)} = \sum_{\tau\in\mathcal{T}^{(m)}} w^{(m)}(\tau)\tau, \qquad (\sigma^{(m+1)})^2 = \sum_{\tau\in\mathcal{T}^{(m)}} w^{(m)}(\tau)\left(\tau - \mu^{(m+1)}\right)^2, \tag{2}$$

where $\mu^{(m)}, \sigma^{(m)} \in \mathbb{R}^D$ are the mean and element-wise standard deviation, and vector squares are element-wise. After M iterations, the final weighted candidates before Gaussian projection define the empirical planner proposal $\pi_P(\tau \mid z_t) = w^{(M)}(\tau)$ for $\tau \in \mathcal{T}^{(M)}$, with first-action marginal $\pi_{P,0}(a_t \mid z_t)$. Only the first planned action is executed before replanning.

Policy-Planner Mismatch. In critic-guided MPPI-based methods, the critic learns from planner-generated transitions in $\mathcal{B}$, with actions selected using $\pi_{P,0}$, but bootstraps with actions from π_{θ_a}. Since π_{θ_a} need not match $\pi_{P,0}$, the terminal value estimates used by MPPI may not reflect the return under subsequent replanning (Wang et al., 2025a; Serra-Gomez et al., 2026). To reduce the mismatch, BMPC (Wang et al., 2025a) learns π_{θ_a} through planner imitation, minimizing the Kullback–Leibler (KL) divergence $\mathrm{KL}(\pi^{\mathrm{G}}_{P,0} \| \pi_{\theta_a})$, where $\pi^{\mathrm{G}}_{P,0}$ is the Gaussian approximation to $\pi_{P,0}$. Alternatively, BOOM (Zhan et al., 2026) combines critic-guided policy updates with soft value-weighted alignment to planner actions in replay, while PO-MPC (Serra-Gomez et al., 2026) regularizes policy optimization toward an adaptive prior distilled from the planner's first-action distribution. *We omit the common conditioning on z_t in following KL expressions for simplicity.*

4 Proposed SWAN

Motivation. Although the methods such as BMPC and PO-MPC described in Sec. 3 reduce policy-planner mismatch, two gaps remain in learning the complete planner proposal π_P as follows.

1) Proposal representation gap. Existing alignment objectives operate at the action level, whereas the complete planner proposal factorizes as

$$\pi_P(\tau \mid z_t) = \pi_{P,0}(a_t \mid z_t)\pi_P(a_{t+1:t+H-1} \mid a_t, z_t). \tag{3}$$

Matching $\pi_{P,0}$ therefore does not determine the conditional distribution of the remaining actions.

2) Proposal optimization gap. Even with complete action chunks, critic-guided updates remain susceptible to mismatch since the temporal difference (TD) bootstrapping follows π_{θ_a}, which need not match $\pi_{P,0}$. To reduce the mismatch, planner-derived regularization balances planner alignment against value maximization. In contrast, planner imitation uses $\pi^{\mathrm{G}}_{P,0}$ as the learning target. However, extending imitation to complete chunks still uses the current π_P as the learning target, leaving further target improvement to MPPI refinement.

Proposed Method. To address both gaps, we introduce Fa**S**t **W**orld Pl**AN**ner (SWAN), an advantage-aware MPPI-based MBRL framework shown in Fig. 2. The proposed method combines k-MPPI and

an MFA action-chunk policy for complete proposal representation with APO to improve the learning target while preserving planner alignment, thereby improving policy guidance for efficient online planning. We detail the two novel components below.

4.1 Multimodal Action-Chunk Proposal Representation

In this section, we introduce a novel approach to explicitly representing the complete H-step planner proposal. Motivated by the diagonal Gaussian projection in Eq. 2, which cannot capture multiple modes or correlations across time and action dimensions, we propose k-MPPI to preserve multiple search modes and the MFA action-chunk policy to model the planner proposal.

k-MPPI. k-MPPI initializes K local Gaussian search distributions from the corresponding policy components and maintains a global Gaussian for exploration. For each refinement iteration, both local and global Gaussians are updated from their respective elites using Eq. 2 under a fixed total sampling budget. After M refinement iterations, the local and global search results form a proposal with K modes, each represented by a batch $\mathcal{E}_k$ of N elite action chunks. To obtain candidate weights and mode probabilities, we concatenate the batches into $\mathcal{E}$ and obtain

$$w^P(\tau) = \mathrm{softmax}_{\mathcal{E}}\left(\widehat{G}_H(z_t,\tau)/\eta_{\mathrm{MPPI}}\right), \qquad \alpha_k^P = \sum_{\tau\in\mathcal{E}_k} w^P(\tau), \tag{4}$$

where $w^P(\tau)$ is the normalized weight of candidate τ, α_k^P is the probability of mode k, and superscript P denotes planner quantities. The retained chunks and their weights define the empirical planner proposal π_P, with branch labels providing the mode assignments for policy learning. Appendix B.1 provides the complete k-MPPI algorithm.

MFA action-chunk policy (Fig. 2 (b)). In order to learn the proposal constructed by k-MPPI, we associate each retained mode with a Gaussian component, following the Gaussian projection in Eq. 2. We therefore parameterize π_θ as a mixture of factor analyzers (MFA) (Ghahramani et al., 1996), using a low-rank covariance term and diagonal residual variances to *capture correlations across time and action dimensions* without parameterizing full covariance matrices. The policy network with parameters θ uses z_t and learnable mode embeddings e_k to predict the MFA parameters $\{(\alpha_k^\theta, \mu_k^\theta, U_k^\theta, \sigma_k^\theta)\}_{k=1}^K$. Omitting their dependence on z_t, the policy is

$$\pi_\theta(\tau \mid z_t) = \sum_{k=1}^K \alpha_k^\theta \mathcal{N}\big(\tau \mid \mu_k^\theta, \Sigma_k^\theta\big), \qquad \Sigma_k^\theta = U_k^\theta (U_k^\theta)^\top + \mathrm{Diag}\big((\sigma_k^\theta)^2\big), \tag{5}$$

where $\alpha_k^\theta \geq 0$ is the component probability with $\sum_k \alpha_k^\theta = 1$, $\mu_k^\theta \in \mathbb{R}^D$ is the mean, $U_k^\theta \in \mathbb{R}^{D\times r}$ is the factor loading matrix with rank parameter $r \ll D$, and $\sigma_k^\theta \in \mathbb{R}^D$ contains residual standard deviations. We select k with probability α_k^θ and sample an action chunk without policy rollouts.

To support alignment with the action-chunk policy, we project the empirical planner proposal π_P onto the same MFA family, extending the Gaussian projection in Eq. 2. The projection preserves the mode probabilities α_k^P and produces the parameters $\{(\alpha_k^P, \mu_k^P, U_k^P, \sigma_k^P)\}_{k=1}^K$, which define the continuous MFA approximation $\widetilde{\pi}_P$,

$$\widetilde{\pi}_P(\tau \mid z_t) = \sum_{k=1}^K \alpha_k^P \mathcal{N}\big(\tau \mid \mu_k^P, \Sigma_k^P\big), \qquad \Sigma_k^P = U_k^P (U_k^P)^\top + \mathrm{Diag}\big((\sigma_k^P)^2\big). \tag{6}$$

We also detail the projection and sampling procedures in Appendix B.2 and B.3.

4.2 Advantage-Guided Proposal Optimization

After establishing the complete proposal representation in Sec. 4.1, we introduce APO to improve the planner proposal with advantage estimates while preserving planner alignment.

Advantage-guided proposal improvement. To improve the current planner proposal π_P, we seek an updated proposal π that maximizes $\mathbb{E}_{\tau\sim\pi}[\widehat{G}_H(z_t,\tau)]$. Since improvement is measured relative to π_P, we express the expected return gain as $\mathbb{E}_{\tau\sim\pi}[A(\tau)]$, where $A(\tau) = \widehat{G}_H(z_t,\tau) - \mathbb{E}_{\tau'\sim\pi_P}[\widehat{G}_H(z_t,\tau')]$.

However, advantage maximization alone cannot constrain how far π departs from π_P. We therefore impose a KL trust region on the proposal update at a given latent state z_t and solve

$$\max_{\pi} \quad \mathbb{E}_{\tau\sim\pi}[\bar{A}(\tau)] \quad \text{s.t.} \quad \mathrm{KL}(\pi\|\pi_P) \leq \epsilon_{\mathrm{MD}}, \tag{7}$$

where ϵ_{MD} is the KL radius and $\bar{A}(\tau) = A(\tau)/(s_A \sum_{h=0}^{H-1} \gamma^h)$ is the advantage normalized to account for reward scale, with s_A denoting the weighted standard deviation of the horizon-normalized advantage under the current π_P. Then we derive the proposal update by introducing a Lagrange multiplier $\lambda \geq 0$ for the KL constraint and forming

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_{\tau\sim\pi}[\bar{A}(\tau)] - \lambda \left(\mathrm{KL}(\pi \| \pi_P) - \epsilon_{\mathrm{MD}}\right). \tag{8}$$

For fixed $\lambda > 0$, removing the constant $\lambda\epsilon_{\mathrm{MD}}$ leaves a KL-regularized advantage objective, corresponding to a mirror-descent update from π_P (Beck & Teboulle, 2003). Thus, maximizing the Lagrangian over normalized distributions yields the optimal target π^*,

$$\pi^*(\tau \mid z_t) = \frac{\pi_P(\tau \mid z_t) \exp\big(\bar{A}(\tau)/\lambda\big)}{\mathbb{E}_{\tau'\sim\pi_P}\left[\exp\big(\bar{A}(\tau')/\lambda\big)\right]}. \tag{9}$$

The target reuses MPPI evaluations to favor advantageous action chunks without additional model rollouts. With $\lambda > 0$ fixed in implementation, π^* also solves Eq. 7 with the induced KL radius $\epsilon_{\mathrm{MD}} = \mathrm{KL}(\pi^* \| \pi_P)$, as shown in Appendix C.2.

Planner alignment. After obtaining the optimal target π^*, we learn π_θ to incorporate the proposal improvement into subsequent planning. For policy learning, we construct the MFA approximation $\widetilde{\pi}^*$ of π^* using the projection in Sec. 4.1. However, the trust region in Eq. 7 constrains π^* rather than π_θ. We thus introduce a separate forward-KL trust region around $\widetilde{\pi}_P$ to preserve planner alignment and optimize the policy parameters over replay states s_t,

$$\min_\theta \quad \mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}^* \| \pi_\theta)\right] \quad \text{s.t.} \qquad \mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta)\right] \leq \epsilon, \tag{10}$$

where ϵ bounds the average alignment KL. The MFA KL terms retain branch labels, as defined in Appendix B.4. To optimize the constrained policy objective, we introduce a Lagrange multiplier $\beta \geq 0$ for the alignment constraint and form

$$\mathcal{L}_{\mathrm{APO}}(\theta, \beta) = \mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}^* \| \pi_\theta)\right] + \beta \left(\mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta)\right] - \epsilon\right). \tag{11}$$

Rather than fixing β, we alternate policy minimization with projected dual ascent, adapting the penalty to constraint violation. Since the unnormalized dual update depends on the absolute KL scale, we use an exponential moving average (EMA) of the alignment KL as the current budget and normalize the update accordingly

$$\beta \leftarrow \left[\beta + \eta_\beta \left(\frac{\mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta)\right]}{\epsilon} - 1\right)\right]_+, \quad \epsilon = \left[\mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta)\right]\right]_{\mathrm{EMA}}, \tag{12}$$

where η_β is the dual step size, $[\cdot]_{\mathrm{EMA}}$ denotes the exponential moving average, and $[\cdot]_+$ projects onto nonnegative values. The current budget ϵ is held fixed during each policy and dual update. Further, Appendix C.4 presents a **unified view** that derives the core policy learning objectives of APO and prior methods from a common formulation, clarifying their connections and differences.

4.3 Theoretical Analysis

We further theoretically analyze proposal improvement at z_t with the latent world model held fixed and define $\widehat{J}_H(\pi) = \mathbb{E}_{\tau\sim\pi}[\widehat{G}_H(z_t, \tau)]$.

Theorem 1 (Exact target improvement)**.** *For $\lambda > 0$, the optimal target in Eq. 9 satisfies*

$$\widehat{J}_H(\pi^*) - \widehat{J}_H(\pi_P) = \mathbb{E}_{\tau\sim\pi^*}[A(\tau)] \geq 0. \tag{13}$$

Transferring the target gain to π_θ requires MFA projection and policy learning. We thus bound the resulting improvement over ideal planner imitation, defined by $\pi_{\mathrm{IL}} = \widetilde{\pi}_P$.

Theorem 2 (Projection-aware policy improvement)**.** *Assume the expected returns and KL divergence below are finite and there exists a finite constant $A_{\max} > 0$ such that $|\widehat{G}_H(z_t, \tau) - \widehat{J}_H(\widetilde{\pi}_P)| \leq A_{\max}$ for every τ in the union of the supports of π_θ, $\widetilde{\pi}^*$, and $\widetilde{\pi}_P$. Then, for any $\beta \geq 0$,*

$$\widehat{J}_H(\pi_\theta) - \widehat{J}_H(\pi_{\mathrm{IL}}) \geq \frac{\widehat{J}_H(\widetilde{\pi}^*) - \widehat{J}_H(\widetilde{\pi}_P)}{1+\beta} - A_{\max}\sqrt{\frac{2\left[\mathrm{KL}(\widetilde{\pi}^* \| \pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta)\right]}{1+\beta}}. \tag{14}$$

Together, the above theorems establish nonnegative target improvement for APO and a sufficient condition for improvement over ideal planner imitation after MFA projection and policy learning. We also provide the proofs of both theorems in Appendix C.3.

5 EXPERIMENTS

We first evaluate SWAN in terms of *(i)* control performance, sample efficiency, and online planning efficiency, *(ii)* the benefit of complete H-step planner proposal representation with k-MPPI and MFA under matched planning budgets, and *(iii)* whether APO improves control performance over planner imitation under reduced MPPI refinement budgets. We further evaluate SWAN on Unitree G1 and Go2 in simulation, then deploy it on a physical Go2 robot for real-world validation.

Benchmarks. We evaluate 31 tasks comprising 7 from DMControl (Tunyasuvunakool et al., 2020), 14 from HumanoidBench (Sferrazza et al., 2024), and 10 from MyoSuite (Caggiano et al., 2022), spanning locomotion and manipulation with maximum action dimensions of $38, 61, 39$, respectively. Additional Unitree G1 and Go2 tasks (Zakka et al., 2026) have 29 and 12 action dimensions. Appendix D lists all tasks and descriptions.

Baselines. We compare against planning-based BMPC (Wang et al., 2025a), BOOM (Zhan et al., 2026), and PO-MPC (Serra-Gomez et al., 2026), imagination-based DreamerV3 (Hafner et al., 2023), and model-free FPO (McAllister et al., 2026) and SAC (Haarnoja et al., 2018). For action-chunk comparison, *BMPC-DP* replaces BMPC's one-step policy with a diffusion policy (Chi et al., 2025), retaining MPPI and learning the complete H-step planner proposal through planner imitation.

Implementation Details. We match environment steps across methods. In the **full-budget** comparison, MPPI-based baselines use $M = 8$ refinement iterations, while SWAN uses $M = 4$. In the **half-budget** comparison, all MPPI-based methods use $M = 4$. We provide detailed implementation settings and hyperparameters in Appendix E.

5.1 MAIN RESULTS

Control performance and sample efficiency. As shown in Figs. 3, 4, and 5, we compare SWAN's control performance and sample efficiency with baselines under full- and half-budget settings. Each figure plots means over environment steps for the suite average and four representative tasks, with shaded 95% confidence intervals (CIs) over 3 random seeds. We provide all task curves in Appendix F.

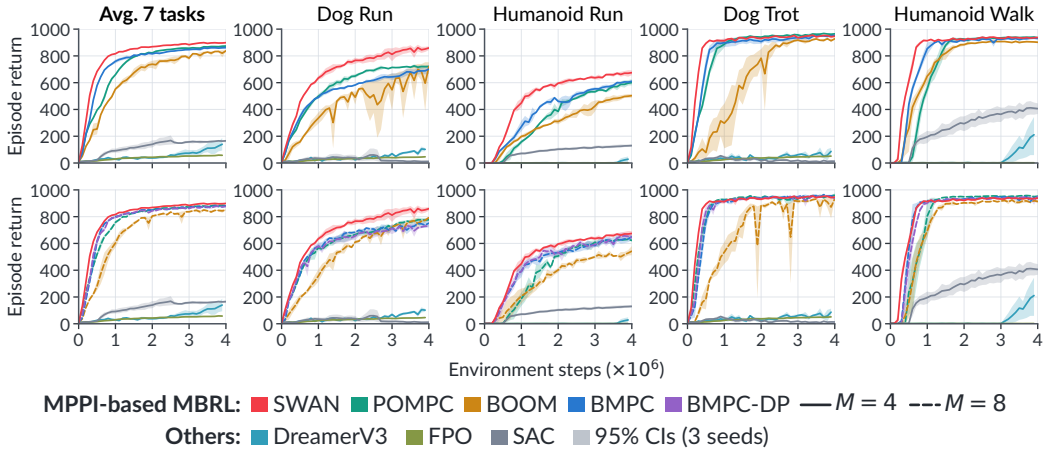


Figure 3: **DMControl locomotion suite.** SWAN uses $M = 4$ MPPI refinement iterations, while MPPI-based baselines use $M = 4$ and $M = 8$ in the top (half-budget) and bottom (full-budget) rows, respectively. Curves and shading show means and 95% CIs over 3 random seeds.

In the full-budget comparison, SWAN with $M = 4$ refinement iterations outperforms MPPI-based baselines with $M = 8$ in suite-average performance. Following threshold-based evaluations (Wang et al., 2025a; Lange et al., 2023), we measure sample efficiency by T_{80}, the environment steps until the mean evaluation curve first reaches 80% of SWAN's final performance, a common threshold across methods for each task or suite average. On DMControl, SWAN achieves 898.8 average return versus PO-MPC's 883.7, with $T_{80} = 0.60$M versus 1.00M, a 40% reduction. In particular, SWAN exceeds BOOM on *Dog-Run* and BMPC on *Humanoid-Run* by 67.2 and 25.8 in episode return, with 53.6% and 33.3% lower T_{80}, respectively. On HumanoidBench, SWAN reaches 688.0 versus PO-MPC's 604.0, while reducing T_{80} from 1.25M to 0.90M by 28%. Against the same PO-MPC baseline, SWAN improves returns on *Run* and *Sit Hard* by 112.2 and 92.0, with 37.9% and 62.5%

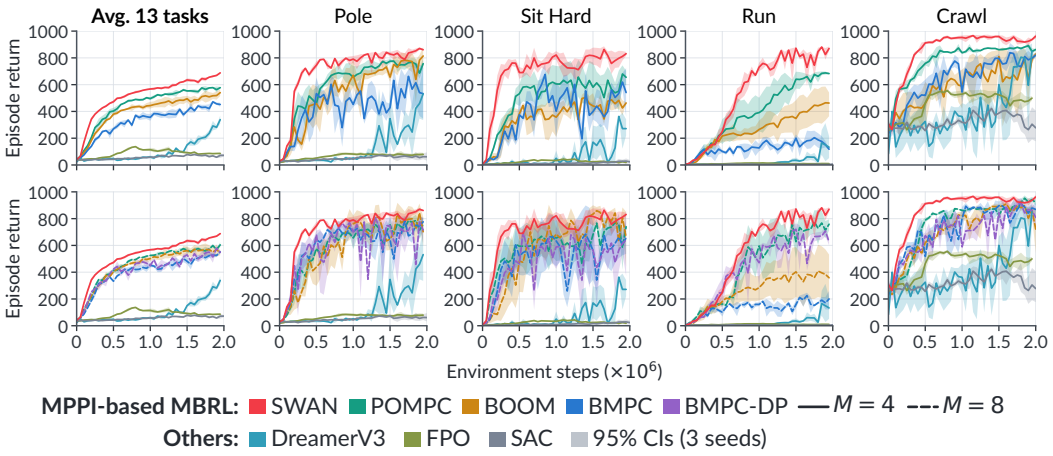


Figure 4: **HumanoidBench locomotion suite.** SWAN uses $M = 4$ MPPI refinement iterations, while MPPI-based baselines use $M = 4$ and $M = 8$ in the top (half-budget) and bottom (full-budget) rows, respectively. Curves and shading show means and 95% CIs over 3 random seeds. The suite average excludes Reach.

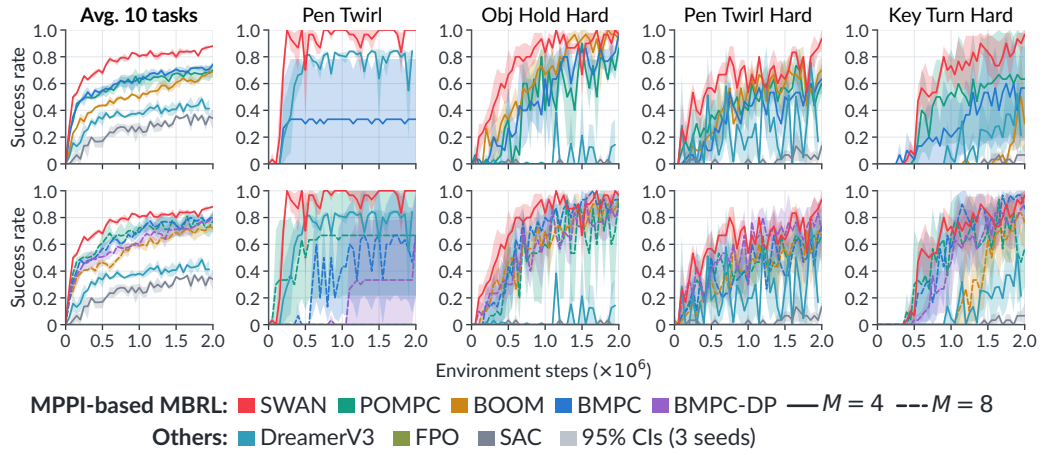


Figure 5: **MyoSuite manipulation suite.** SWAN uses $M = 4$ MPPI refinement iterations, while MPPI-based baselines use $M = 4$ and $M = 8$ in the top (half-budget) and bottom (full-budget) rows, respectively. Curves and shading show means and 95% CIs over 3 random seeds.

lower T_{80}, respectively. The gains extend to MyoSuite, where SWAN attains 0.88 average success rate compared with BMPC's 0.81, with $T_{80} = 0.55$M versus 1.05M, a 47.6% reduction. In particular, on *Pen Twirl*, SWAN reaches 1.0 success and $T_{80} = 0.25$M, whereas the MPPI-based baselines reach at most 0.667 and do not reach the threshold during evaluation.

Additionally, the half-budget setting ($M = 4$) lowers BMPC, BOOM, and PO-MPC's performance, reflecting continued dependence on MPPI refinement. SWAN's average return gains over PO-MPC widen to 25.3 and 110.5 on DMControl and HumanoidBench, with 50.0% and 37.9% lower T_{80}, respectively. On MyoSuite, SWAN's average success rate exceeds BMPC's by 0.137, with 63.3% lower T_{80}. All final performance and T_{80} results are in Appendix G.

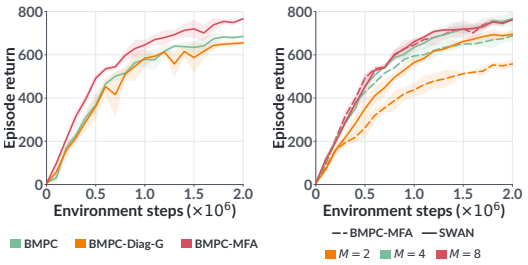
Table 1: Evaluation efficiency averaged over 7 DMControl tasks over 10 episodes with planning horizon $H = 3$. Relative time is $T_{\text{method}}/T_{\text{SWAN}}$.

Method	Trainable Params. (M) ↓	Eval. Time (s) ↓	Rel. Time ↓
BMPC	**3.29**	27.90	1.54×
BOOM	5.11	54.26	2.99×
PO-MPC	8.64	33.17	1.83×
BMPC-DP	10.56	234.38	12.93×
SWAN (ours)	3.67	**18.12**	**1.00×**

Online planning efficiency. We next assess whether the control gains are accompanied by lower wall-clock evaluation cost, averaging evaluation times over seven DMControl tasks with 10 episodes per task and planning horizon $H = 3$ on a single NVIDIA RTX A6000. As shown in Tab. 1, SWAN takes 18.12 seconds, 35.1% less than BMPC, while BOOM, PO-MPC, and BMPC-DP take 2.99×, 1.83×, and 12.93× as long, respectively. Moreover, according to Fig. 3 and Tab. 1, SWAN achieves stronger control performance than BMPC at lower evaluation cost, with 3.67M trainable parameters compared with 3.29M.

Findings from main results. We further compare SWAN, BMPC-DP and BMPC in Figs. 3, 4, 5. BMPC-DP learns complete action chunks with Diffusion Policy, yet remains close to BMPC in suite-average performance, despite much higher evaluation cost. As discussed in Sec. 2, online proposal learning requires fast sampling and adaptation to changing planner proposals. Iterative denoising adds cost, while the limited gains may reflect difficulty adapting an implicit representation online. In contrast, SWAN uses k-MPPI and MFA to explicitly model proposal modes and correlations across time and action dimensions. APO further improves the learning target using advantage estimates while preserving planner alignment in policy learning. Thus, results support combining explicit proposal representation with advantage-guided proposal optimization for efficient online planning.

Effect of action-chunk proposal representation. We compare proposal representations on *Dog-Run* under matched training and planning budgets ($M = 8$), retaining BMPC's planner imitation. BMPC uses a one-step policy, whereas *BMPC-Diag-G* models complete H-step chunks with a single diagonal Gaussian. *BMPC-MFA*, denoted BMPC-Chunk in Fig. 1, instead combines k-MPPI and MFA. As shown in Fig. 6a, BMPC-MFA's gain over BMPC supports our complete proposal representation, while the gain over BMPC-Diag-G highlights the benefits of explicitly modeling multiple modes and correlations across time and action dimensions.



(a) Proposal representation. (b) Effect of APO.

Figure 6: **Ablations on *Dog-Run*. (a)** Comparison of proposal representations in BMPC, BMPC-Diag-G, and BMPC-MFA, all using planner imitation under matched planning budgets. **(b)** Comparing SWAN (solid) with BMPC-MFA (dashed) under $M = 2, 4, 8$ MPPI refinement iterations. Curves and shading show means and 95% CIs over 3 random seeds.

Effect of APO. With the k-MPPI and MFA policy, we then isolate APO by comparing SWAN with BMPC-MFA, which retains planner imitation, under $M = 2, 4, 8$ MPPI refinement iterations. Fig. 6b shows larger gains from APO at $M = 2$ and $M = 4$, while the gap narrows at $M = 8$. Moreover, SWAN with $M = 4$ approaches its performance with $M = 8$, whereas BMPC-MFA benefits more from the additional refinement. The comparison therefore shows that policy learning with APO reduces dependence on MPPI refinement while maintaining strong control performance.

5.2 Sim-to-Real Transfer

We further train and evaluate SWAN on simulated Unitree G1 and Go2 velocity-tracking tasks (Zakka et al., 2026), where it outperforms BMPC in final return and sample efficiency with half as many MPPI refinement iterations per task. **We then demonstrate zero-shot sim-to-real transfer to a physical Go2.** The robot tracks forward, backward, and lateral velocity commands, turns in both directions, and follows figure-eight trajectories. Motion capture measurements over selected figure-eight and straight-line segments give planar velocity mean absolute errors (MAEs) of 0.126 and $0.149\,\mathrm{m/s}$, with yaw angular velocity MAEs of 0.112 and $0.068\,\mathrm{rad/s}$, respectively. We further assess locomotion stability with external pushes during motion, observing stable locomotion without falling. With only $M = 3$ MPPI refinement iterations per control step, the controller operates at average $308.16\,\mathrm{Hz}$ on NVIDIA GeForce RTX 5090. We detail simulation and real-world results in Appendix H. Overall, our SWAN achieve reliable zero-shot sim-to-real transfer ability in stable high-frequency locomotion control. Demonstrations are available at `https://swanfastworldplanner.github.io/`

6 Conclusion and Limitation

We proposed SWAN, an advantage-aware MPPI-based MBRL framework that improves online planning efficiency through better policy guidance. SWAN combines k-MPPI with an MFA action-chunk policy to capture multimodal proposal structure and introduces APO, which uses advantage estimates to construct an improved target from the planner proposal while preserving planner alignment during policy learning. Extensive experiments across multiple continuous-control benchmarks demonstrated strong control performance and sample efficiency while requiring fewer MPPI refinement iterations. We further demonstrated SWAN's zero-shot sim-to-real transfer to a physical Unitree Go2 for locomotion. One limitation is SWAN's current focus on state-based, task-specific continuous-control settings. Future work will extend SWAN to multimodal observations and multi-task learning, enabling shared proposal learning across diverse tasks and sensory modalities.

AI USE STATEMENT

In this work, we used generative AI tools to assisting with translation, assist with manuscript editing, including improving the wording and presentation of author-developed mathematical derivations and proofs. Generative AI tools were also used to assist with robot deployment integration and limited software engineering tasks, including code cleanup, refactoring, documentation, and software-level optimization. The core SWAN method and all simulation, training, and evaluation code were developed and implemented by the authors. We did not use generative AI tools for generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, or designing or providing feedback on research methodology or experiments, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for brainstorming, sourcing and searching for information, and assisting with the visualization of author-provided experimental results. Generative AI was not used to generate or modify the underlying experimental data or quantitative results. We reviewed and verified all AI-assisted text, code, and visualizations, and manually checked AI-sourced information and brainstorming suggestions against the relevant literature where applicable. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed implementation settings, hyperparameters, and benchmark configurations in Appendices E and D. Complete derivations and proofs of the theoretical results are also provided in Appendix C. The code and benchmark will be released upon acceptance.

REFERENCES

Joseph Amigo, Rooholla Khorrambakht, Elliot Chane-Sane, Nicolas Mansard, and Ludovic Righetti. First order model-based RL through decoupled backpropagation. In *9th Annual Conference on Robot Learning*, 2025.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite: A contact-rich simulation suite for musculoskeletal motor control. In *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 492–507, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=vtEn8NJWlz.

Zoubin Ghahramani, Geoffrey E Hinton, et al. The em algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

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 domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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. In *ICML*, 2022.

Kohei Honda, Naoki Akai, Kosuke Suzuki, Mizuho Aoki, Hirotaka Hosogaya, Hiroyuki Okuda, and Tatsuya Suzuki. Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7020–7026. IEEE, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=jnpILGz9gQ.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein variational model predictive control. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1278–1297. PMLR, 2021.

Moritz Lange, Noah Krystiniak, Raphael C Engelhardt, Wolfgang Konen, and Laurenz Wiskott. Improving reinforcement learning efficiency with auxiliary tasks in non-visual environments: A comparison. In *International Conference on Machine Learning, Optimization, and Data Science*, pp. 177–191. Springer, 2023.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6Anv3KB9lz.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021.

Jacob Sacks and Byron Boots. Learning sampling distributions for model predictive control. In *Conference on Robot Learning*, pp. 1733–1742. PMLR, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Alvaro Serra-Gomez, Daniel Jarne Ornia, Dhruva Tirumala, and Thomas M. Moerland. A KL-regularization framework for learning to plan with adaptive priors. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=zO8vzSGgTn.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *International conference on learning representations*, volume 2025, pp. 54241–54259, 2025a.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=E2VsqgKNlr.

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.

Kevin Zakka, Qiayuan Liao, Brent Yi, Louis Le Lay, Koushil Sreenath, and Pieter Abbeel. mjlab: A lightweight framework for gpu-accelerated robot learning. *arXiv preprint arXiv:2601.22074*, 2026.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *Advances in Neural Information Processing Systems*, 38:134093–134121, 2026.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Diffusion model predictive control. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

A NOTATION

Table 2 summarizes the symbols used in the main text.

Table 2: Symbols used in the main text.

Symbol	Meaning
Environment and value learning	
$\mathcal{M}$	Discounted Markov decision process, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.
$\mathcal{S}$	Environment state space.
$\mathcal{A}$	Continuous action space.
$\mathcal{P}$	Environment transition kernel.
$\mathcal{R}$	Environment reward function.
γ	Discount factor.
t	Environment time step.
s_t	Environment state at time t.
a_t	Action at time t.
r_t	Reward at time t.
d_s, d_a, d_z	Dimensions of the environment state, action, and latent state, respectively.
z_t	Encoded latent state, $z_t = E_\psi(s_t)$.
E_ψ	Encoder mapping an environment state to a latent state.
F_ψ	Dynamics model predicting the next latent state.
R_ψ	Reward model predicting the immediate reward.
V_ϕ	Value model approximating the state-value function and providing terminal value estimates.
$\mathcal{B}$	Replay buffer.
π_{θ_a}	Learned one-step action policy used in prior methods.
π	Policy in the RL objective or proposal distribution optimized by APO, according to context.
$J(\pi)$	Expected infinite-horizon discounted environment return.
Planning and planner proposals	
H, h	Planning horizon and latent-rollout step index, respectively.
D	Action-chunk dimension, $D = Hd_a$.
τ	Complete H-step action chunk, $\tau = a_{t:t+H-1}$.
$\widehat{z}_h$	Predicted latent state at rollout step h.
M, m	Number of MPPI refinement iterations and iteration index, respectively.
$\mathcal{T}^{(m)}$	Candidate action-chunk batch at refinement iteration m.
$\mu^{(m)}$	Mean of the Gaussian search distribution at iteration m.
$\sigma^{(m)}$	Standard deviations of the Gaussian search distribution at iteration m.
$w^{(m)}(\tau)$	Normalized weight of candidate τ at iteration m.
η_{MPPI}	Temperature for MPPI candidate weighting.
$\widehat{G}_H(z_t, \tau)$	Estimated H-step return combining predicted rewards and the terminal value estimate.
K, k	Number of proposal modes and mode index, respectively.
N	Number of elite action chunks retained per mode in k-MPPI.
$\mathcal{E}_k$	Batch of elite action chunks retained in mode k.
$\mathcal{E}$	Concatenation of all retained batches $\mathcal{E}_k$.
$w^P(\tau)$	Candidate weight normalized over all retained action chunks.
α_k^P	Planner probability of mode k, obtained by summing candidate weights within $\mathcal{E}_k$.
π_P	Empirical planner proposal over complete action chunks.
$\pi_{P,0}$	First-action marginal of the planner proposal π_P.
$\pi_{P,0}^{\mathrm{G}}$	Gaussian approximation of the first-action marginal $\pi_{P,0}$.
MFA representation	
π_θ	MFA action-chunk policy.
θ	Parameters of the MFA action-chunk policy.
e_k	Learnable embedding for mode k.

Continued on the next page

Table 2 continued

Symbol	Meaning
r	Rank parameter of the factor covariance, with $r \ll D$.
α_k^θ	Probability of component k predicted by the policy.
μ_k^θ	Mean of policy component k.
U_k^θ	Factor loading matrix of policy component k, with dimensions $D \times r$.
σ_k^θ	Residual standard deviations of policy component k.
Σ_k^θ	Covariance of policy component k.
μ_k^P	Mean of planner component k after MFA projection.
U_k^P	Factor loading matrix of planner component k after MFA projection.
σ_k^P	Residual standard deviations of planner component k after MFA projection.
Σ_k^P	Covariance of planner component k after MFA projection.
$\widetilde{\pi}_P$	Continuous MFA approximation of the planner proposal π_P.
APO and theoretical analysis	
$A(\tau)$	Advantage of chunk τ relative to the expected return under π_P.
$\bar{A}(\tau)$	Advantage normalized by the discounted horizon and the weighted standard deviation, $\bar{A}(\tau) = A(\tau)/(s_A \sum_{h=0}^{H-1} \gamma^h)$.
s_A	Weighted standard deviation of the horizon-normalized advantage under π_P and floored at 10^{-6}.
ϵ_{MD}	KL radius for the proposal improvement step.
λ	Lagrange multiplier for the proposal improvement constraint.
$\mathcal{L}(\pi, \lambda)$	Lagrangian for advantage-guided proposal improvement.
π^*	Optimal target of the KL-regularized proposal improvement objective.
$\widetilde{\pi}^*$	MFA approximation of the optimal target π^*.
ϵ	Budget for the average planner alignment KL.
β	Lagrange multiplier for the planner alignment constraint.
$\mathcal{L}_{\mathrm{APO}}(\theta, \beta)$	Lagrangian for policy learning under the planner alignment constraint.
η_β	Step size of the normalized dual update for β.
$[\cdot]_{\mathrm{EMA}}$	Exponential moving average used to set the alignment budget.
$[\cdot]_+$	Projection onto nonnegative values in the dual update.
$\widehat{J}_H(\pi)$	Expected H-step return under π with the fixed latent world model at z_t.
π_{IL}	Ideal planner imitation policy, defined by $\pi_{\mathrm{IL}} = \widetilde{\pi}_P$.
$A_{\max}$	Finite bound on the absolute estimated return centered at $\widehat{J}_H(\widetilde{\pi}_P)$.
KL	Kullback–Leibler divergence, retaining branch labels for MFA distributions.
Evaluation	
T_{80}	Environment steps until the mean evaluation curve first reaches 80% of SWAN's final performance. All methods share a threshold for each task or suite average.
$T_{\mathrm{method}}, T_{\mathrm{SWAN}}$	Wall-clock evaluation times of a compared method and SWAN, respectively.

B Details of k-MPPI and MFA

We provide the k-MPPI procedure and the MFA operations used for proposal representation and policy learning.

B.1 k-MPPI

We describe how k-MPPI constructs the empirical planner proposal. Unlike the single Gaussian update in Eq. 2, k-MPPI maintains local Gaussians initialized from the policy components and a global Gaussian for exploration. After M refinement iterations, the local and global search results form a proposal with K modes, each represented by N elite action chunks in $\mathcal{E}_k$. The branch labels retain mode assignments for MFA projection and policy learning. Algorithm 1 gives the refinement and final replacement procedure.

The sampling budget is shared across all branches, and batch concatenation preserves source labels and repeated entries. Actions obey the task bounds and masks, and search standard deviations are clipped to $[\sigma_{\min}, \sigma_{\max}]$.

In Algorithm 1, $k = 1, \ldots, K$ indexes local search branches and g denotes the global branch. For $b \in \{1, \ldots, K, g\}$, $\mu_b^{(m)}$ and $\sigma_b^{(m)}$ are the mean and standard deviations of the Gaussian search distribution at iteration m.

Algorithm 1 k-MPPI

Input z_t, π_θ, K, M, N, previous global mean, sampling allocation, and $\sigma_{\min}, \sigma_{\max}$

Initialize $\mu_k^{(1)}$ and $\sigma_k^{(1)}$ from the mean and marginal standard deviations of policy component k
Draw policy samples $\mathcal{S}_k$ from each component using Eq. 16 and reuse them across refinement iterations
Shift the previous global mean by one action and append zero to obtain $\mu_g^{(1)}$, using zero at episode start, and set $\sigma_g^{(1)} = \sigma_{\max}\mathbf{1}$
for $m = 1, \ldots, M$ **do**
    Form local batches $\mathcal{T}_k^{(m)}$ from $\mathcal{S}_k$ and fresh local Gaussian samples, and draw a global batch $\mathcal{T}_g^{(m)}$
    Concatenate the batches into $\mathcal{T}^{(m)}$ and evaluate $\widehat{G}_H$ using Eq. 1
    Retain N elites from each local batch as $\mathcal{E}_k^{(m)}$ and N elites from $\mathcal{T}^{(m)}$ as $\mathcal{E}_g^{(m)}$
    Update each Gaussian using Eq. 2 with weights normalized over the corresponding elite batch
end for
Compute final candidate weights $w^{(M)}(\tau) \leftarrow \mathrm{softmax}_{\mathcal{T}^{(M)}}(\widehat{G}_H(z_t, \tau)/\eta_{\mathrm{MPPI}})$
Sum the weights by source branch $p_b \leftarrow \sum_{\tau \in \mathcal{T}_b^{(M)}} w^{(M)}(\tau)$ for $b \in \{1, \ldots, K, g\}$
Select the nearest local branch $k^\star \leftarrow \arg\min_k \sum_{i=1}^{D} \dfrac{(\mu_{g,i}^{(M+1)} - \mu_{k,i}^{(M+1)})^2}{(\sigma_{k,i}^{(M+1)})^2 + 10^{-6}}$
if $p_g > p_{k^\star}$ **then**
    Replace $\mathcal{E}_{k^\star}^{(M)}$ and the associated scores with the global elites and scores
end if
Set $\mathcal{E}_k = \mathcal{E}_k^{(M)}$ and compute w^P, α^P using Eq. 4
Select τ with probability $w^P(\tau)$, or the highest-return chunk for greedy evaluation
Save $\mu_g^{(M+1)}$ for the next warm start
if evaluation **then**
    return the first action of τ
end if
Construct $\widetilde{\pi}_P$ using Appendix B.2 and add execution noise to τ from the selected mode
return the bounded first action, retained candidates and scores, w^P, and $\widetilde{\pi}_P$

Execution adds independent Gaussian noise using the marginal standard deviations of Σ_k^P for the selected mode, clipped to $[\sigma_{\min}, \sigma_{\max}]$. Proposal statistics are computed before execution noise.

B.2 MFA Projection

We construct the MFA approximations $\widetilde{\pi}_P$ and $\widetilde{\pi}^*$ from weighted candidates for policy learning. Since k-MPPI supplies branch assignments, each component is fitted separately using weights normalized within its branch. For nonnegative candidate weights $\omega(\tau)$ with positive total weight in each branch, the conditional weights are $\bar{\omega}_k(\tau) = \omega(\tau)/\sum_{\tau' \in \mathcal{E}_k} \omega(\tau')$. An ideal component fit minimizes the weighted negative log-likelihood

$$\min_{\mu, U, \sigma} \quad -\sum_{\tau \in \mathcal{E}_k} \bar{\omega}_k(\tau) \log \mathcal{N}\big(\tau \mid \mu, UU^\top + \mathrm{Diag}(\sigma^2)\big)\,, \tag{15}$$

where $U \in \mathbb{R}^{D \times r}$ and residual standard deviations are bounded away from zero. In implementation, we approximate the component distributions from their weighted moments. The component parameters are combined with mode probabilities obtained by summing the globally normalized input weights within each branch. Weights $\omega = w^P$ therefore give $\widetilde{\pi}_P$ with α_k^P, whereas $\omega(\tau) = \pi^*(\tau \mid z_t)$ gives $\widetilde{\pi}^*$ with α_k^*, as detailed in Appendix C.2.

B.3 Direct Sampling

For direct action-chunk generation, we select component k with probability α_k^θ. Independent Gaussian noise $\xi \sim \mathcal{N}(0, I_r)$ and $\zeta \sim \mathcal{N}(0, I_D)$ then gives

$$\tau = \mu_k^\theta + U_k^\theta \xi + \sigma_k^\theta \odot \zeta. \tag{16}$$

k-MPPI initialization applies the same expression separately to each component.

B.4 MFA KL Divergence

We compute the MFA KL used in policy learning and Theorem 2 with branch labels retained. The labels identify corresponding Gaussian components, giving the joint distributions $q(k, \tau) = \alpha_k^q q_k(\tau)$ and $p(k, \tau) = \alpha_k^p p_k(\tau)$, where q_k, p_k are Gaussian components. The joint KL therefore decomposes as

$$\mathrm{KL}(q\|p) = \sum_{k=1}^{K} \alpha_k^q \log \frac{\alpha_k^q}{\alpha_k^p} + \sum_{k=1}^{K} \alpha_k^q \mathrm{KL}(q_k\|p_k). \tag{17}$$

For each pair of Gaussian components, omitting parameter subscripts,

$$\mathrm{KL}(q_k\|p_k) = \frac{1}{2}\left[\mathrm{tr}((\Sigma^p)^{-1}\Sigma^q) + (\mu^p - \mu^q)^\top (\Sigma^p)^{-1}(\mu^p - \mu^q) - D + \log \frac{|\Sigma^p|}{|\Sigma^q|}\right]. \tag{18}$$

The inverse and determinant are evaluated using the factor covariance with the Woodbury and determinant identities. Writing $q(\tau) = \sum_k q(k, \tau)$ and $p(\tau) = \sum_k p(k, \tau)$, marginalization gives $\mathrm{KL}(q(\tau)\|p(\tau)) \le \mathrm{KL}(q\|p)$. The joint KL therefore bounds the marginal action-chunk divergence used in the policy improvement proof.

C Details of APO

C.1 Algorithm

SWAN follows the BMPC training loop and lazy reanalyze scheme (Wang et al., 2025a), retaining its model-training objectives for E_ψ, F_ψ, R_ψ while replacing the planner and policy with k-MPPI and the MFA action-chunk policy. Replay buffer stores the parameters of $\widetilde{\pi}_P$ and $\widetilde{\pi}^*$ alongside environment transitions, so policy learning uses Eq. 11 without repeating planning at every update. For value learning, a chunk is sampled from π_θ and the first action is used at each policy call in the TD rollout. Algorithm 2 summarizes the changes to the inherited training procedure.

Target construction uses the planner evaluations from the same planning call. The stored distribution parameters and current budget are fixed during policy and dual updates. The EMA coefficient and reanalysis schedule are training settings, while the corresponding update rules remain those given in the main text.

C.2 Target Derivation

We derive the optimal target π^* in Eq. 9, starting with the advantage normalization introduced in Sec. 4.2. To reduce sensitivity to reward scale and facilitate reuse of λ across tasks, we divide the horizon-normalized advantage by its weighted standard deviation under π_P. Since π_P assigns weight $w^P(\tau)$ to each retained entry in $\mathcal{E}$ and $\mathbb{E}_{\pi_P}[A] = 0$, we compute $s_A = \sqrt{\sum_{\tau\in\mathcal{E}} w^P(\tau)\left(\frac{A(\tau)}{\sum_{h=0}^{H-1}\gamma^h}\right)^2}$, We floor s_A at 10^{-6} before normalization and obtain $\bar{A}(\tau) = A(\tau)/(s_A \sum_{h=0}^{H-1}\gamma^h)$, as defined in Sec. 4.2.

Using $\bar{A}$ as the improvement signal, we obtain π^* by maximizing the Lagrangian in Eq. 8 over π, taking the current planner proposal and trajectory evaluations at z_t as given. For fixed $\lambda > 0$, the term $\lambda\epsilon_{\mathrm{MD}}$ is independent of π, so the maximization reduces to $\mathbb{E}_\pi[\bar{A}] - \lambda\,\mathrm{KL}(\pi\|\pi_P)$, which defines a mirror-descent update from π_P. Because finite KL excludes probability mass outside $\mathcal{E}$, we optimize the retained-candidate probabilities subject to $\sum_{\tau\in\mathcal{E}} \pi(\tau \mid z_t) = 1$. Introducing a multiplier ν for

Algorithm 2 SWAN training

Input Latent world model $E_\psi, F_\psi, R_\psi, V_\phi$, policy π_θ, replay buffer $\mathcal{B}$, planning and learning settings

```
Initialize the networks, target value model, and β ≥ 0
while the training budget remains do
    for each environment interaction do
        Encode the state as z_t = E_ψ(s_t)
        Run Algorithm 1 to obtain the action, retained candidates, evaluations, and π̃_P
        Construct π* using Eq. 9 and project the target using Appendix B.2
        Execute the action and observe r_t, s_{t+1}
        Store the transition and parameters of π̃_P, π̃* in B
    end for
    for each learning update do
        Sample replay sequences with their planner and target parameters
        Update E_ψ, F_ψ, R_ψ using model-training objectives in BMPC (Wang et al., 2025a)
        Update the value model V_φ with BMPC model-based TD learning, using the first action
    of a sampled MFA chunk at each policy call
        Update θ using Eq. 11 and the MFA KL in Eq. 17
        Update the current budget ε and multiplier β using Eq. 12
        Update the target value model following BMPC
        if a reanalysis update is scheduled then
            Run k-MPPI on the selected replay states and construct new targets using Eq. 9
            Project the planner and target distributions and replace their stored parameters
        end if
    end for
end while
```

this normalization constraint and differentiating with respect to each candidate probability gives

$$\bar{A}(\tau) - \lambda \left[\log \frac{\pi(\tau \mid z_t)}{\pi_P(\tau \mid z_t)} + 1 \right] - \nu = 0. \tag{19}$$

Solving Eq. 19 for $\pi(\tau \mid z_t)$ gives $\pi(\tau \mid z_t) = w^P(\tau) \exp(\bar{A}(\tau)/\lambda) \exp(-1 - \nu/\lambda)$, where $\exp(-1 - \nu/\lambda)$ is independent of τ. Normalizing over $\mathcal{E}$ therefore eliminates ν and recovers Eq. 9 as

$$\pi^*(\tau \mid z_t) = \frac{w^P(\tau) \exp(\bar{A}(\tau)/\lambda)}{\sum_{\tau' \in \mathcal{E}} w^P(\tau') \exp(\bar{A}(\tau')/\lambda)}, \qquad \tau \in \mathcal{E}. \tag{20}$$

Since the objective is strictly concave for $\lambda > 0$, the normalized stationary solution is its unique maximizer. Computing π^* only reweights the retained candidates using their existing MPPI evaluations, so no additional model rollouts are required.

Induced KL radius. To relate the target derived above to the trust region in Eq. 7, define $\epsilon_{\mathrm{MD}} = \mathrm{KL}(\pi^* \| \pi_P)$. For any π with $\mathrm{KL}(\pi \| \pi_P) \le \epsilon_{\mathrm{MD}}$, optimality of π^* in the regularized objective implies

$$\mathbb{E}_\pi[\bar{A}] - \mathbb{E}_{\pi^*}[\bar{A}] \le \lambda \left[\mathrm{KL}(\pi \| \pi_P) - \epsilon_{\mathrm{MD}} \right] \le 0. \tag{21}$$

Hence, π^* also solves Eq. 7 with the induced radius. The radius is nonnegative and becomes zero if the advantage is constant on the retained support. A fixed λ therefore specifies a regularized update rather than a common prescribed radius across states.

Projection of the target. For policy learning, we approximate the empirical target π^* with an MFA distribution $\widetilde{\pi}^*$. The mixture coefficients of $\widetilde{\pi}^*$ preserve the target probability mass in each branch

$$\alpha_k^* = \sum_{\tau \in \mathcal{E}_k} \pi^*(\tau \mid z_t). \tag{22}$$

Within branch k, the normalized target weights $\pi^*(\tau \mid z_t)/\alpha_k^*$ provide the weights for the component fitting objective in Eq. 15. We obtain the component parameters using the approximation described in Appendix B.2 and combine them with α_k^* to construct $\widetilde{\pi}^*$. The MFA approximation is used for policy learning, while Theorem 1 concerns the exact target π^*.

C.3 PROOFS

We restate and prove the two theorems in Sec. 4.3. Throughout, the latent state z_t, latent world model, and reference distributions are fixed. KL divergences between MFA distributions retain branch labels as in Appendix B.4, while expected returns and L^1 norms use the corresponding action-chunk marginals. We omit the common conditioning on z_t.

Theorem 1 (Exact target improvement)**.** *For $\lambda > 0$, the optimal target in Eq. 9 satisfies*

$$\widehat{J}_H(\pi^*) - \widehat{J}_H(\pi_P) = \mathbb{E}_{\tau\sim\pi^*}[A(\tau)] \geq 0. \tag{13}$$

Proof. By definition, $\mathbb{E}_{\pi_P}[A] = 0$. Normalization by the positive scale $s_A \sum_{h=0}^{H-1} \gamma^h$ therefore gives $\mathbb{E}_{\pi_P}[\bar{A}] = 0$. Since $\mathrm{KL}(\pi_P\|\pi_P) = 0$, comparing π^* with π_P in the KL-regularized objective yields

$$\mathbb{E}_{\pi^*}[\bar{A}] \geq \lambda\,\mathrm{KL}(\pi^*\|\pi_P) \geq 0. \tag{23}$$

Restoring the original return scale gives

$$\mathbb{E}_{\pi^*}[A] = s_A \left(\sum_{h=0}^{H-1} \gamma^h\right) \mathbb{E}_{\pi^*}[\bar{A}] \geq 0.$$

The definition of A then establishes

$$\widehat{J}_H(\pi^*) - \widehat{J}_H(\pi_P) = \mathbb{E}_{\tau\sim\pi^*}[A(\tau)] \geq 0. \qquad \square$$

Theorem 2 (Projection-aware policy improvement)**.** *Assume the expected returns and KL divergence below are finite and there exists a finite constant $A_{\max} > 0$ such that $|\widehat{G}_H(z_t,\tau) - \widehat{J}_H(\widetilde{\pi}_P)| \leq A_{\max}$ for every τ in the union of the supports of π_θ, $\widetilde{\pi}^*$, and $\widetilde{\pi}_P$. Then, for any $\beta \geq 0$,*

$$\widehat{J}_H(\pi_\theta) - \widehat{J}_H(\pi_{\mathrm{IL}}) \geq \frac{\widehat{J}_H(\widetilde{\pi}^*) - \widehat{J}_H(\widetilde{\pi}_P)}{1+\beta} - A_{\max}\sqrt{\frac{2\left[\mathrm{KL}(\widetilde{\pi}^*\|\pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|\pi_\theta)\right]}{1+\beta}}. \tag{14}$$

Proof. We bound the return improvement over $\pi_{\mathrm{IL}} = \widetilde{\pi}_P$ using the policy-fitting terms in Eq. 11. For fixed $\beta \geq 0$, both forward KL terms fit π_θ to their respective target distributions. Combining their contributions to the expected negative log-likelihood gives the normalized reference

$$q = \frac{\widetilde{\pi}^* + \beta\widetilde{\pi}_P}{1+\beta}, \tag{24}$$

where q is a joint distribution over branch labels and action chunks. Expanding the KL divergences yields

$$\mathrm{KL}(\widetilde{\pi}^*\|\pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|\pi_\theta) = (1+\beta)\,\mathrm{KL}(q\|\pi_\theta) + \mathrm{KL}(\widetilde{\pi}^*\|q) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|q). \tag{25}$$

Since the two divergences to q are nonnegative, the policy-fitting terms bound

$$\mathrm{KL}(q\|\pi_\theta) \leq \frac{\mathrm{KL}(\widetilde{\pi}^*\|\pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|\pi_\theta)}{1+\beta}. \tag{26}$$

Marginalization and Pinsker's inequality then imply

$$\|\pi_\theta - q\|_1 \leq \sqrt{2\,\mathrm{KL}(q\|\pi_\theta)} \leq \sqrt{\frac{2[\mathrm{KL}(\widetilde{\pi}^*\|\pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|\pi_\theta)]}{1+\beta}}. \tag{27}$$

To relate this distributional bound to policy improvement, we decompose the return difference as

$$\widehat{J}_H(\pi_\theta) - \widehat{J}_H(\widetilde{\pi}_P) = \widehat{J}_H(q) - \widehat{J}_H(\widetilde{\pi}_P) + \widehat{J}_H(\pi_\theta) - \widehat{J}_H(q). \tag{28}$$

The reference contribution to the centered return under q therefore vanishes, and linearity of expectation yields

$$\widehat{J}_H(q) - \widehat{J}_H(\widetilde{\pi}_P) = \frac{\widehat{J}_H(\widetilde{\pi}^*) - \widehat{J}_H(\widetilde{\pi}_P)}{1+\beta}. \tag{29}$$

It remains to bound the return difference between π_θ and q. Since their marginal densities both integrate to one, subtracting $\widehat{J}_H(\widetilde{\pi}_P)$ from the return does not change this difference. The boundedness

assumption therefore gives

$$\begin{aligned}\widehat{J}_H(\pi_\theta) - \widehat{J}_H(q) &= \int \big(\pi_\theta(\tau) - q(\tau)\big)\widehat{G}_H(z_t,\tau)\,d\tau \\ &= \int \big(\pi_\theta(\tau) - q(\tau)\big)\big(\widehat{G}_H(z_t,\tau) - \widehat{J}_H(\widetilde{\pi}_P)\big)\,d\tau \\ &\geq -A_{\max}\int \big|\pi_\theta(\tau) - q(\tau)\big|\,d\tau \\ &= -A_{\max}\|\pi_\theta - q\|_1.\end{aligned} \tag{30}$$

Adding $\widehat{J}_H(q) - \widehat{J}_H(\widetilde{\pi}_P)$ to both sides yields

$$\widehat{J}_H(\pi_\theta) - \widehat{J}_H(\widetilde{\pi}_P) \geq \widehat{J}_H(q) - \widehat{J}_H(\widetilde{\pi}_P) - A_{\max}\|\pi_\theta - q\|_1. \tag{31}$$

Substituting Eqs. 27 and 29, and using $\pi_{\mathrm{IL}} = \widetilde{\pi}_P$, gives

$$\widehat{J}_H(\pi_\theta) - \widehat{J}_H(\pi_{\mathrm{IL}}) \geq \frac{\widehat{J}_H(\widetilde{\pi}^*) - \widehat{J}_H(\widetilde{\pi}_P)}{1+\beta} - A_{\max}\sqrt{\frac{2\left[\mathrm{KL}(\widetilde{\pi}^*\|\pi_\theta) + \beta\,\mathrm{KL}(\widetilde{\pi}_P\|\pi_\theta)\right]}{1+\beta}}. \qquad \square$$

C.4 Unified View

In this section, we derive the policy learning objectives of prior methods and APO from a common distribution improvement problem. The derivation fixes the reference and evaluation signal for each update. Outer state expectations and auxiliary regularization are omitted unless needed to specify the learning distribution.

Distribution improvement. Let x denote an action or action chunk, ρ a reference distribution, and $f(x)$ a value or advantage estimate. Starting from $\max_\pi \mathbb{E}_\pi[f]$, a KL constraint limits departure from ρ. The Lagrangian in Eq. 8 gives the regularized objective and optimal distribution

$$q^* = \arg\max_\pi \left\{\mathbb{E}_\pi[f] - \lambda\,\mathrm{KL}(\pi\|\rho)\right\}, \qquad q^*(x) = \frac{\rho(x)e^{f(x)/\lambda}}{Z}, \tag{32}$$

where $\lambda > 0$ and $Z = \mathbb{E}_\rho[e^{f/\lambda}] < \infty$. The reference determines where improvement starts, and the signal changes the relative probability of the action chunks.

Direct policy optimization. Substituting Eq. 32 into KL gives

$$\mathbb{E}_\pi[f] - \lambda\,\mathrm{KL}(\pi\|\rho) = \lambda\log Z - \lambda\,\mathrm{KL}(\pi\|q^*). \tag{33}$$

Since $\log Z$ is fixed, restricting π to a parameterized policy family recovers minimization of reverse KL toward q^*. Two choices recover the critic-guided objectives below.

For TD-MPC2 (Hansen et al., 2024), let $x = a$, $f(a) = Q_\phi(z_t, a)$, and let ρ be uniform on the bounded action space $\mathcal{A}$. The identity $\mathrm{KL}(\pi\|\rho) = -\mathcal{H}(\pi) + \log|\mathcal{A}|$ then gives the maximum-entropy policy objective

$$\max_{\theta_a} \quad \mathbb{E}_{a\sim\pi_{\theta_a}}[Q_\phi(z_t,a)] + \lambda\,\mathcal{H}(\pi_{\theta_a}), \tag{34}$$

where $\mathcal{H}$ is differential entropy. Value scaling is absorbed into the relative coefficient.

For PO-MPC (Serra-Gomez et al., 2026), replace the uniform reference with the adaptive prior π_p learned from the planner's first-action distribution and set $f = Q_{\mathrm{reg}}$, the regularized action-value function. Equation 33 then recovers

$$\max_{\theta_a} \quad \mathbb{E}_{a\sim\pi_{\theta_a}}[Q_{\mathrm{reg}}(z_t,a)] - \lambda\,\mathrm{KL}(\pi_{\theta_a}\|\pi_p). \tag{35}$$

Both updates optimize an action policy, with planner information entering through the reference in Eq. 35.

Policy learning from the improved target. A second route constructs q^* first and learns a policy by minimizing $\mathrm{KL}(q^*\|\pi)$. For compatible densities with finite terms, the target entropy is constant during the policy update, giving

$$\arg\min_\pi \mathrm{KL}(q^*\|\pi) = \arg\min_\pi \left\{-\mathbb{E}_{x\sim\rho}\left[\frac{e^{f(x)/\lambda}}{Z}\log\pi(x)\right]\right\}. \tag{36}$$

The forward and reverse objectives use the same improved distribution but define different updates in a restricted policy family.

BMPC (Wang et al., 2025a) follows by setting $\rho = \pi^{\mathrm{G}}_{P,0}$ and $f = 0$. Then $q^* = \rho$, and Eq. 36 becomes

$$\min_{\theta_a} \quad \mathrm{KL}(\pi^{\mathrm{G}}_{P,0} \| \pi_{\theta_a}). \tag{37}$$

The resulting objective is the planner imitation described in Sec. 3. The zero signal denotes no additional proposal improvement.

For BOOM (Zhan et al., 2026), apply Eq. 32 to a uniform reference over a replay batch $\mathcal{C}$ of latent-state and action pairs (z, a), using $f(z, a) = Q_\phi(z, a)$. The resulting entry weights yield the alignment objective

$$\omega(z, a) = \mathrm{softmax}_{\mathcal{C}}(Q_\phi(z, a)/\lambda), \qquad \min_{\theta_a} \quad -\sum_{(z,a)\in\mathcal{C}} \omega(z, a) \log \pi_{\theta_a}(a \mid z). \tag{38}$$

BOOM combines the weighted alignment term with a separate critic-guided objective. The weights are normalized across replay entries rather than a complete proposal at one state.

APO. Setting $x = \tau$, $\rho = \pi_P$, and $f = \bar{A}$ in Eq. 32 gives the optimal target $q^* = \pi^*$ from Eq. 9. The reference is the current complete planner proposal, and the signal comes from the same MPPI trajectory evaluations. MFA projection gives the approximation $\widetilde{\pi}^*$ for policy learning, while the separate alignment constraint preserves alignment with $\widetilde{\pi}_P$. For fixed β, Eq. 11 therefore reduces to

$$\min_{\theta} \quad \mathrm{KL}(\widetilde{\pi}^* \| \pi_\theta) + \beta\, \mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta), \tag{39}$$

after omitting the constant budget term.

With no additional advantage improvement, $f = 0$ gives $\pi^* = \pi_P$ and $\widetilde{\pi}^* = \widetilde{\pi}_P$. Equation 39 becomes

$$\min_{\theta} \quad (1 + \beta)\, \mathrm{KL}(\widetilde{\pi}_P \| \pi_\theta), \tag{40}$$

which has the same minimizers as planner imitation of the complete proposal. APO thus incorporates additional proposal improvement into policy learning while preserving planner alignment. Theorems 1 and 2 establish nonnegative gain for the exact target and a sufficient condition for improvement over ideal planner imitation after projection and policy learning. The common formulation separates the choices of reference, evaluation signal, and policy update, providing a basis for further developments in learning from planning.

D BENCHMARK TASK DETAILS AND VISUALIZATIONS

We report details for the 31 benchmark tasks from DMControl, HumanoidBench, and MyoSuite, together with the two Unitree locomotion tasks used in our robot experiments. Table 3 summarizes the task dimensions and objectives, and Figs. 7–10 show representative environments.

Table 3: Task details used in this paper. The state and action dimensions and task descriptions are listed below.

Task	State	Action	Task descriptions
DMControl (Tunyasuvunakool et al., 2020)			
dog-stand	223	38	Stand stably.
dog-walk	223	38	Walk forward.
dog-trot	223	38	Trot forward.
dog-run	223	38	Run forward.
humanoid-stand	67	21	Stand upright.
humanoid-walk	67	21	Walk forward.
humanoid-run	67	21	Run forward.
HumanoidBench (Sferrazza et al., 2024)			
humanoid_h1hand-stand-v0	151	61	Stand stably.
humanoid_h1hand-walk-v0	151	61	Track walking velocity.
humanoid_h1hand-run-v0	151	61	Track running velocity.
humanoid_h1hand-crawl-v0	151	61	Crawl through a tunnel.
humanoid_h1hand-hurdle-v0	151	61	Cross hurdles.
humanoid_h1hand-stair-v0	151	61	Traverse stairs.
humanoid_h1hand-slide-v0	151	61	Traverse inclined slides.
humanoid_h1hand-pole-v0	151	61	Avoid poles while moving forward.
humanoid_h1hand-sit_simple-v0	151	61	Sit on a fixed chair.
humanoid_h1hand-sit_hard-v0	164	61	Sit under randomized configurations.
humanoid_h1hand-balance_simple-v0	164	61	Balance on a fixed-pivot board.
humanoid_h1hand-balance_hard-v0	177	61	Balance on a moving-pivot board.
humanoid_h1hand-maze-v0	151	61	Navigate through a maze.
humanoid_h1hand-reach-v0	157	61	Reach a 3-D target.
MyoSuite (Caggiano et al., 2022)			
myo-reach	115	39	Reach fixed fingertip targets.
myo-reach-hard	115	39	Reach randomized fingertip targets.
myo-pose	108	39	Match a fixed hand pose.
myo-pose-hard	108	39	Match randomized hand poses.
myo-obj-hold	91	39	Hold an object.
myo-obj-hold-hard	91	39	Hold an object from randomized states.
myo-key-turn	93	39	Turn a key to a fixed target.
myo-key-turn-hard	93	39	Turn a key to randomized targets.
myo-pen-twirl	83	39	Reorient a pen to a fixed target.
myo-pen-twirl-hard	83	39	Reorient a pen to randomized targets.
Unitree RL MjLab (Zakka et al., 2026)			
Unitree-G1-Flat	98	29	Track commanded planar velocity.
Unitree-Go2-Flat	47	12	Track commanded planar velocity.

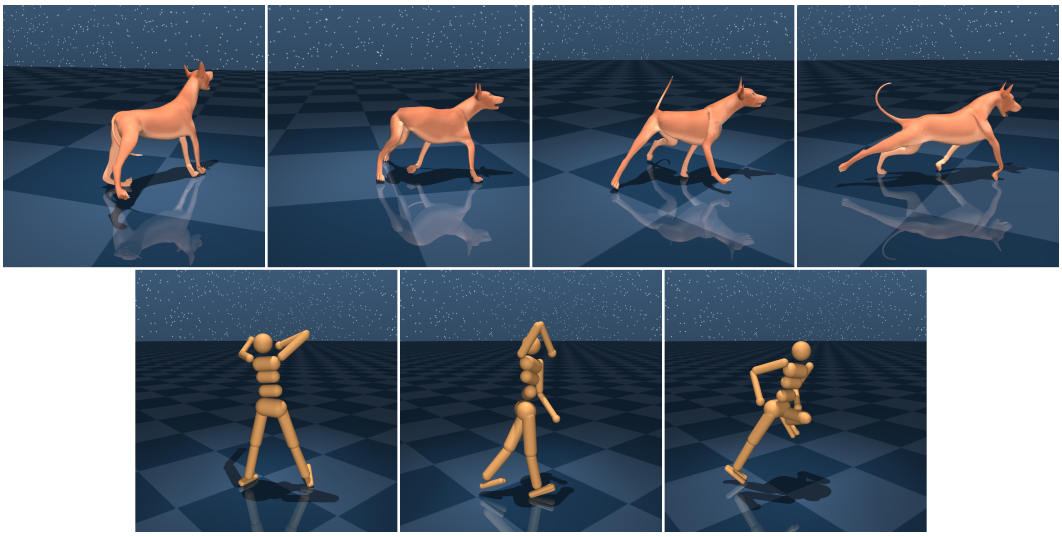


Figure 7: **DMControl task visualization.** The seven Dog and Humanoid locomotion tasks used in our experiments span standing, walking, trotting, and running.

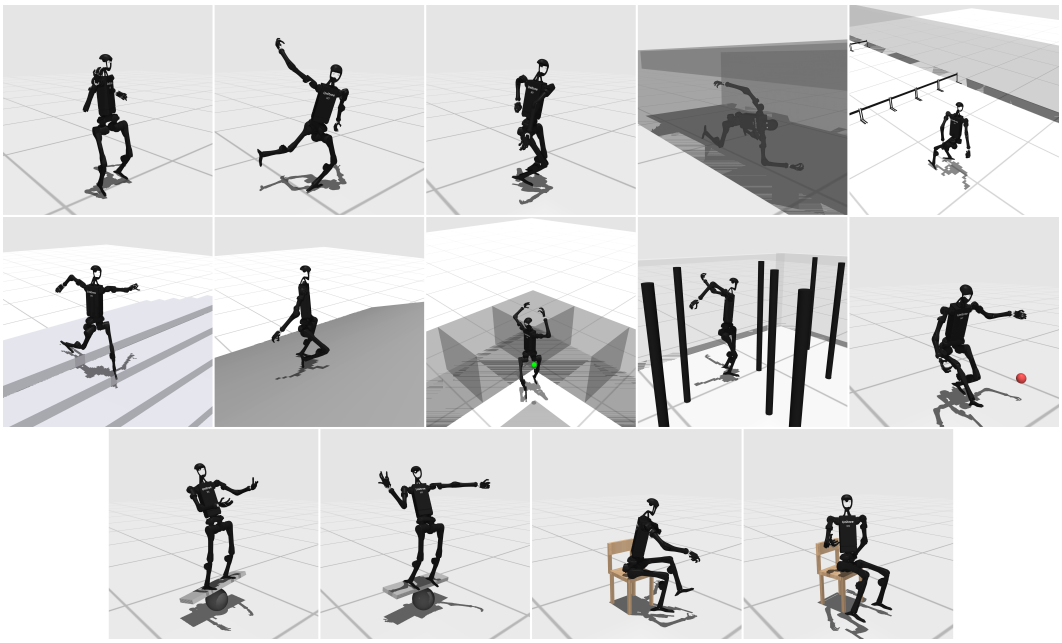


Figure 8: **HumanoidBench task visualization.** The 14 whole-body control tasks used in our experiments cover locomotion, obstacle traversal, navigation, balancing, sitting, and reaching.

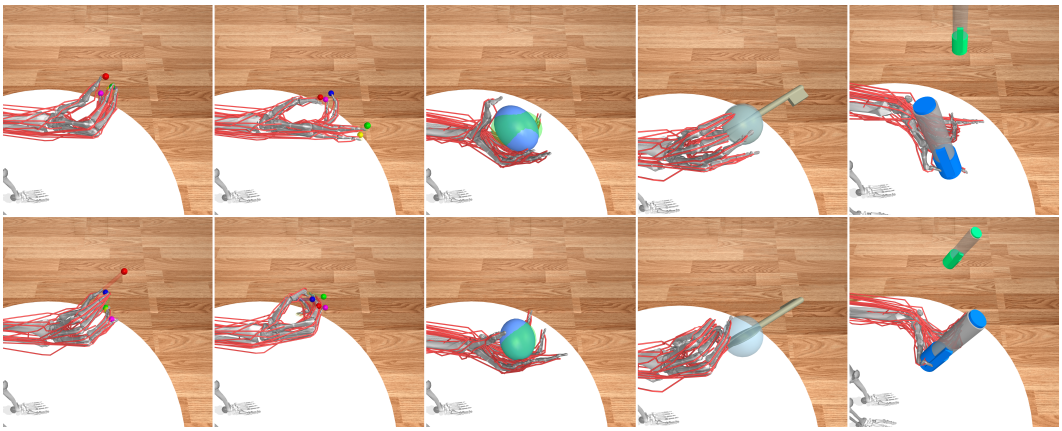


Figure 9: **MyoSuite task visualization.** Ten musculoskeletal hand-control tasks spanning reaching, posing, object holding, key turning, and pen twirling, each with standard and harder randomized variants.

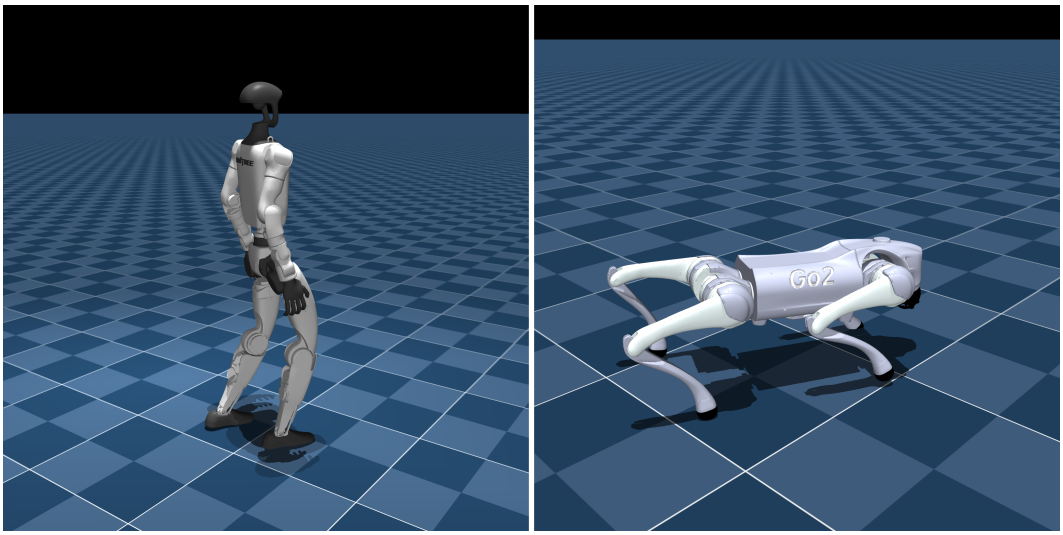


Figure 10: **Unitree locomotion visualization.** Unitree G1 and Go2 velocity-tracking tasks.

E Detailed Experimental Settings

This section provides additional experimental details for the results reported in Sec. 5. We first describe the common **training and evaluation protocol**, followed by the **implementation settings of the baselines**. We then provide the **hyperparameters** specific to SWAN and finally describe the **benchmark rewards**.

E.1 Training and Evaluation Protocol

Training details. All benchmark experiments are conducted in an online reinforcement learning setting, where agents continuously collect environment transitions during training. For fair comparison, we match the total number of environment steps across methods on each task. We define environment steps as the number of agent decision steps multiplied by the environment action repeat. The action repeat is 2 for DMControl and 1 for HumanoidBench, MyoSuite, Unitree G1, and Unitree Go2. We train agents for 4M environment steps on DMControl, 2M steps on HumanoidBench and MyoSuite, 2M steps on Unitree G1, and 1M steps on Unitree Go2. All simulation experiments are conducted with three independent random seeds on single NVIDIA RTX A6000 GPU.

For MPPI-based methods, we consider two planning-budget settings. In the **full-budget** comparison, BMPC, BOOM, PO-MPC, and BMPC-DP use $M = 8$ MPPI refinement iterations, whereas SWAN uses $M = 4$. In the **half-budget** comparison, all MPPI-based methods use $M = 4$. The number of candidate trajectories per refinement iteration is kept unchanged when varying M, such that reducing the number of refinement iterations directly reduces the number of model-rollout evaluations during planning.

Evaluation protocol. We evaluate the current controller every $50{,}000$ agent steps, corresponding to $100{,}000$ environment steps on DMControl, and using 10 evaluation episodes per task. MPPI-based methods retain online planning during evaluation using the refinement budget associated with the corresponding experimental setting. For DMControl, HumanoidBench, Unitree Go2, and Unitree G1, we report the mean episode return, whereas for MyoSuite we report the task success rate.

E.2 Baseline Implementation Details

We compare SWAN with the planning-based MBRL methods BMPC (Wang et al., 2025a), BOOM (Zhan et al., 2026), and PO-MPC (Serra-Gomez et al., 2026), the imagination-based MBRL method DreamerV3 (Hafner et al., 2023), and the model-free methods FPO (McAllister et al., 2026) and SAC (Haarnoja et al., 2018). For BMPC, BOOM, and PO-MPC, we follow algorithm-specific hyperparameters reported in their original implementations, and use the same model architectures and optimization settings across these methods for fair comparison. We detail specific and common hyperparameters of these planning-based methods in E.3. The only modification in the half-budget setting is reducing the number of MPPI refinement iterations to $M = 4$, while all other settings are kept unchanged. For DreamerV3, FPO, and SAC, we use their default configurations for DMControl tasks and match the environment-interaction budget to the other methods.

We additionally construct BMPC-DP by replacing BMPC's one-step policy with a diffusion policy that generates complete H-step action chunks, while retaining the original MPPI planner and planner-imitation objective. The diffusion policy is parameterized by a conditional 1D residual CNN operating over the action sequence, with four residual blocks, a hidden width of 512, and kernel size 3. During training, we use a `DDPMScheduler` with $1{,}000$ diffusion timesteps. During inference, action chunks are generated using a `DDIMScheduler` with 20 denoising steps.

E.3 Hyperparameter Settings

In this section, we detail specific and common hyperparameters of all planning-based methods. As shown in Tab 5, we summarize the shared architecture and optimization settings for BMPC, BOOM, PO-MPC, and SWAN. Since the methods differ in their planners, policies, and value models, we list their specific settings separately in Tab. 4.

In the full- and half-budget comparisons, the MPPI-based baselines use $M = 8$ and $M = 4$, respectively, while SWAN uses $M = 4$ in both.

E.4 Hyperparameter Selection for SWAN

Further, we show the hyperparameter selection of proposal modes K, the MFA rank parameter r, and the APO target temperature λ through comparisons on *Dog-Run*. As shown in Fig. 11 (left), increasing r from 1 to 2 provides no consistent return gain for either $K = 1$ or $K = 2$, supporting $r = 1$ with fewer covariance parameters. The middle panel further shows that $K = 2$ improves returns over $K = 1$, while $K = 4$ provides no further gain. For APO, $\lambda = 0.1$ achieves the highest final return among the five tested values in the right panel. We therefore use $K = 2$, $r = 1$, and $\lambda = 0.1$ as the default SWAN settings.

E.5 Benchmark Rewards

For each task, all methods use the same reward definition to ensure a fair comparison. Unless otherwise specified below, we use the reward functions provided by the corresponding benchmark environments.

Table 4: Planning and method-specific settings for the full- and half-budget benchmark comparisons.

Hyperparameter	Value
Shared planning settings	
Planning horizon H	3
Total candidates per refinement iteration	512
MPPI temperature η_{MPPI}	2
Search standard deviation bounds $\sigma^{(m)}$	$[0.05,\ 2]$
BMPC, BOOM, and PO-MPC	
Refinement iterations M in the full-budget comparison	8
Refinement iterations M in the half-budget comparison	4
Policy trajectories	24
Elite count	64
One-step policy π_{θ_a} hidden layers / width	2×512
Entropy coefficient	10^{-4}
BMPC	
Policy log standard deviation bounds	$[-3,\ 1]$
BOOM	
Number of Q networks	5
Policy log standard deviation bounds	$[-10,\ 2]$
Alignment coefficient / d_a for $24 \leq d_a \leq 48$	0.001
Alignment coefficient / d_a otherwise	0.02
PO-MPC	
Number of Q networks / Q^λ networks	5 / 5
Policy and prior log standard deviation bounds	$[-2,\ 2]$
Learned prior hidden layers / width	2×512
KL-regularized value loss coefficient	0.1
Policy / value KL coefficient	1
Prior learning loss	Reverse KL
Value models for BMPC and SWAN	
Number of value models V_ϕ	2
Reanalyze for BMPC, PO-MPC, and SWAN	
Interval / batch size / horizon	10 / 20 / 3
SWAN k-MPPI	
Refinement iterations M in both comparisons	4
Local modes K	2
Policy samples per mode	24
Exploration samples per mode	104
Global proposal samples	256
Elites per mode N	64
SWAN MFA action-chunk policy π_θ	
Rank parameter r	1
Mode embedding dimension $\dim(e_k)$	16
Hidden layers / width	2×512
Marginal log standard deviation bounds	$[-3,\ 1]$
SWAN APO	
Target temperature λ	0.1
Initial alignment multiplier β_0	1.0
Dual step size η_β	0.001
Alignment-budget EMA coefficient	0.001
Entropy coefficient	0

Table 5: Shared architecture and optimization settings for BMPC, BOOM, PO-MPC, and SWAN.

Hyperparameter	**Value**
General	
Training steps	2,000,000
Replay buffer capacity	1,000,000
Random seeds	1, 10, 100
Network architecture	
Latent dimension d_z	512
SimNorm group size	8
Number of bins	101
Symlog minimum / maximum	-10 / 10
Encoder E_ψ width	256
Number of encoder layers	2
Dynamics model F_ψ hidden layers / width	2×512
Reward model R_ψ hidden layers / width	2×512
Critic hidden layers / width	2×512
Optimization	
Learning rate	3×10^{-4}
Encoder learning rate	9×10^{-5}
Batch size	256
Maximum gradient norm	20
Adam betas	(0.9, 0.999)
Adam epsilon for world model / policy	10^{-8} / 10^{-5}
Target-network EMA coefficient	0.01
Model learning	
Discount γ for DMControl / HumanoidBench / MyoSuite	0.99 / 0.995 / 0.95
Model time discount ρ	0.5
Consistency coefficient	20
Reward loss coefficient	0.1
Value loss coefficient	0.1

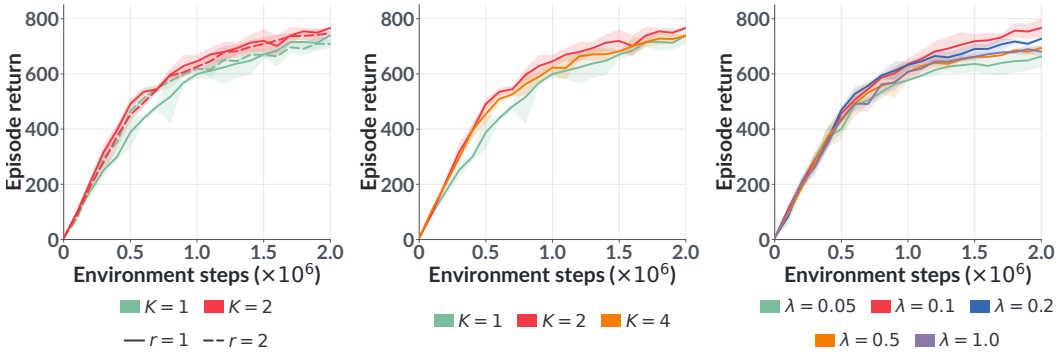


Figure 11: **Hyperparameter selection on *Dog-Run*. Left.** Comparison of $r \in \{1, 2\}$ for $K \in \{1, 2\}$. **Middle.** Comparison of $K \in \{1, 2, 4\}$ with $r = 1$. **Right.** Comparison of APO target temperatures $\lambda \in \{0.05, 0.1, 0.2, 0.5, 1.0\}$. All panels show episode return over 2M environment steps.

DMControl, HumanoidBench, and MyoSuite. For DMControl (Tunyasuvunakool et al., 2020), HumanoidBench (Sferrazza et al., 2024), and MyoSuite (Caggiano et al., 2022), we use the reward functions provided by the original benchmark environments.

Unitree Go2. The Go2 velocity-tracking task is based on the locomotion reward of DMO (Amigo et al., 2025), reimplemented in MJLab. For the simulation experiments, we retain the original DMO reward terms and coefficients without modification. For the refinement stage used prior to real-world deployment, we modify only two reward components to improve tracking precision and standing behavior.

First, we refine the linear- and yaw-velocity tracking rewards using mixtures of wide and narrow exponential kernels:

$$r_t^{\text{lin}} = 0.5 \exp\left(-\frac{\|\mathbf{c}_t^{xy} - \mathbf{v}_t^{xy}\|_2^2}{0.5^2}\right) + 0.5 \exp\left(-\frac{\|\mathbf{c}_t^{xy} - \mathbf{v}_t^{xy}\|_2^2}{0.15^2}\right), \tag{41}$$

$$r_t^{\text{yaw}} = 0.75 \exp\left(-\frac{(c_t^{\omega} - \omega_t^z)^2}{0.5^2}\right) + 0.25 \exp\left(-\frac{(c_t^{\omega} - \omega_t^z)^2}{0.10^2}\right), \tag{42}$$

where $\mathbf{c}_t^{xy}$ and c_t^{ω} denote the commanded planar and yaw velocities, and $\mathbf{v}_t^{xy}$ and ω_t^z are the corresponding base velocities. The narrower kernels provide stronger incentives for precise velocity tracking.

Second, for near-zero commands, $\|\mathbf{c}_t\|_2 < 0.1$, we disable the swing-contact and swing-clearance penalties and instead encourage the nominal standing pose using

$$\ell_t^{\text{stand}} = \|\mathbf{q}_t - \mathbf{q}_0\|_2^2, \tag{43}$$

where $\mathbf{q}_t$ is the current joint configuration and $\mathbf{q}_0$ is the nominal standing pose. All other DMO reward terms and coefficients remain unchanged during this refinement stage.

Unitree G1. The G1 task is based on the official `Unitree-G1-Flat` environment in MJLab (Zakka et al., 2026). Unless specified below, we retain its original reward terms, coefficients, and control-timestep scaling.

We replace the native foot-gait reward with our defined gait reward,

$$r_t^{\text{gait}} = -0.8H_t - 0.8H_t^{\text{sym}} + 0.2C_t + 0.5P_t + 0.5G_t, \tag{44}$$

where H_t penalizes insufficient swing-foot clearance, H_t^{sym} penalizes left–right asymmetry in swing clearance, C_t rewards transferring contact load from the swing foot to the stance foot, P_t rewards airborne foot progress along the commanded motion, and G_t rewards agreement between the planned gait phase and the observed contact pattern. The clearance target increases from $0.05\,\text{m}$ to $0.10\,\text{m}$ with commanded foot speed, and the symmetry coefficient is 0.25. The gait reward and phase observation are disabled when $\|\mathbf{c}_t\|_2 < 0.1$, consistent with the standing branch of the nominal-pose reward.

We additionally replace the native torso-orientation penalty with

$$-3 \max\left(o_t^{\text{torso}}, o_t^{\text{pelvis}}\right), \tag{45}$$

where each term measures the squared horizontal component of gravity projected into the corresponding body frame. We also terminate an episode when the root height falls below $0.2\,\text{m}$ or when a non-foot ground collision exceeds $10\,\text{N}$; both events receive the same -200 non-time-limit termination penalty. As in the original MJLab implementation, the complete weighted reward is multiplied by the $0.02\,\text{s}$ control timestep.

F Full Benchmark Learning Curves

We provide the complete learning curves for all 31 benchmark tasks evaluated in Sec. 5.1. Figs. 12–17 report all 7 DMControl (Tunyasuvunakool et al., 2020), 14 HumanoidBench (Sferrazza et al., 2024), and 10 MyoSuite (Caggiano et al., 2022) tasks under the half- and full-budget comparisons. SWAN uses $M = 4$ MPPI refinement iterations throughout. The MPPI-based baselines use $M = 4$ in the half-budget setting and $M = 8$ in the full-budget comparison. Curves and shaded regions show means and 95% CIs over three seeds.

Across the complete task set, SWAN consistently achieves strong control performance relative to the competing baselines. Despite using only $M = 4$ refinement iterations, SWAN remains competitive with or outperforms the baselines on many tasks across all three benchmark suites, with larger gains under the matched refinement budget. These results further demonstrate the effectiveness of SWAN across diverse locomotion and manipulation tasks.

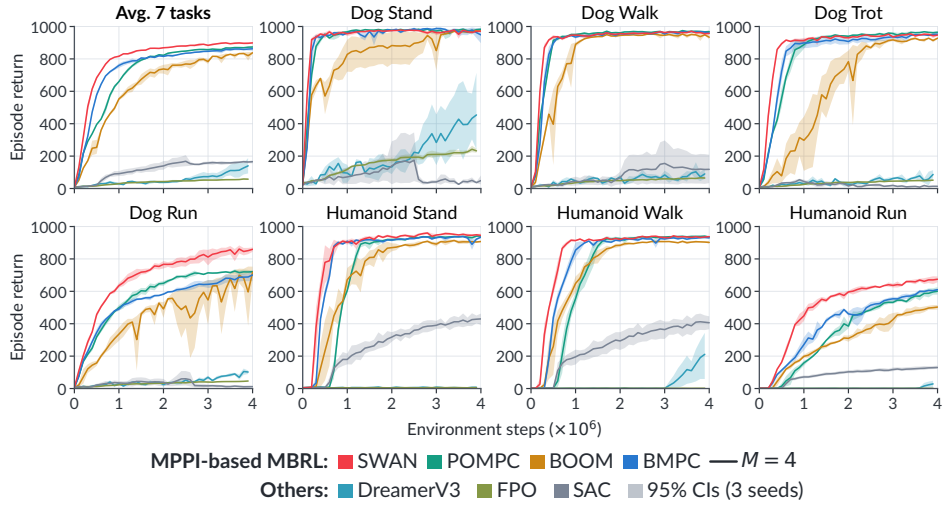


Figure 12: **Full learning curves on all 7 DMControl tasks under the half-budget setting.** All MPPI-based methods use $M = 4$ refinement iterations.

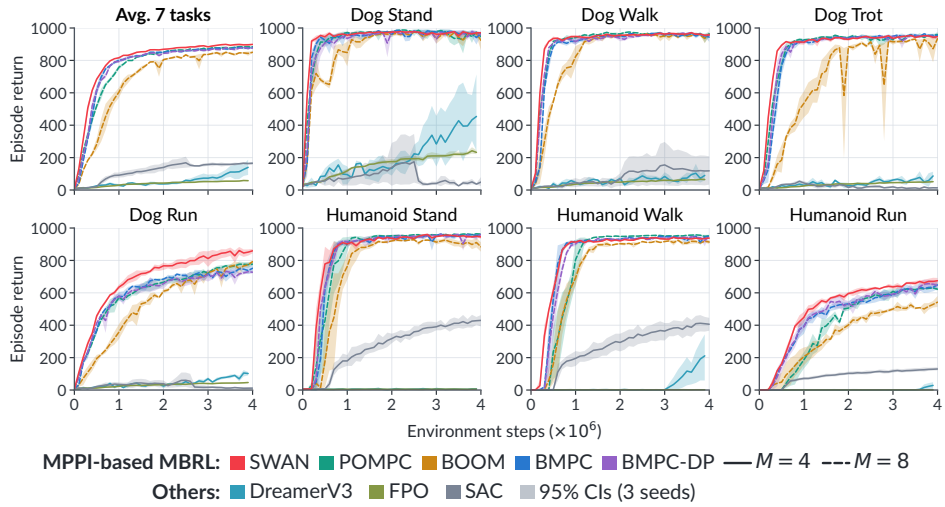


Figure 13: **Full learning curves on all 7 DMControl tasks under the full-budget comparison.** SWAN uses $M = 4$, while the MPPI-based baselines use $M = 8$.

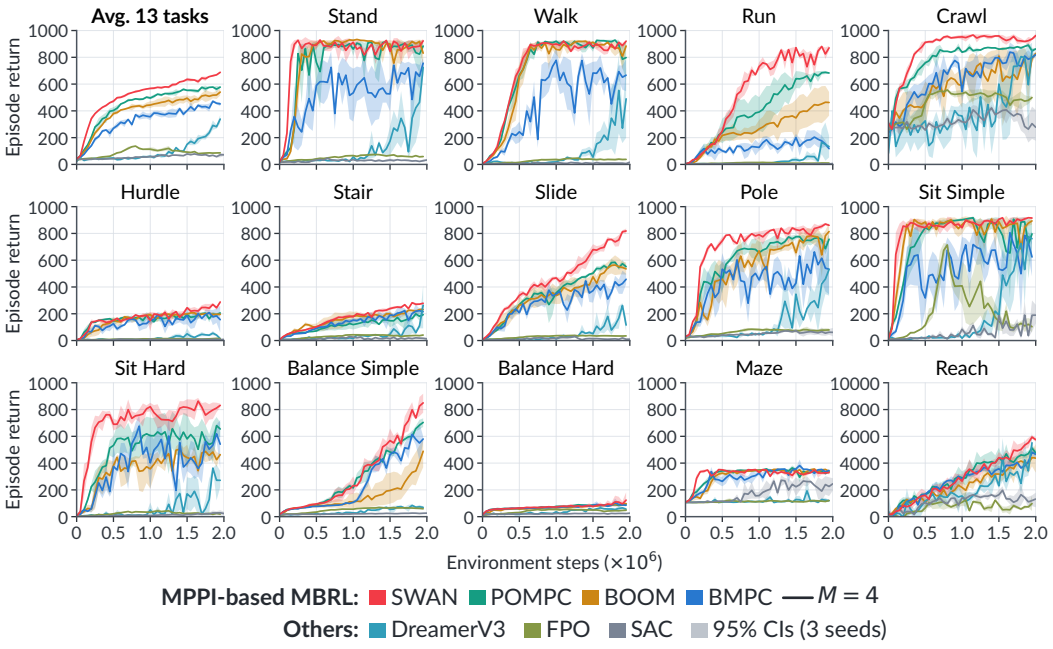


Figure 14: **Full learning curves on all** 14 **HumanoidBench tasks under the half-budget setting.** All MPPI-based methods use $M = 4$ refinement iterations.

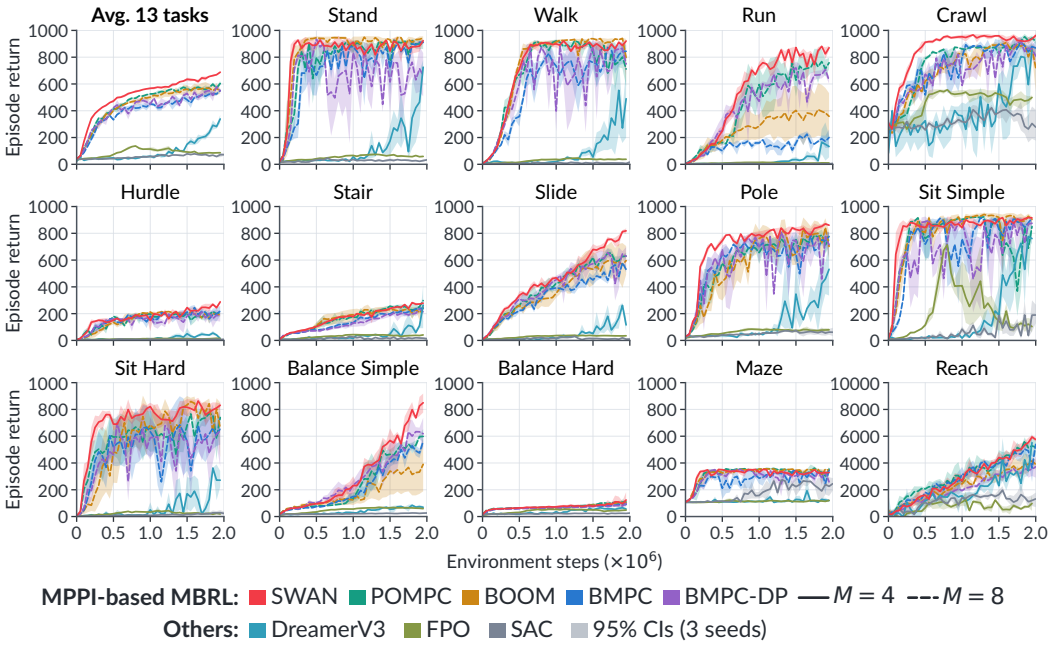


Figure 15: **Full learning curves on all** 14 **HumanoidBench tasks under the full-budget comparison.** SWAN uses $M = 4$, while the MPPI-based baselines use $M = 8$.

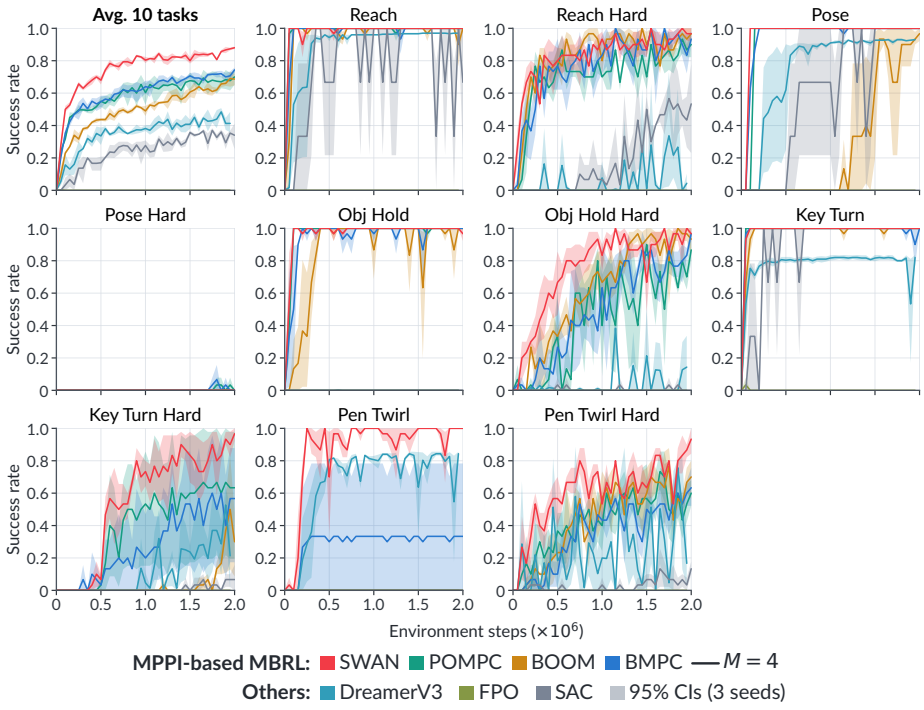


Figure 16: **Full learning curves on all** 10 **MyoSuite tasks under the half-budget setting.** All MPPI-based methods use $M = 4$ refinement iterations.

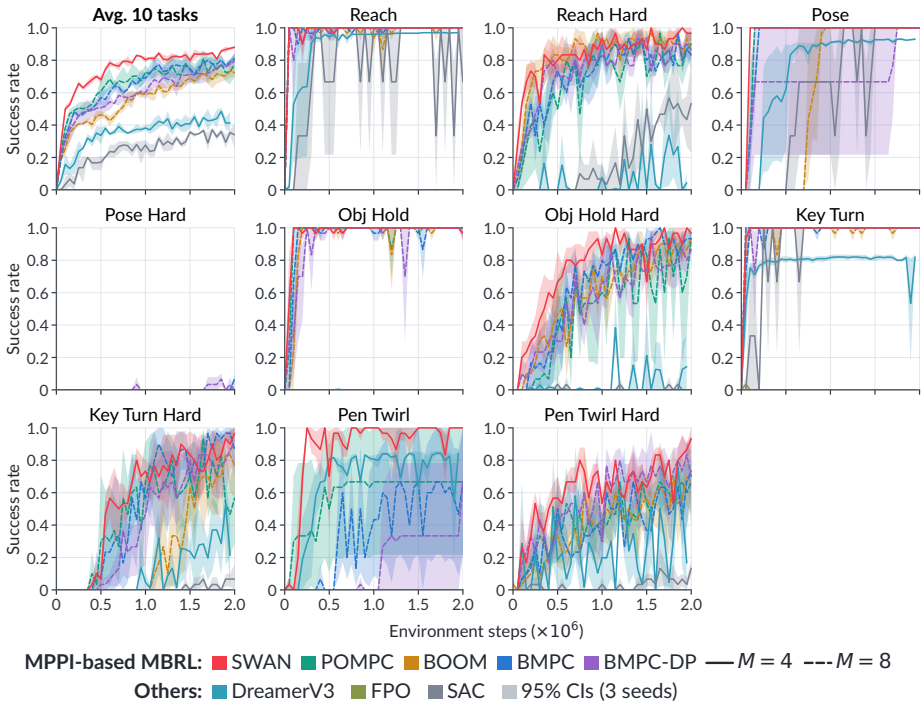


Figure 17: **Full learning curves on all** 10 **MyoSuite tasks under the full-budget comparison.** SWAN uses $M = 4$, while the MPPI-based baselines use $M = 8$.

G Final Performance and Sample Efficiency

We provide complete final-performance and sample-efficiency statistics for Sec. 5.1, reporting full- and half-budget comparisons separately. SWAN uses $M = 4$ MPPI refinement iterations in both settings, while the MPPI-based baselines use $M = 8$ and $M = 4$, respectively.

For each method, task-level evaluation curves are averaged over three random seeds. Suite-average curves use evaluation steps shared by all included tasks and seeds, averaging first across tasks within each seed and then across seeds. The HumanoidBench suite average includes 13 tasks, excluding Reach. For each task or suite average, *Final* is the last observed value of the corresponding mean evaluation curve, without smoothing or averaging over a terminal window. Because individual task curves may extend beyond the last shared evaluation step, suite-level *Final* need not equal the average of task-level *Final* values.

We measure sample efficiency by T_{80}, the first observed environment step at which a method's mean evaluation curve reaches 80% of SWAN's *Final* for the corresponding task or suite average. All methods therefore share the same reference threshold. We report T_{80} in millions of environment steps, with lower values indicating better sample efficiency. The symbol "–" indicates that the threshold is not reached within the recorded evaluations.

Analysis. Under the full-budget comparison in Tables 6, 8, and 10, SWAN achieves the highest suite-average performance across all three benchmarks despite using only $M = 4$ refinement iterations. On DMControl, SWAN obtains an average return of 898.8 with $T_{80} = 0.60$M, compared with 883.7 and 1.00M for PO-MPC with $M = 8$. On *Dog-Run*, SWAN improves the final return over BOOM by 67.2, while reducing T_{80} from 2.80M to 1.30M; on *Humanoid-Run*, it improves over BMPC by 25.8 with T_{80} reduced from 2.10M to 1.40M. On HumanoidBench, SWAN reaches an average return of 688.0 with $T_{80} = 0.90$M, compared with 604.0 and 1.25M for PO-MPC. The gains are particularly pronounced on *Run* and *Sit Hard*, where SWAN improves the final return over PO-MPC by 112.2 and 92.0, while reducing T_{80} by 37.9% and 62.5%, respectively. On MyoSuite, SWAN achieves 88.0% average success with $T_{80} = 0.55$M, compared with 81.0% and 1.05M for BMPC. On *Pen Twirl*, SWAN reaches 100% final success with $T_{80} = 0.25$M, whereas the MPPI-based baselines reach at most 66.7% and do not reach the shared threshold.

Under the matched $M = 4$ half-budget setting in Tables 7, 9, and 11, the gaps generally increase. SWAN improves the suite-average return over PO-MPC by 25.3 on DMControl and 110.5 on HumanoidBench, while reducing T_{80} by 50.0% and 37.9%, respectively. On MyoSuite, SWAN improves the average success rate over BMPC by 13.7 percentage points and reduces T_{80} by 63.3%. The wider gaps therefore indicate that the competing MPPI-based methods benefit more from additional refinement, whereas SWAN maintains strong control performance and sample efficiency under the reduced planning budget. Moreover, BMPC and BOOM no longer reach the HumanoidBench suite-average threshold with four iterations, while PO-MPC and BOOM no longer reach the MyoSuite threshold.

Table 6: **DMControl full-budget comparison.** SWAN uses $M = 4$ and MPPI-based baselines use $M = 8$. Paired rows report final episode return ↑ and T_{80} ↓.

Task	Metric	MPPI-based methods					Other baselines		
		SWAN $M = 4$	PO-MPC $M = 8$	BMPC $M = 8$	BOOM $M = 8$	BMPC-DP $M = 8$	DreamerV3	FPO	SAC
Avg. 7 tasks	Final	**898.8**	883.7	881.5	850.1	876.0	140.0	57.6	165.4
	T_{80}	**0.60**	1.00	0.70	1.40	0.80	–	–	–
Dog Stand	Final	**970.5**	947.5	965.7	924.3	958.1	454.0	232.5	48.9
	T_{80}	**0.20**	0.30	**0.20**	0.70	0.30	–	–	–
Dog Walk	Final	957.5	**964.1**	943.8	960.5	957.0	90.1	64.6	118.3
	T_{80}	**0.30**	0.40	0.40	1.00	0.40	–	–	–
Dog Trot	Final	946.6	**960.4**	954.5	934.2	936.6	86.7	51.6	13.3
	T_{80}	**0.40**	0.50	0.50	1.60	0.50	–	–	–
Dog Run	Final	**859.4**	777.0	751.5	792.2	727.2	101.8	46.4	11.8
	T_{80}	**1.30**	2.50	2.10	2.80	2.30	–	–	–
Humanoid Stand	Final	945.1	**962.5**	957.0	883.9	960.0	7.5	5.5	429.8
	T_{80}	**0.50**	0.70	0.60	1.00	0.70	–	–	–
Humanoid Walk	Final	937.8	**952.2**	949.3	913.6	948.9	211.1	1.6	406.2
	T_{80}	0.70	1.00	**0.60**	1.10	0.80	–	–	–
Humanoid Run	Final	**674.7**	622.5	648.9	542.2	644.2	28.5	1.1	129.8
	T_{80}	**1.40**	2.30	2.10	4.00	1.60	–	–	–

Table 7: **DMControl half-budget comparison.** All methods use $M = 4$. Each method reports final episode return ↑ and T_{80} ↓.

Task	**SWAN**		PO-MPC		BMPC		BOOM	
	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓
Avg. 7 tasks	**898.8**	**0.60**	873.5	1.20	862.4	0.80	837.7	1.90
Dog Stand	970.5	**0.20**	**983.0**	0.30	948.4	**0.20**	972.0	1.00
Dog Walk	957.5	**0.30**	**968.5**	0.50	960.3	0.40	933.9	0.80
Dog Trot	946.6	**0.40**	**964.6**	0.70	952.8	0.60	927.7	2.00
Dog Run	**859.4**	**1.30**	721.2	2.30	701.2	3.80	718.1	3.70
Humanoid Stand	**945.1**	**0.50**	941.3	1.20	934.4	0.70	907.4	1.40
Humanoid Walk	**937.8**	**0.70**	934.9	1.30	929.6	0.90	901.8	1.20
Humanoid Run	**674.7**	**1.40**	600.7	3.10	610.4	2.80	502.9	–

Note. T_{80} is in millions of environment steps.

Table 8: **HumanoidBench full-budget comparison.** SWAN uses $M = 4$ and MPPI-based baselines use $M = 8$. Paired rows report final episode return ↑ and T_{80} ↓.

Task	Metric	MPPI-based methods					Other baselines		
		SWAN $M = 4$	PO-MPC $M = 8$	BMPC $M = 8$	BOOM $M = 8$	BMPC-DP $M = 8$	DreamerV3	FPO	SAC
Avg. 13 tasks	Final	**688.0**	604.0	551.1	545.9	548.3	338.7	86.5	72.0
	T_{80}	**0.90**	1.25	1.95	1.45	1.80	–	–	–
Stand	Final	922.6	896.3	910.3	**943.4**	663.3	725.5	59.0	33.1
	T_{80}	**0.15**	0.30	0.40	0.20	0.25	–	–	–
Walk	Final	**920.8**	711.6	844.0	796.1	749.3	489.8	37.6	15.6
	T_{80}	0.55	0.60	0.80	**0.50**	0.55	–	–	–
Run	Final	**870.4**	758.2	200.1	358.3	625.9	133.1	10.1	6.8
	T_{80}	**0.90**	1.45	–	–	1.45	–	–	–
Crawl	Final	**961.9**	919.6	868.9	721.6	809.4	852.3	499.3	277.7
	T_{80}	**0.35**	0.50	0.75	0.95	0.65	1.80	–	–
Hurdle	Final	**287.8**	208.7	219.7	193.0	180.1	13.7	14.3	8.1
	T_{80}	**1.50**	–	1.60	–	–	–	–	–
Stair	Final	278.4	**296.9**	238.1	245.2	213.1	264.0	41.0	12.7
	T_{80}	1.35	1.25	1.85	**1.15**	1.90	1.95	–	–
Slide	Final	**818.5**	630.1	529.7	596.2	628.6	115.5	34.7	10.0
	T_{80}	**1.50**	–	–	–	1.70	–	–	–
Pole	Final	**860.8**	755.6	776.3	702.3	715.9	531.4	79.4	61.2
	T_{80}	**0.35**	0.90	0.65	0.90	0.70	–	–	–
Sit Simple	Final	**915.1**	850.8	913.4	903.6	875.3	760.7	103.1	189.8
	T_{80}	**0.15**	0.30	0.30	0.30	0.45	1.90	–	–
Sit Hard	Final	**831.3**	739.3	657.4	659.8	642.2	271.6	18.9	23.6
	T_{80}	**0.30**	0.80	0.95	0.65	0.55	–	–	–
Balance Simple	Final	**849.8**	598.5	561.2	389.9	618.4	69.5	59.1	26.5
	T_{80}	**1.75**	–	–	–	–	–	–	–
Balance Hard	Final	**123.6**	108.8	89.6	101.0	97.4	55.9	47.9	26.1
	T_{80}	1.80	**1.60**	–	1.80	–	–	–	–
Maze	Final	328.4	**354.4**	345.6	343.7	259.3	119.6	120.2	245.4
	T_{80}	**0.15**	0.25	0.30	0.20	0.25	–	–	1.20
Reach	Final	**5750.5**	5173.3	5387.1	3893.4	3712.8	5517.0	977.6	1317.2
	T_{80}	**1.70**	**1.70**	**1.70**	–	–	1.95	–	–

Table 9: **HumanoidBench half-budget comparison.** All methods use $M = 4$. Each method reports final episode return ↑ and T_{80} ↓.

Task	**SWAN**		PO-MPC		BMPC		BOOM	
	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓
Avg. 13 tasks	**688.0**	**0.90**	577.5	1.45	452.0	–	540.3	–
Stand	**922.6**	**0.15**	884.0	0.25	756.2	1.55	830.8	0.30
Walk	**920.8**	**0.55**	798.0	0.60	666.6	1.00	884.2	0.60
Run	**870.4**	**0.90**	682.9	–	117.7	–	462.7	–
Crawl	**961.9**	**0.35**	862.5	0.55	819.9	0.90	828.3	1.50
Hurdle	**287.8**	**1.50**	205.9	–	155.9	–	195.6	–
Stair	**278.4**	**1.35**	193.1	–	215.5	1.70	231.4	1.75
Slide	**818.5**	**1.50**	551.7	–	457.8	–	535.2	–
Pole	**860.8**	**0.35**	756.9	1.15	532.6	–	814.2	1.35
Sit Simple	**915.1**	**0.15**	796.6	0.40	624.0	1.65	893.5	0.25
Sit Hard	**831.3**	**0.30**	654.9	1.90	541.1	0.85	463.0	–
Balance Simple	**849.8**	**1.75**	703.8	1.90	580.1	–	488.5	–
Balance Hard	**123.6**	**1.80**	94.0	–	85.4	**1.80**	97.3	–
Maze	328.4	**0.15**	337.4	0.30	**344.4**	0.35	335.7	0.40
Reach	**5750.5**	1.70	4755.5	**1.60**	4658.0	1.95	4370.1	–

Note. T_{80} is in millions of environment steps. The suite average excludes Reach, which is reported separately.

Table 10: **MyoSuite full-budget comparison.** SWAN uses $M = 4$ and MPPI-based baselines use $M = 8$. Paired rows report final success rate (%) ↑ and T_{80} ↓.

Task	Metric	MPPI-based methods					Other baselines		
		SWAN $M = 4$	PO-MPC $M = 8$	BMPC $M = 8$	BOOM $M = 8$	BMPC-DP $M = 8$	DreamerV3	FPO	SAC
Avg. 10 tasks	Final	**88.0**	77.3	81.0	72.3	79.7	41.3	0.0	34.0
	T_{80}	**0.55**	1.00	1.05	1.65	1.40	–	–	–
Reach	Final	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	97.0	0.0	66.7
	T_{80}	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	0.30	–	0.35
Reach Hard	Final	**96.7**	90.0	83.3	86.7	86.7	4.3	0.0	53.3
	T_{80}	**0.35**	0.60	0.70	0.45	0.45	–	–	–
Pose	Final	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	93.0	0.0	**100.0**
	T_{80}	**0.10**	0.15	0.20	0.90	1.75	0.55	–	1.15
Pose Hard	Final	0.0	0.0	**6.7**	0.0	0.0	0.0	0.0	0.0
	T_{80}	–	–	–	–	–	–	–	–
Obj Hold	Final	96.7	**100.0**	**100.0**	**100.0**	96.7	0.0	0.0	0.0
	T_{80}	**0.10**	**0.10**	0.15	0.15	0.25	–	–	–
Obj Hold Hard	Final	**96.7**	93.3	93.3	93.3	86.7	14.3	0.0	0.0
	T_{80}	**0.60**	1.10	1.00	1.40	1.10	–	–	–
Key Turn	Final	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	82.0	0.0	**100.0**
	T_{80}	**0.05**	**0.05**	0.10	**0.05**	**0.05**	0.45	–	0.25
Key Turn Hard	Final	**96.7**	56.7	86.7	76.7	90.0	21.3	0.0	6.7
	T_{80}	**0.90**	1.30	1.10	1.90	1.50	–	–	–
Pen Twirl	Final	**100.0**	66.7	66.7	0.0	66.7	84.3	0.0	0.0
	T_{80}	**0.25**	–	–	–	–	0.50	–	–
Pen Twirl Hard	Final	**93.3**	66.7	73.3	66.7	70.0	16.7	0.0	13.3
	T_{80}	**0.75**	–	–	–	1.05	1.85	–	–

Table 11: **MyoSuite half-budget comparison.** All methods use $M = 4$. Each method reports final success rate (%) ↑ and T_{80} ↓.

Task	**SWAN**		PO-MPC		BMPC		BOOM	
	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓	Final ↑	T_{80} ↓
Avg. 10 tasks	**88.0**	**0.55**	70.0	–	74.3	1.50	68.7	–
Reach	**100.0**	**0.05**	**100.0**	**0.05**	**100.0**	0.10	**100.0**	0.10
Reach Hard	**96.7**	0.35	90.0	0.55	93.3	0.70	**96.7**	**0.30**
Pose	**100.0**	**0.10**	**100.0**	0.15	**100.0**	0.15	96.7	1.55
Pose Hard	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**	–
Obj Hold	96.7	**0.10**	**100.0**	**0.10**	**100.0**	0.15	**100.0**	0.35
Obj Hold Hard	**96.7**	**0.60**	86.7	0.95	**96.7**	1.25	93.3	1.20
Key Turn	**100.0**	**0.05**	**100.0**	**0.05**	**100.0**	**0.05**	**100.0**	0.10
Key Turn Hard	**96.7**	**0.90**	63.3	–	56.7	–	30.0	–
Pen Twirl	**100.0**	**0.25**	0.0	–	33.3	–	0.0	–
Pen Twirl Hard	**93.3**	**0.75**	60.0	–	63.3	–	70.0	1.75

Note. T_{80} is in millions of environment steps. For Pose Hard, SWAN's final success rate and the reference threshold are zero. Its unreported T_{80} entries should not be interpreted as failure to reach a positive threshold.

H UNITREE SIM-TO-REAL TRANSFER

Unitree G1 and Go2 Locomotion. In this section, we evaluate on velocity tracking tasks for Unitree G1 and Go2 in MJLab (Zakka et al., 2026) to assess whether SWAN's performance and sample-efficiency gains extend to robot locomotion. As shown in Fig. 18, SWAN outperforms BMPC in final return and sample efficiency while using half as many MPPI refinement iterations on both tasks. Specifically, BMPC uses $M = 8$ and $M = 6$ on G1 and Go2, respectively, whereas SWAN uses $M = 4$ and $M = 3$. The sample-efficiency advantage is particularly pronounced on G1, demonstrating effective learning on the higher-dimensional task despite the smaller refinement budget.

Unitree Go2 Sim-to-Real Transfer. Beyond simulation, we evaluate SWAN's ability to zero-shot sim-to-real transfer by deploying the trained SWAN controller on a physical Unitree Go2. To facilitate sim-to-real transfer, we apply domain randomization alongside the terrain curriculum in policy learning. After training on the simulation, SWAN demonstrates zero-shot sim-to-real transfer on a physical Unitree Go2 with single NVIDIA GeForce RTX 5090. Under a limited online planning budget, the robot tracks commanded forward, backward, and lateral velocities, turns in both directions, and follows figure-eight trajectories that combine translation with changes in turning direction.

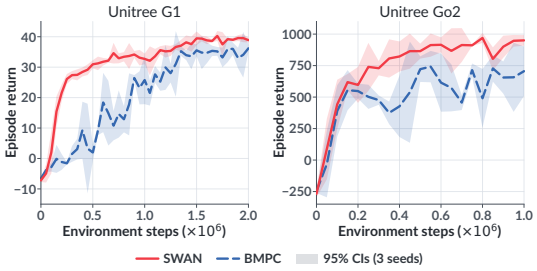
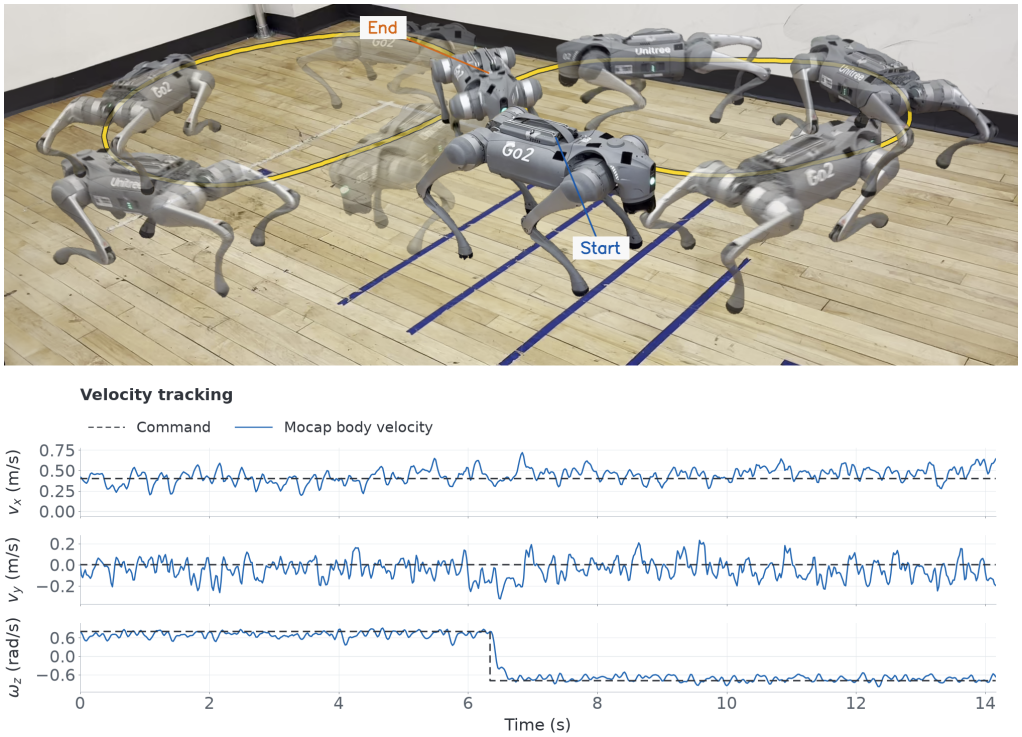


Figure 18: **Unitree G1 and Go2 locomotion in simulation.** Comparison between SWAN (solid) and BMPC (dashed), where SWAN uses half as many MPPI refinement iterations. Curves and shading show means and 95% CIs over 3 seeds.

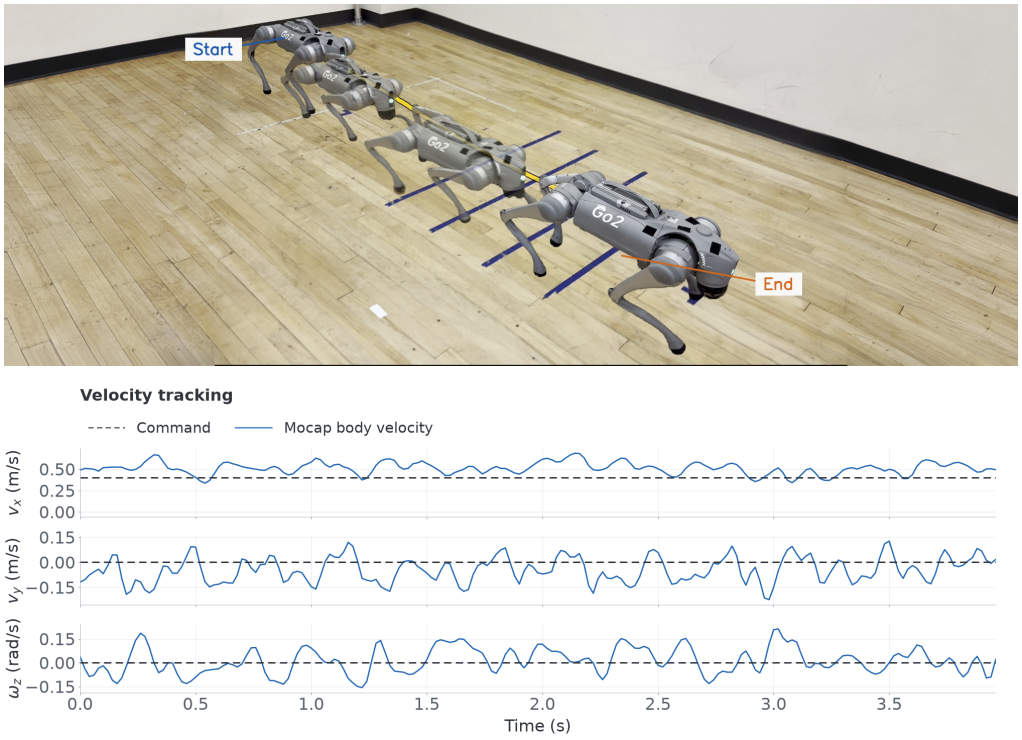
To quantify tracking during figure-eight and straight-line walking, we compare velocity commands with body-frame velocities from motion capture (mocap), as shown in Fig. 19. Using aligned measurements before visualization smoothing, we compute the planar velocity error at each sample as the Euclidean norm of the forward and lateral velocity errors. For yaw angular velocity, we use the absolute difference between measured and commanded values. We then summarize tracking over each plotted interval using mean absolute error (MAE) and root mean square error (RMSE), computed as the mean error magnitude and the square root of the mean squared magnitude, respectively, with equal weight per sample.

For figure-eight and straight-line walking, the planar velocity MAEs are 0.126 and $0.149\,\mathrm{m/s}$, respectively, with RMSEs of 0.144 and $0.161\,\mathrm{m/s}$. The corresponding yaw angular velocity MAEs are 0.112 and $0.068\,\mathrm{rad/s}$, with RMSEs of 0.177 and $0.084\,\mathrm{rad/s}$.

We further assess locomotion stability by applying external pushes during motion. The robot maintains stable locomotion without falling. With only $M = 3$ MPPI refinement iterations per control step, the controller operates at an average frequency of $308.16\,\mathrm{Hz}$, demonstrating high-frequency control under a limited online planning budget. Real-world demonstrations are available at `https://swanfastworldplanner.github.io/`.



(a) **Figure-eight walking.**



(b) **Straight-line walking.**

Figure 19: **Zero-shot sim-to-real velocity tracking on a physical Unitree Go2.** The two panels show figure-eight and straight-line walking after training in simulation. In each panel, the montage visualizes the executed motion, while the curves compare velocity commands (dashed) with body-frame mocap measurements (solid). Tracking errors are summarized in the text using MAE and RMSE over the plotted post-startup intervals.