HaM-World: Soft-Hamiltonian World Models with Selective Memory for Planning

Haoyun Tang[1*] **Haodong Cui**[2*] **Keyao Xu**[3]
Zhandong Mei[1†] **Kun Wang**[4†]
[1]Xi'an Jiaotong University [2]Huazhong University of Science and Technology
[3]Nankai University [4]Nanyang Technological University

Abstract

World models enable model-based planning through learned latent dynamics, but imagined rollouts become unstable as the planning horizon grows or the dynamics distribution shifts: an unconstrained predictor contracts, because each predicted state is fed back as the next input, so the latent distribution shrinks over the rollout until distinct states merge. We argue that this drift and contraction reflect two missing structures in planner-facing latents: a history-conditioned predictive state that summarizes past observations and actions, and geometric organization that separates configuration, momentum, and task semantics. We propose HaM-World (HMW), a structured world model that decomposes the latent state into a canonical-like $(\mathbf{q}, \mathbf{p})$ subspace and a context subspace $\mathbf{c}$, while using Mamba selective state-space memory as the history-conditioned input to the same latent dynamics. Within this interface, $(\mathbf{q}, \mathbf{p})$ evolves through an energy-derived Hamiltonian vector field plus learnable residual/control dynamics, while $\mathbf{c}$ captures semantic, dissipative, and non-conservative factors. This gives the planner a single latent state shared by dynamics prediction, reward/value estimation, imagined rollouts, and CEM action search. On six DeepMind Control Suite tasks, HaM-World ranks first on four and second on the remaining two, with the highest Avg. AUC over all six tasks (91.9, $+10.3\%$ over TD-MPC2). Within the horizon range the planner actually unrolls, it reduces imagined-rollout error to 45% of TD-MPC2's and wins 11/12 $k \in \{3, 5, 7\}$ MSE cells. Under 12 OOD perturbations spanning dynamics shifts, action delay, and observation masking it attains the highest return in every condition ($+10.2\%$ on Finger Spin, $+13.6\%$ on Reacher Easy). Ablation studies show that both selective memory and the soft Hamiltonian prior contribute to improved model performance. Mechanism diagnostics support the intended design, showing bounded action-free energy drift, structured energy variation under policy rollouts, and coherent control-induced energy transfer. Code: https://github.com/HaoyunT/HaM_World.

1 Introduction

In recent years, world models have demonstrated strong capability in representing real-world dynamics through structured latent spaces [1, 2]. However, as the planning horizon increases or the dynamics shift, errors in imagined rollouts accumulate rapidly, leading to instability in long-horizon planning and poor physical-structure extrapolation under perturbed control regimes. This limitation arises because existing approaches primarily capture statistical correlations rather than the underlying generative structure of the environment. Rather than relying solely on correlation fitting, an effective world model should incorporate structure-aware inductive biases that reflect physical regularities, such as invariances and conservation laws [3, 4]. In particular, a monolithic latent representation

*Equal contribution.
†Corresponding authors: Zhandong Mei (zhdmei@mail.xjtu.edu.cn) and Kun Wang (wang.kun@ntu.edu.sg).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

entangles configuration, momentum, and task semantics, causing multi-step prediction errors to grow without structural constraints during planning. An unconstrained predictor also contracts: each imagined state is fed back as the next input, so a small per-step contraction compounds over the planning horizon and the latent distribution narrows until distinct states merge. A Hamiltonian vector field is divergence-free and therefore neither contracts nor expands the distribution to first order, which is why geometric structure is a candidate remedy rather than more capacity [5–7].

We view this limitation as stemming from two missing pieces in planner-facing latent dynamics. **On the input side**, partial observability and action delay make a single-frame latent insufficient to determine the next transition; selective sequence models address this through input-dependent memory [8]. **On the output side**, a unified latent variable entangles configuration, velocity trends, and task semantics, leaving multi-step rollouts without structure-consistent constraints. Physics priors and predictive representations attack related issues from complementary angles [9, 10]. These two sides affect different stages of the same latent dynamics, and strengthening either one alone is unlikely to make the planner improve simultaneously in training return, long-horizon rollout consistency, and physical-structure extrapolation under distribution shifts and perturbations.

Our central view is that *a planning-oriented world model should treat history-conditioned state sufficiency and geometric consistency as two coupled stages of unified latent dynamics*. Memory supplies the conditioning that lets the latent dynamics predict the next transition from a summary of the past rather than a single frame, upon which Soft-Hamiltonian structure in $(\mathbf{q}, \mathbf{p})$ constrains error accumulation over the horizons the planner actually unrolls. The structure does three things for the planner that correlation fitting does not: it counteracts contraction of the latent distribution under repeated prediction; it organizes rollouts around a learned scalar energy H_ϕ, so that energy changes are accounted for by action work rather than by unconstrained drift; and it makes the dynamics partly inspectable, because the learned energy field, its contours, and their coupling to the control signal can be plotted and tested rather than only measured by return. Based on this perspective, we propose HaM-World (HMW), whose unified latent interface $\mathbf{z}_t{=}[\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t]$ is shared across dynamics prediction, reward/value estimation, imagined rollouts, and cross-entropy method (CEM) planning [11]. In the canonical $(\mathbf{q}, \mathbf{p})$ subspace, the Hamiltonian flow field supplies energy-organized local directions, while learnable residual/control dynamics capture controlled deviations; $\mathbf{c}$ captures semantic and non-conservative factors relevant to reward prediction, value estimation, and action search. Mamba selective memory [8] provides history-conditioned inputs to the same latent dynamics, keeping the planner interface unified across training, imagined rollout, and CEM search without a separate recurrent rollout state or planner-specific latent state.

Contributions. (i) **Structured planner-facing dynamics.** We propose HaM-World, a q/p/c latent factorization that couples Mamba selective memory [8] with Soft-Hamiltonian $(\mathbf{q}, \mathbf{p})$ dynamics, giving CEM a single latent interface shared by prediction, reward/value estimation, and action search, and exposing a single learned energy H_ϕ that makes the planner-facing dynamics inspectable; (ii) **Mechanism-aware evaluation.** We evaluate return, rollout MSE for k up to 20, 12 out-of-distribution (OOD) perturbations, and Hamiltonian mechanism analysis on six DeepMind Control Suite (DMC) tasks [12]; (iii) **Empirical performance.** HaM-World reaches the highest Avg. AUC over six tasks (91.9, $+10.3\%$), reduces rollout error *within the planning-relevant range* to 45% of a strong baseline, wins 11/12 $k \in \{3, 5, 7\}$ MSE cells, and obtains the highest absolute return in every OOD condition, with 10.2% and 13.6% average OOD-return gains on Finger Spin and Reacher Easy. Ablations show that selective memory and the Soft-Hamiltonian prior both contribute to improved performance.

2 Related Work

World models, predictive latents, and factorized state. Model-based reinforcement learning learns dynamics that can be queried by a planner or behavior optimizer, from Dyna [13] and World Models [1] to PlaNet/Dreamer [14–16, 2], MuZero [17], TD-MPC [18, 19], MBPO [5], and recent planner-policy, multi-task, and scalable MBRL variants [20–24]. Model-free actor-critic methods such as PPO and SAC remain strong control references [25, 26], but they do not expose latent rollouts. Transformers [27–29], diffusion world models [30, 31], large-scale generative or multimodal environments [32–34], operator-learning views [35], and policy-shaped prediction [36] expand the transition-model design space; the taxonomy in Ding et al. [37] places these families in a broader world-model landscape. Reconstruction-free learning further replaces pixel reconstruction

with objectives closer to prediction and decision making: JEPA-style models [10, 38–40], invariant or control-oriented representations [41], Dreaming/DreamerPro [42, 43], and TD-JEPA [44] all suggest that representation objectives should be aligned with downstream behavior. Orthogonally, latent factorization has been used to improve control and generalization, including SOLAR [45], Denoised MDPs [6], IFactor [7], Plan2Explore [46], and recent semantic/dynamic or long-horizon decompositions [47–50]. HaM-World combines these threads in a narrower planner-facing form: the latent is not only predictive, but explicitly split into coordinates used by reward/value estimation, imagined rollouts, and CEM action search throughout training and evaluation.

Physics priors and memory for long-horizon rollouts. Hamiltonian Neural Networks [3], Symplectic ODE-Net [4], action-conditioned Hamiltonian models [9], stable port-Hamiltonian networks [51], and embodied physics-prior world models [52] show that energy-based geometric structure can improve stability, plausibility, and extrapolation. Strict conservation is nevertheless too restrictive for planning in contact-rich controlled systems, where friction, actuation, task semantics, observation noise, and reward-relevant context create non-conservative effects. HaM-World therefore uses a Soft-Hamiltonian prior on $(\mathbf{q}, \mathbf{p})$: the Hamiltonian vector field gives a structured backbone, residual/control dynamics absorb controlled deviations, and $\mathbf{c}$ carries semantic and dissipative context. This differs from simply adding smoothness constraints such as spectral normalization [53], because the prior is placed on the state actually queried by the planner. On the input side, partial observability makes a single observation insufficient to determine the next transition, so history-dependent representations are necessary; recurrent state-space models, Transformers, and state-space sequence models (SSMs) address this through learned memory [14, 27, 54–56]. For planner rollouts, however, the memory must be both efficient over long horizons and adaptive to the current action-conditioned transition: RNN/GRU memory offers a compact recurrent state with input-dependent gates, while Transformer memory can model rich context at higher sequence cost. Structured SSMs provide recurrent long-context filtering for RL, and Mamba further makes the state-space update input-selective [54, 8, 57, 55, 56]. We therefore use Mamba as a history-conditioned input to the same latent dynamics, forming a history-conditioned predictive state within the planner interface.

3 Preliminaries and Problem Setup

We consider continuous-control trajectories $\tau = \{(o_t, \mathbf{a}_t, r_t)\}_{t=0}^{T}$ with observations o_t, actions $\mathbf{a}_t$, and rewards r_t. A planner-facing world model maps observations to latents and rolls them forward under candidate actions over a finite planning horizon,

$$\mathbf{z}_t = E_\phi(o_t), \qquad (\hat{\mathbf{z}}_{t+1}, \mathbf{h}_{t+1}) = F_\phi(\hat{\mathbf{z}}_t, \mathbf{a}_t, \mathbf{h}_t), \qquad \hat{r}_t = R_\phi(\hat{\mathbf{z}}_t, \mathbf{a}_t), \quad \hat{V}_t = V_\phi(\hat{\mathbf{z}}_t). \tag{1}$$

Here $E_\phi, F_\phi, R_\phi, V_\phi$ are the encoder, transition, reward, and value heads; $\hat{\mathbf{z}}_t$ denotes the model-predicted latent for imagined rollouts, and $\mathbf{h}_t$ is the history state. At decision time, CEM searches action sequences by maximizing model-imagined return from predicted rewards and terminal value,

$$\mathbf{a}^{\star}_{t:t+H-1} \in \arg\max_{\mathbf{a}_{t:t+H-1}} \sum_{k=0}^{H-1} \gamma^k \hat{r}_{t+k} \; + \; \gamma^H \hat{V}_\phi(\hat{\mathbf{z}}_{t+H}), \quad \hat{\mathbf{z}}_{t+k+1} = F_\phi(\hat{\mathbf{z}}_{t+k}, \mathbf{a}_{t+k}, \mathbf{h}_{t+k}). \tag{2}$$

The central design question is therefore not only how to learn an accurate one-step predictor, but how to choose a latent state $\mathbf{z}_t$ whose repeated rollout remains stable under partial observations and directly useful for reward/value prediction and action search.

The planner-facing state. Strict predictive sufficiency would require a statistic $s_t = \psi(\mathcal{H}_t)$ of the history $\mathcal{H}_t = (o_{\le t}, \mathbf{a}_{<t})$ with $p(o_{t+1}, r_t \mid \mathcal{H}_t, \mathbf{a}_t) = p(o_{t+1}, r_t \mid s_t, \mathbf{a}_t)$. We use $s_t = (\mathbf{z}_t, \mathbf{h}_{t-1})$ as the planner-facing state, where $\mathbf{z}_t = E_\phi(o_t)$ encodes the current observation and $\mathbf{h}_t = \mathrm{Memory}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_{t-1})$ is a fixed-size summary of the past that conditions the transition. The no-memory and GRU ablations (Section 7) and the multi-step rollout and reward errors test whether this state retains what finite-horizon planning needs. "History-conditioned" refers throughout to this construction.

What $(\mathbf{q}, \mathbf{p})$ and H mean in a learned latent space. Classical mechanics describes a system by its configuration q and conjugate momentum p. One scalar $H(q, p)$, the energy, determines both updates: $\dot{q} = \partial_p H$ says position changes at a rate set by momentum, and $\dot{p} = -\partial_q H$ says momentum

changes with the energy slope along position, so motion follows energy contours instead of drifting — the property we want when a rollout is fed back into itself many times. In a learned latent space, $\mathbf{q}$ and $\mathbf{p}$ are *learned* canonical-like coordinate blocks rather than dimension-wise physical quantities, H_ϕ is an energy-like scalar field rather than identified mechanical energy, and we borrow only the *directions* of the flow: $\xi_H = (\partial_{\mathbf{p}} H_\phi, -\partial_{\mathbf{q}} H_\phi)$ satisfies $\nabla H_\phi^\top \xi_H = 0$, so moving along it leaves the learned energy unchanged to first order — exactly what repeated unrolling needs. Strict conservation is neither required nor desirable here: the learned residual fits dissipative dynamics, the control drive injects action work, and $\mathbf{c}$ carries non-Hamiltonian information. Appendix A.2 expands on this reading.

4 Method

HaM-World instantiates the planner-facing world model above with a q/p/c latent split and a single shared dynamics interface. Actions are searched by CEM inside the learned dynamics. The overall architecture is shown in Figure 1.

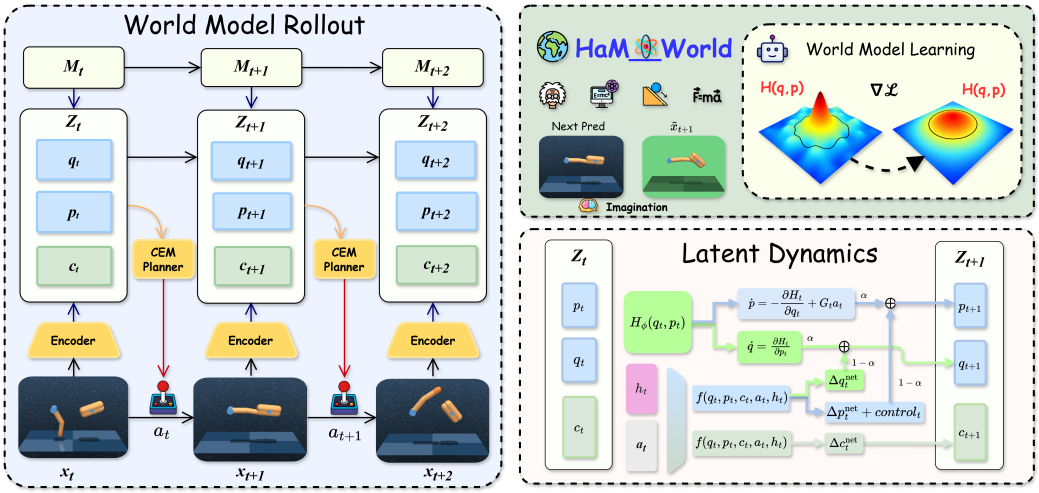


Figure 1: Architecture of HaM-World. Observations are encoded into Hamiltonian state $(\mathbf{q}_t, \mathbf{p}_t)$, context $\mathbf{c}_t$, and Mamba memory $\mathbf{h}_t$; a shared latent transition supports reward/value prediction and CEM planning inside one unified planner-facing interface.

4.1 q/p/c-Structured Latents and Soft-Hamiltonian Pair Dynamics

Long-horizon planning asks the same latent dynamics to satisfy two requirements. On one hand, repeated planner rollouts need a stable geometric backbone. On the other, real control tasks contain contact, friction, reward semantics, and partial observability. If all of these factors are compressed into a single unstructured latent, the model can oscillate between short-term fitting and long-term rollout stability. HaM-World therefore writes the encoder output as $\mathbf{z}_t = E_\phi(o_t) = [\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t]$: $(\mathbf{q}, \mathbf{p})$ form a canonical subspace and $\mathbf{c}$ forms a context subspace. The reward head, value head, imagined rollout, and CEM planner all share the same $\mathbf{z}_t$, so this division of roles acts directly on the state queried by the planner, rather than as a training-only or post-hoc explanatory variable. The Soft-Hamiltonian pair dynamics balance structural stability and expressive flexibility. A generic MLP latent dynamics lacks energy-field constraints, whereas a strictly Hamiltonian model over-constrains dissipative controlled dynamics. We therefore allow only $(\mathbf{q}, \mathbf{p})$ to form the main dynamics backbone through a learnable energy field $\mathcal{H} = H_\phi(\mathbf{q}, \mathbf{p})$, while augmenting that pair with residual/control dynamics and using $\mathbf{c}$ to represent complementary semantic and non-conservative context. The controlled Hamiltonian equations $\dot{\mathbf{q}} = \partial\mathcal{H}/\partial\mathbf{p}$, $\dot{\mathbf{p}} = -\partial\mathcal{H}/\partial\mathbf{q} + g(\mathbf{q})\mathbf{a}$ [4, 9] decompose internal energy geometry from action work; HaM-World discretizes this structure and embeds it into the planner-facing dynamics used by CEM during every imagined rollout within the shared transition interface.

Why an energy-organized field. The difficulty is repeated self-iteration: to score one candidate the planner feeds the model's own output back in for H steps, and an unconstrained transition can then *collapse* trajectories into a low-dimensional subspace, making candidates indistinguishable to the reward/value heads, or *expand* them outside the training region, where those heads score unreliably. The raw Hamiltonian field addresses exactly these two modes: it is orthogonal to the energy gradient and divergence-free, so in continuous time it neither changes H nor compresses or expands phase-space volume, and a single H_ϕ drives both updates so the coordinates are jointly constrained (Appendix A.3). With a residual, control drive, context channel, and forward Euler these survive only as a bias.

Hamiltonian-pair update. Given the Mamba memory output $\mathbf{h}_t = \mathrm{Memory}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_{t-1})$, the Soft-Hamiltonian update for the Hamiltonian pair produces

$$\begin{aligned} \left[\Delta\mathbf{q}_t^{\text{net}}, \Delta\mathbf{p}_t^{\text{net}}\right] = f_{\text{core}}(\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t, \mathbf{a}_t, \mathbf{h}_t), \quad \mathcal{H}_t = H_\phi(\mathbf{q}_t, \mathbf{p}_t), \quad \mathbf{G}_t = G_\phi(\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t, \mathbf{a}_t, \mathbf{h}_t), \\ \Delta\mathbf{q}_t = (1-\alpha)\,\Delta\mathbf{q}_t^{\text{net}} + \alpha\,\partial_{\mathbf{p}}\mathcal{H}_t, \quad \Delta\mathbf{p}_t = (1-\alpha)\,\Delta\mathbf{p}_t^{\text{net}} - \alpha\,\partial_{\mathbf{q}}\mathcal{H}_t + \mathbf{G}_t\mathbf{a}_t, \end{aligned} \tag{3}$$

followed by $\mathbf{q}_{t+1}{=}\mathbf{q}_t{+}\Delta\mathbf{q}_t$ and $\mathbf{p}_{t+1}{=}\mathbf{p}_t{+}\Delta\mathbf{p}_t$. The term "Soft-Hamiltonian" reflects two design choices: α mixes the action-conditioned network update with the Hamiltonian vector field on the canonical pair, while $\mathbf{G}_t\mathbf{a}_t$ adds an explicit learned control drive to the momentum update. As $\alpha{\to}0$, the pair update degenerates to a data-driven update plus control; as $\alpha{\to}1$, the canonical part approaches a controlled Hamiltonian update with the same control channel. Intermediate α keeps action-free energy drift small when the network update is aligned with the Hamiltonian direction, while retaining the ability to fit contact, friction, and task objectives during closed-loop planning across horizons and perturbations seen by the planner during model-predictive control rather than only one-step prediction in isolation during training.

Eq. (3) describes the update before Hamiltonian-gradient clipping and latent norm projection; Appendix B.7 specifies these safeguards.

Context and selective memory. The context variable $\mathbf{c}$ is introduced because not all control-relevant information should obey the Hamiltonian energy field. Forcing contact switches, friction, task semantics, or observation noise into $(\mathbf{q}, \mathbf{p})$ can damage the stabilizing role of the energy field; discarding them would harm reward/value prediction and planning. The context update is $\Delta\mathbf{c}_t = f_c(\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t, \mathbf{a}_t, \mathbf{h}_t)$, so $\mathbf{c}$ remains available within the unified latent dynamics as complementary information for these non-Hamiltonian factors. "Context" names an architectural role: a channel not required to follow the vector field of H_ϕ. The Mamba selective scan [8, 57] outputs $\mathbf{h}_t$ only as a *history-conditioned input* to the same latent dynamics, entering f_{core}, G_ϕ, and f_c simultaneously. The model has one planner-facing latent dynamics; memory supplies the missing history information under partial observability, action delay, and long horizons during imagined search.

4.2 Mechanism-Oriented Stability Analysis

We package the mechanism as a local statement about the map the planner actually iterates, together with the diagnostics it predicts. Accumulated prediction error over multiple steps is characterized empirically by the multi-horizon rollout errors in Section 6 and Appendix C.

Soft-Hamiltonian Mechanism. The planner unrolls the same transition used in training, including memory and clipping safeguards (Appendix B.7). On steps without Hamiltonian-gradient clipping or latent norm projection, this is the unroll of Eq. (3) (Appendix A.1). The raw Hamiltonian field is energy-orthogonal and divergence-free on the canonical pair (Eq. (13)); on such steps, control and the alignment residual are the only first-order sources of energy change. Active clipping or projection can add a first-order contribution (Appendix A.3). The corresponding diagnostics are slow action-free energy drift and push-dependent contour crossing, tested empirically in Section 6.3 and Appendix D.1.

4.3 Training Objective and Planning

The training objectives are designed to approximate the desirable behaviors characterized in Section 4.2, rather than enforcing them as strict constraints. The total loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{repr}} + \beta_{\text{dyn}}\mathcal{L}_{\text{dyn}} + \beta_{\text{roll}}\mathcal{L}_{\text{roll}} + \beta_{\text{r}}\mathcal{L}_{\text{reward}} + \beta_{\text{v}}\mathcal{L}_{\text{value}} + \beta_{\text{p}}\mathcal{L}_{\text{policy}} + \beta_{\text{geo}}\mathcal{L}_{\text{geo}}, \tag{4}$$

where $\mathcal{L}_{\text{repr}}$ is a JEPA-style online/EMA latent alignment objective [10, 38], $\mathcal{L}_{\text{dyn}}$ is a one-step latent consistency loss, reward/value use DreamerV3-style symlog two-hot regression with EMA value targets [2], and $\mathcal{L}_{\text{policy}}$ trains the CEM warm-start prior by behavior cloning. Below we highlight three representative losses that are directly aligned with the Soft-Hamiltonian structure; all other terms and weights are in Appendix B.

Multi-step rollout consistency.

$$\mathcal{L}_{\text{roll}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} \left\| \hat{\mathbf{z}}_{s+k}^{(s)} - \text{sg}(\mathbf{z}_{s+k}) \right\|_2^2, \tag{5}$$

where $\hat{\mathbf{z}}_{s+k}^{(s)}$ is the k-step prediction rolled out from $\mathbf{z}_s$ with $\mathbf{h}_s$ as the initial memory condition. This term provides a discrete surrogate for controlling multi-step error accumulation, directly constraining the rollout horizon relevant for planning.

Hamiltonian alignment.

$$\mathcal{L}_{\text{ham}} = \left\| \Delta \mathbf{q}_t^{\text{net}} - \partial_{\mathbf{p}} \mathcal{H}_t \right\|_2^2 + \left\| \Delta \mathbf{p}_t^{\text{net}} + \partial_{\mathbf{q}} \mathcal{H}_t \right\|_2^2. \tag{6}$$

This loss acts as a soft directional bias, aligning the network branch with the Hamiltonian vector field while leaving the learned control drive outside the alignment residual.

Action-free energy regularization.

$$\mathcal{L}_{\text{energy}} = \mathbb{E}_{t:\|\mathbf{a}_t\|<\epsilon} \left[\left(\mathcal{H}(\mathbf{q}_{t+1}, \mathbf{p}_{t+1}) - \mathcal{H}(\mathbf{q}_t, \mathbf{p}_t) \right)^2 \right]. \tag{7}$$

This term directly suppresses energy drift in the action-free setting, encouraging the stable behavior described in Section 4.2 for passive rollouts.

The geometric regularization term is defined as

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{ham}} \mathcal{L}_{\text{ham}} + \lambda_{\text{en}} \mathcal{L}_{\text{energy}} + \lambda_{\text{sa}} \mathcal{L}_{\text{sa}} + \lambda_{\text{temp}} \mathcal{L}_{\text{temp}} + \lambda_{\text{dec}} \mathcal{L}_{\text{dec}} + \lambda_{\text{c}} \mathcal{L}_{\text{c}}.$$

Full formulas and weights are provided in Appendix B. The planner performs horizon-H CEM optimization over $(\mathbf{z}_t, \mathbf{h}_t)$, searching action sequences using the shared latent dynamics, reward head, and value head.

5 Experimental Setup

We evaluate on a four-task core suite — Reacher Easy, Finger Spin, Cheetah Run, and Cartpole Swingup — from the DeepMind Control Suite [12], plus two additional tasks, Walker Run and Hopper Hop, chosen because the core suite underrepresents sustained coordination, momentum, and repeated contact. All six use state observations, 100k environment steps, and 3 seeds (7/8/9). We report final return, AUC, latent-rollout MSE for $k \in \{3, 5, 7\}$ in the main text and k up to 20 in Appendix C, and zero-shot OOD return. The OOD protocol uses 12 perturbations over Reacher Easy and Finger Spin, covering dynamics changes, action delay, and observation masking; full task and evaluation details are in Appendix B.

Baselines. We compare model-free actor-critic baselines (PPO [25], SAC [26]) with model-based world models (DreamerV3 [2], TD-MPC2 [19]). PPO and SAC lack explicit latent rollouts, so they are omitted from latent-rollout MSE. Methods using external pretraining or visual encoders are excluded.

6 Main Results

We present the evidence as a sequence of observations: standard-control return, planner-relevant rollout consistency, zero-shot OOD return, and finally the Hamiltonian mechanism diagnostics that connect the gains to the proposed memory-and-geometry design, summarized here and given in full in Appendix D.1.

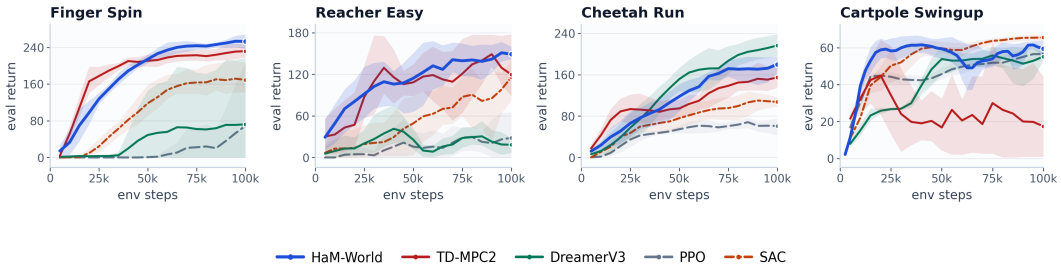


Figure 2: Return learning curves on the four core tasks (mean curve with seed envelope). HaM-World obtains the highest final return on Finger and Reacher and the second-best on Cheetah and Cartpole; summarized by training AUC it obtains the highest average score. Final returns for Walker Run and Hopper Hop are in Appendix D.2.

6.1 Control Performance and Long-Horizon Consistency

Table 1 groups two core metrics by row: (I) final return at 100k environment steps and Avg. AUC; (II) $k \in \{3, 5, 7\}$-step imagined-rollout latent MSE. We report coordinate-summed latent error as the main latent-MSE metric because planning/reward/value heads consume the full latent vector, so total rollout deviation, rather than only per-coordinate drift, matters for imagined planning. **Bold**/underline denote the best/second-best value in each column within each task block of the table.

Table 1: Main table (mean±std, 3 seeds): (I) final return and average AUC; (II) $k \in \{3, 5, 7\}$-step latent-rollout MSE, applicable only to model-based methods. Two further tasks (Walker Run, Hopper Hop) and the Avg. AUC over all six tasks are reported in Appendix D.2, and k up to 20 in Appendix C.

Method	Finger Spin	Reacher Easy	Cheetah Run	Cartpole Swingup	Avg.
(I) Final return @ 100k env steps ↑ (Avg. = AUC over training, core suite)					
PPO	70.3±99.5	18.9±15.1	67.5±17.4	56.8±2.5	30.3
SAC	173.6±26.1	<u>118.9</u>±3.9	108.5±6.5	**65.8**±1.3	70.9
DreamerV3	72.3±102.3	18.4±7.9	**216.4**±17.0	55.8±0.4	58.6
TD-MPC2	<u>232.2</u>±8.2	105.1±52.3	155.7±17.4	15.0±17.0	<u>107.7</u>
HaM-World	**254.0**±14.1	**150.6**±7.5	<u>184.4</u>±8.4	<u>58.9</u>±3.8	**117.9**

(II) k-step latent rollout MSE ↓ (model-based methods only)													
	k=3	k=5	k=7	k=3	k=5	k=7	k=3	k=5	k=7	k=3	k=5	k=7	
DreamerV3	7.20	9.29	11.51	10.46	14.26	18.22	13.29	18.17	24.63	7.25	8.59	10.34	12.77
TD-MPC2	3.48	3.58	3.61	3.34	3.38	**3.42**	4.28	4.40	4.47	4.66	4.79	4.87	4.02
HaM-World	**0.72**	**1.43**	**2.53**	**1.28**	**2.36**	3.68	**1.21**	**1.88**	**2.74**	**0.77**	**1.30**	**1.97**	**1.82**

Observation 1: HaM-World improves sample efficiency without sacrificing final control. As shown in Table 1 and Figure 2, HaM-World reaches the highest Avg. AUC (117.9) on the core suite, improving over TD-MPC2 by +9.5% (117.9 vs. 107.7), SAC by +66.3%, and DreamerV3 by +101.2%. Pooling all six tasks (Table 14), its Avg. AUC is 91.9 against 83.3 for TD-MPC2 (+10.3%), 52.8 for SAC (+74.1%), and 51.7 for DreamerV3 (+77.8%). In final return, it is best on Finger Spin (254.0±14.1) and Reacher Easy (150.6±7.5), and second best on Cheetah Run (184.4±8.4) and Cartpole Swingup (58.9±3.8). On the two additional tasks it is best on both (Appendix D.2), so over six tasks it ranks first on four and second on two. The gains are largest where control requires sustained coordination and repeated contact.

Observation 2: rollout error is much lower within the planning-relevant range. The model-based baselines show different failure modes in Table 1. DreamerV3 grows sharply with horizon, e.g., Cheetah MSE rises from 13.29 to 24.63 as k moves from 3 to 7, indicating unstable extrapolation. TD-MPC2 grows slowly ($< 5\%$ from k=3 to k=7 on each task), but stays on a high ∼3–5 error plateau. In contrast, HaM-World wins 11/12 MSE cells and reaches Avg. MSE 1.82, only 45% of TD-MPC2 (4.02) and 14.3% of DreamerV3 (12.77).

The range $k \in \{3, 5, 7\}$ is what each method's native decision procedure consumes (our H=6, TD-MPC2 H=5, DreamerV3 H=8). Extending to k=10, 15, 20 reveals task-dependent crossovers: our error starts lower but has a steeper slope, and are lowest on 4/4, 4/4, 3/4, 2/4, 1/4 tasks at k=3, 5, 7, 10, 15 and on 0/4 at k=20, with crossovers at Reacher (k=7), Finger (10), Cheetah (15) and Cartpole (20). These results locate the rollout advantage *within the planning-relevant range*.

Appendix C gives the full curves and shows why this suffices: closed-loop replanning re-anchors every real step, and raising the CEM horizon past 6–14 degrades return on three of four tasks at 3.2–4.6× the latency.

6.2 OOD Generalization

The OOD evaluation focuses on **Finger Spin** and **Reacher Easy**. They cover contact-dominated spinning and end-effector reaching, with 6 perturbation axes per task: dynamics changes (mass, damping, friction, actuator) plus partial-observability settings (delay and masking).

Table 2: OOD return across 12 perturbations (mean±std, 3 seeds; higher is better). **Bold** marks the best value in each column.

	Finger Spin						
Method	fric×0.5	fric×1.5	mass×1.3	mass×1.5	delay=2	mask 30%	Avg.
HaM-World	**242.8**±8.5	**253.1**±10.4	**232.6**±10.2	**211.2**±10.7	**74.4**±4.5	**91.5**±3.5	**184.3**±4.0
TD-MPC2	227.9±16.2	227.7±13.5	220.2±10.6	196.3±7.6	47.9±2.6	84.0±4.7	167.3±7.8
SAC	152.6±15.4	175.8±31.1	141.1±14.1	129.5±16.4	58.0±14.9	53.2±10.2	118.4±13.1
DreamerV3	65.0±92.0	75.4±106.6	59.7±84.4	58.7±83.1	26.2±37.1	45.9±64.9	55.2±78.0
PPO	67.4±95.4	75.3±106.5	63.1±89.2	58.3±82.4	6.3±9.0	13.5±19.1	47.3±66.9
	Reacher Easy						
Method	mass×0.7	mass×1.3	damp×0.5	damp×2.0	act×0.7	act×1.3	Avg.
HaM-World	**158.5**±5.9	**158.8**±9.4	**141.7**±4.1	**139.7**±9.3	**148.8**±7.5	**151.8**±1.5	**149.9**±5.6
TD-MPC2	135.3±35.6	130.1±42.4	131.8±46.7	125.1±29.6	131.5±32.5	138.1±36.9	132.0±37.2
SAC	98.5±12.7	98.0±15.1	86.0±10.1	83.0±9.5	88.6±12.9	100.2±9.3	92.4±8.7
DreamerV3	8.7±2.8	11.4±5.7	9.3±2.1	8.1±3.7	8.9±5.2	9.4±3.6	9.3±3.5
PPO	11.7±9.6	11.8±9.7	14.6±14.4	13.9±11.2	12.7±9.7	12.9±11.6	12.9±10.9

Observation 3: HaM-World obtains the highest absolute return under all 12 OOD perturbations. Table 2 shows that HaM-World obtains the highest return in every OOD column. On Finger Spin, its OOD average is $184.3_{\pm 4.0}$, exceeding TD-MPC2 ($167.3_{\pm 7.8}$, $+10.2\%$) and SAC ($118.4_{\pm 13.1}$, $+55.7\%$), while also leading under delay$=2$ (74.4) and mask 30% (91.5). On Reacher Easy, it reaches $149.9_{\pm 5.6}$ versus TD-MPC2's $132.0_{\pm 37.2}$ ($+13.6\%$), with much lower variance.

Normalized retention for all methods is reported in Appendix B.13. Mechanistically, the OOD results are consistent with memory absorbing the shifted observation–action response into a runtime recurrent state while $(\mathbf{q}, \mathbf{p})$ encodes relations that stay valid after rescaling; both are zero-shot, with no parameter update at deployment.

6.3 Hamiltonian Mechanism Diagnostics

The Soft-Hamiltonian design predicts two falsifiable behaviours (Section 4.2): action-free rollouts should barely change the learned energy, while action work should move the state across energy layers. Both hold on Cheetah Run. Under zero external force and zero damping the energy trace stays flat to about 1% drift within episodes and widens immediately once random actions inject work; under policy rollouts H_t varies with the ongoing behaviour without abrupt jumps, separating passive energy stability from policy-induced modulation. Projecting rollouts onto a canonical slice, energy change tracks the control projection $\mathrm{push}_t = (\nabla_{\mathbf{p}} \mathcal{H}_t) \cdot \mathrm{control}_t$, with $\mathrm{corr}(\mathrm{sign}\, \Delta\mathcal{H}, \mathrm{push}) = 0.757$ and $\mathrm{corr}(|\Delta\mathcal{H}|, |\mathrm{push}|) = 0.745$, and a threshold sweep keeps the High-$|\mathrm{push}|$ contour-crossing rate above the Low-$|\mathrm{push}|$ one throughout (lift_auc $= 0.595$). The canonical and context coordinates stay distinguishable rather than collapsing, with $\mathbf{q}, \mathbf{p}$ forming elongated manifolds and $\mathbf{c}$ a compact cloud. Appendix D.1 gives all figures, the non-conjugate control slice, and the protocol.

6.4 Result Summary

Across metrics, the evidence supports the coupled design in Section 1. HaM-World has the highest Avg. AUC on the core suite (117.9) and over all six tasks (91.9), and the highest return on both additional tasks, wins 11 out of 12 cells in $k \in \{3, 5, 7\}$ MSE, and obtains the highest absolute return under all 12 OOD conditions. The Cheetah Run diagnostics (Section 6.3, Appendix D.1) show near-conservation without action, structured energy variation during policy rollouts, separated q/p/c

geometry, and action push modulating energy-layer crossing. Thus, selective memory supplies the input-side history-conditioned state, while Soft-Hamiltonian geometry on $(\mathbf{q}, \mathbf{p})$ supplies output-side rollout consistency within the same planner-facing latent dynamics used by CEM.

The gains are consistent across control, planning-relevant rollouts, and absolute OOD return, supporting the combination of selective memory and Soft-Hamiltonian structure.

7 Ablation Study

The ablation study analyzes *two mechanisms acting at different stages of the same latent dynamics*, rather than asking which one is important in isolation. Table 3 evaluates: (A1) removing the q/p/c split and the Soft-Hamiltonian q/p bias as one geometric mechanism, since the Hamiltonian constraint loses its target without the split; (A2) removing memory; and (A3) replacing Mamba memory with a GRU. The last two columns add representative Reacher Easy OOD perturbations; Full-model OOD entries reuse the reference evaluation in Table 2.

Table 3: Ablations (3 seeds, mean±std). *Upper block:* the two mechanisms — A1 removes the q/p/c split together with the whole geometric loss package, A2 removes memory, A3 replaces Mamba with a GRU. *Lower block:* isolation of the three additive parts of the pair update in Eq. (3) — A4 makes it purely Soft-Hamiltonian, A5 sets α=1 (field kept, residual dropped), A6 deletes only the control port $\mathbf{G}_t\mathbf{a}_t$. MSE@6 is per-coordinate here, so its scale differs from Table 1.

	Return ↑		MSE@6 ↓		Reacher OOD ↑	
Variant	Cheetah	Finger	Cartpole	Reacher	mass×0.7	damp×2.0
Full HaM-World	**184.4**±8.4	**254.0**±14.1	**0.009**±0.003	**0.0778**±0.043	**158.5**±5.9	**139.7**±9.3
Memory and geometry as two mechanisms						
A1: w/o geom. struct.	169.8±13.5	247.7±12.5	0.012±0.003	0.0852±0.022	154.4±15.2	131.9±22.9
A2: Memory = None	59.2±11.2	34.0±23.8	0.110±0.068	0.287±0.075	103.6±5.1	115.4±9.0
A3: Memory = GRU	121.1±32.1	235.6±4.6	0.026±0.007	0.148±0.021	<u>157.6</u>±7.7	<u>138.4</u>±3.0
Isolating the parts of the Soft-Hamiltonian pair update						
A4: pure Soft-Ham.	<u>181.7</u>±9.2	<u>251.3</u>±15.6	<u>0.010</u>±0.003	<u>0.080</u>±0.031	156.1±6.8	136.7±8.5
A5: hard (α=1)	158.6±17.1	226.7±19.3	0.034±0.010	0.119±0.029	138.2±13.6	121.5±12.4
A6: no port $\mathbf{G}$	153.6±18.2	219.3±24.5	0.041±0.009	0.129±0.028	130.7±12.1	115.4±10.8

Removing memory (A2) reduces performance: Cheetah/Finger return drops to 59.2/34.0, Cartpole/Reacher MSE@6 rises from $0.009/0.0778$ to $0.110/0.287$, and Reacher OOD return falls to $103.6/115.4$. A3 (Mamba→GRU) restores part of the return, but MSE@6 remains higher than the Full model ($0.026/0.148$ vs. $0.009/0.0778$), indicating a weaker history summary than selective SSM memory. Removing geometric structure (A1) also reduces return and OOD scores (184→170, 254→248; 158.5→154.4, 139.7→131.9).

Complementary roles of memory and geometry. Ablations support the complementary contributions of history-conditioned memory and Soft-Hamiltonian geometry, whose effects are coupled within the shared dynamics: memory reduces state ambiguity from history, and the $(\mathbf{q}, \mathbf{p})$ dynamics then organize candidate futures from that state, so A2 removes the geometry's input together with the history summary. A1 removes the q/p/c split together with $\mathcal{L}_{\text{ham}}, \mathcal{L}_{\text{energy}}, \mathcal{L}_{\text{sa}}, \mathcal{L}_{\text{temp}}, \mathcal{L}_{\text{dec}}, \mathcal{L}_{\text{c}}$ while keeping capacity, planner, recurrence, core prediction and reward/value objectives, and budget fixed, and therefore measures the geometric package as a whole.

Why Mamba rather than another recurrent module. Mamba outperforms the GRU for three reasons. *Input-dependent updates*: GRU gates also vary with the input and hidden state; the learned weights, rather than the gate values, are fixed after training. Our selective SSM instead derives step sizes and state/readout projections from the current $(\mathbf{z}_t, \mathbf{a}_t)$ within a diagonal recurrence. *History retention*: the diagonal recurrence accumulates history with input-dependent decay, while a GRU uses nonlinear gated updates. *Batched-rollout compatibility*: both maintain a fixed-size recurrent state that can condition each imagined transition without reprocessing a token history. A Transformer mixer is worse on 4/4 tasks (Appendix B.10).

Isolating the Hamiltonian field, the residual, and the control port. The lower block of Table 3 separates the three additive parts of Eq. (3) with everything else fixed. A4 degrades slightly and uniformly, supporting the hybrid form. A5 degrades markedly — keeping the energy field while deleting the residual over-constrains the dynamics, the concrete cost of a strict rather than soft prior and the main justification for "Soft". A6 degrades most: without the port, all action influence must pass through the residual branch, so a direct action-to-dynamics path matters for actuated tasks.

8 Conclusion

HaM-World decomposes planner-facing latent dynamics into an input-side history-conditioned state through Mamba selective memory and output-side geometric consistency through a Soft-Hamiltonian bias on $(\mathbf{q}, \mathbf{p})$. On six DMC tasks it ranks first on four and second on two, reaches the highest Avg. AUC (117.9 on the core suite, 91.9 over all six tasks), wins 11 out of 12 cells for $k \in \{3, 5, 7\}$ MSE, and obtains the highest absolute return across all 12 OOD conditions. Ablations and geometry diagnostics support complementary contributions from memory and Soft-Hamiltonian structure in rollout error and OOD planning quality; $\mathbf{c}$ carries semantic, contact, and dissipative information without being part of the Soft-Hamiltonian q/p update. The results suggest evaluating planner-facing representations not only by return, but also by whether imagined trajectories retain action-free invariants and respond coherently to action work.

Limitations and Future Directions. The evaluation covers six state-based DMC tasks; manipulation, contact-rich robotics, and visual observations are directions for future evaluation. A visual encoder can replace the MLP encoder without changing the $(\mathbf{q}, \mathbf{p}, \mathbf{c})$ dynamics, memory, or planner; how stable canonical-like $(\mathbf{q}, \mathbf{p})$ structure can be learned from pixels, and how visual nuisance variation divides against $\mathbf{c}$, remains to be tested. The prior is a first-order local bias under forward Euler, weaker than symplectic structure. Future work should test the same q/p/c split from pixels, evaluate on manipulation and contact-rich benchmarks, and use separable or mechanical parameterizations of H_ϕ to distinguish local energy organization from true Hamiltonian structure.

Acknowledgments and Disclosures

Funding. This work received no external funding or third-party financial or in-kind support during the past 36 months.

Competing interests. The authors declare no competing financial interests or financial relationships outside the submitted work that could be perceived as influencing it.

References

[1] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

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

[3] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[4] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

[5] 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.

[6] Tongzhou Wang, Simon Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings*

of the 39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 22591–22612. PMLR, 17–23 Jul 2022.

[7] Yuren Liu, Biwei Huang, Zhengmao Zhu, Honglong Tian, Mingming Gong, Yang Yu, and Kun Zhang. Learning world models with identifiable factorization. *Advances in Neural Information Processing Systems*, 36:31831–31864, 2023.

[8] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[9] Arne Troch, Kevin Mets, and Siegfried Mercelis. Action-conditioned hamiltonian generative networks (ac-hgn) for supervised and reinforcement learning. In *7th Annual Learning for Dynamics & Control Conference, 04-06 June, 2025, Ann Arbor, Michigan, USA*, pages 310–322, 2025.

[10] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.

[11] Reuven Y Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[12] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[13] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[14] 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*, pages 2555–2565. PMLR, 2019.

[15] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[16] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[17] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[18] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022.

[19] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 54241–54259, 2025.

[22] Ignat Georgiev, Varun Giridhar, Nicklas Hansen, and Animesh Garg. PWM: Policy learning with multi-task world models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] ZiRui Wang, Yue Deng, Junfeng Long, and Yin Zhang. Parallelizing model-based reinforcement learning over the sequence length. *Advances in Neural Information Processing Systems*, 37: 131398–131433, 2024.

[24] Bhavya Sukhija, Lenart Treven, Carmelo Sferrazza, Florian Dorfler, Pieter Abbeel, and Andreas Krause. SOMBRL: Scalable and optimistic model-based RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[26] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[27] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:27147–27166, 2023.

[29] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models. *Advances in Neural Information Processing Systems*, 37:68082–68119, 2024.

[30] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

[31] Jia-Hua Lee, Bor-Jiun Lin, Wei-Fang Sun, and Chun-Yi Lee. EDELINE: Enhancing memory in diffusion-based world models via linear-time sequence modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[32] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[33] Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Aaron Courville, and Sai Rajeswar. Genrl: Multimodal-foundation world models for generalization in embodied agents. *Advances in neural information processing systems*, 37:27529–27555, 2024.

[35] Pietro Novelli, Marco Pratticò, Massimiliano Pontil, and Carlo Ciliberto. Operator world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 111432–111463, 2024.

[36] Miles Hutson, Isaac Kauvar, and Nick Haber. Policy-shaped prediction: avoiding distractions in model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 13124–13148, 2024.

[37] Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

[38] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. Featured Certification.

[39] Shentong Mo and Shengbang Tong. Connecting joint-embedding predictive architecture with contrastive self-supervised learning. *Advances in neural information processing systems*, 37: 2348–2377, 2024.

[40] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[41] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

[42] Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 ieee international conference on robotics and automation (icra)*, pages 4209–4215. IEEE, 2021.

[43] Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International conference on machine learning*, pages 4956–4975. PMLR, 2022.

[44] Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[45] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pages 7444–7453. PMLR, 2019.

[46] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pages 8583–8592. PMLR, 2020.

[47] Qi Wang, Zhipeng Zhang, Baao Xie, Xin Jin, Yunbo Wang, Shiyu Wang, Liaomo Zheng, Xiaokang Yang, and Wenjun Zeng. Disentangled world models: Learning to transfer semantic knowledge from distracting videos for reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2599–2608, 2025.

[48] Boxuan Zhang, Runqing Wang, Wei Xiao, Weipu Zhang, Jian Sun, Gao Huang, Jie Chen, and Gang Wang. Dymodreamer: World modeling with dynamic modulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[49] Frank Röder, Jan Benad, Manfred Eppe, and Pradeep Kr. Banerjee. Dynamics-aligned latent imagination in contextual world models for zero-shot generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[50] Lingyi Wang, Rashed Shelim, Walid Saad, and Naren Ramakrishnan. DMWM: Dual-mind world model with long-term imagination. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[51] Fabian J. Roth, Dominik K. Klein, Maximilian Kannapinn, Jan Peters, and Oliver Weeger. Stable port-hamiltonian neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[52] Yu Shang, Xin Zhang, Yinzhou Tang, Lei Jin, Chen Gao, Wei Wu, and Yong Li. Roboscape: Physics-informed embodied world model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[53] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[54] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47016–47031, 2023.

[55] Qi Lv, Xiang Deng, Gongwei Chen, Michael Y Wang, and Liqiang Nie. Decision mamba: A multi-grained state space model with self-evolution regularization for offline rl. *Advances in neural information processing systems*, 37:22827–22849, 2024.

[56] Elie Aljalbout, Maria Krinner, Angel Romero, and Davide Scaramuzza. Accelerating model-based reinforcement learning with state-space world models. In *ICLR 2025 Workshop on World Models: Understanding, Modelling and Scaling*, 2025.

[57] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 21–27 Jul 2024.

A Supporting Derivations for Soft-Hamiltonian Dynamics

This appendix makes explicit two things that follow directly from the update rule: the exact relationship between the one-step rule and the planner objective it is unrolled into, and the first-order decomposition of the learned energy along that step.

A.1 The Planner Objective is the Unrolled One-Step Rule

Eq. (2) leaves the transition F_ϕ abstract. We make the substitution explicit for the core update in Eq. (3), on steps where Hamiltonian-gradient clipping and latent norm projection are inactive. With either safeguard active, F_ϕ includes the corresponding operation described in Appendix B.7; training and planning still use the same implemented transition.
Write $\hat{\mathbf{z}} = [\mathbf{q}, \mathbf{p}, \mathbf{c}]$ and define the one-step increment of Eq. (3) as

$$\Delta_\phi(\hat{\mathbf{z}}, \mathbf{a}, \mathbf{h}) \;=\; \begin{bmatrix} (1-\alpha)\,\Delta\mathbf{q}^{\mathrm{net}} + \alpha\,\partial_{\mathbf{p}} H_\phi, \\ (1-\alpha)\,\Delta\mathbf{p}^{\mathrm{net}} - \alpha\,\partial_{\mathbf{q}} H_\phi + \mathbf{G}_\phi \mathbf{a}, \\ f_c \end{bmatrix}, \tag{8}$$

with $H_\phi = H_\phi(\mathbf{q}, \mathbf{p})$ and $(\Delta\mathbf{q}^{\mathrm{net}}, \Delta\mathbf{p}^{\mathrm{net}}) = f_{\mathrm{core}}$, $\mathbf{G}_\phi$, and f_c all evaluated at $(\hat{\mathbf{z}}, \mathbf{a}, \mathbf{h})$. The learned transition is then simply

$$F_\phi = \mathrm{id} + \Delta_\phi, \tag{9}$$

and Eq. (2) becomes the single constrained problem

$$\begin{aligned} \mathbf{a}^\star_{t:t+H-1} \;&\in\; \arg\max_{\mathbf{a}_{t:t+H-1}} \sum_{k=0}^{H-1} \gamma^k R_\phi(\hat{\mathbf{z}}_{t+k}, \mathbf{a}_{t+k}) + \gamma^H V_\phi(\hat{\mathbf{z}}_{t+H}), \\ \text{s.t.} \qquad \hat{\mathbf{z}}_{t+k+1} \;&=\; \hat{\mathbf{z}}_{t+k} + \Delta_\phi(\hat{\mathbf{z}}_{t+k}, \mathbf{a}_{t+k}, \mathbf{h}_{t+k}), \\ \mathbf{h}_{t+k} \;&=\; \mathrm{Memory}_\phi(\hat{\mathbf{z}}_{t+k}, \mathbf{a}_{t+k}, \mathbf{h}_{t+k-1}), \\ \hat{\mathbf{z}}_t \;&=\; E_\phi(o_t). \end{aligned} \tag{10}$$

Unrolling the constraint gives

$$\hat{\mathbf{z}}_{t+k} = (\mathrm{id} + \Delta_\phi)^{\circ k}(\hat{\mathbf{z}}_t), \tag{11}$$

so the reward and value terms in the planner objective are evaluated at states produced by k successive applications of Eq. (3). This is the link between the two equations: Eq. (3) is the step rule, and Eq. (2) is its H-fold unroll under the same mixing coefficient α, the same energy field H_ϕ, the same control map G_ϕ, and the same memory. No separate planner-side transition exists.
Two consequences are worth stating. First, every structural property imposed on Δ_ϕ is applied H times per candidate and once more per CEM iteration, which is why a bias that is weak in one

step is not negligible in the objective. Second, because $\mathbf{h}$ is advanced by the same recursion inside the rollout, the memory is part of the optimized map rather than a fixed context computed before planning.

A.2 Boundaries on the Latent-Space Reading of $(\mathbf{q}, \mathbf{p})$ and H_ϕ

As in Section 3, $\mathbf{q}$ and $\mathbf{p}$ are learned canonical-like coordinate *blocks*, H_ϕ is an energy-like scalar field, and we borrow only the *directions* of the Hamiltonian flow. The field $\xi_H = (\partial_{\mathbf{p}} H_\phi, -\partial_{\mathbf{q}} H_\phi)$ is orthogonal to ∇H_ϕ, so moving along it leaves the learned energy unchanged to first order — which is what repeated unrolling needs — but strict conservation is neither required nor desirable here: the learned residual fits dissipative dynamics, $\mathbf{G}_t \mathbf{a}_t$ is an explicit control path, and $\mathbf{c}$ carries non-Hamiltonian information. This is also why the model extrapolates under rescaled dynamics: an unstructured latent learns what a state becomes under *these* parameters, so rescaling mass or damping forces a refit, whereas the $(\mathbf{q}, \mathbf{p})$ branch encodes relations that stay qualitatively valid after rescaling, leaving only their strength to be recalibrated online from the recent observation–action response.

A.3 First-Order Energy Decomposition

Assume H_ϕ is C^3 in a neighborhood of the step and that neither Hamiltonian-gradient clipping nor latent norm projection is active. Let $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{p}_t)$ denote the canonical pair and let

$$\xi_\phi(\mathbf{s}) = \big(\partial_{\mathbf{p}} H_\phi(\mathbf{s}),\, -\partial_{\mathbf{q}} H_\phi(\mathbf{s})\big), \qquad \mathbf{u}_\phi(\mathbf{z}, \mathbf{a}, \mathbf{h}) = \big(0,\ \mathbf{G}_\phi(\mathbf{z}, \mathbf{a}, \mathbf{h})\mathbf{a}\big)$$

be the Hamiltonian field and the explicit control drive; $\mathbf{u}_\phi$ vanishes for zero action because it is multiplied by $\mathbf{a}$. Writing the network branch as $\Delta \mathbf{s}_t^{\text{net}} = \xi_\phi(\mathbf{s}_t) + \boldsymbol{\delta}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$, where $\boldsymbol{\delta}_\phi$ is the alignment residual left over after $\mathcal{L}_{\text{ham}}$, the canonical part of Eq. (3) reads

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \xi_\phi(\mathbf{s}_t) + \mathbf{u}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t) + (1-\alpha)\boldsymbol{\delta}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t). \tag{12}$$

Two properties of the raw field. Let $J = \left(\begin{smallmatrix} 0 & I \\ -I & 0 \end{smallmatrix}\right)$, so that $\xi_\phi(\mathbf{s}) = J \nabla H_\phi(\mathbf{s})$. Since $J^\top = -J$ and J is traceless in the relevant sense,

$$\nabla H_\phi(\mathbf{s})^\top \xi_\phi(\mathbf{s}) = \nabla H_\phi(\mathbf{s})^\top J \nabla H_\phi(\mathbf{s}) = 0, \qquad \nabla_{\mathbf{s}} \cdot \xi_\phi(\mathbf{s}) = \operatorname{tr}\big(J \nabla^2 H_\phi(\mathbf{s})\big) = 0. \tag{13}$$

These are the two properties quoted in Section 4.1: the field is orthogonal to the energy gradient, and it is divergence-free. In continuous time the first would conserve H_ϕ exactly and the second would preserve phase-space volume. Neither survives discretization exactly, which is the point of the next paragraph.

One discrete step. Taylor expansion of the learned energy along one step gives

$$\Delta \mathcal{H}_t = \nabla H_\phi(\mathbf{s}_t)^\top \Delta \mathbf{s}_t + \tfrac{1}{2} \Delta \mathbf{s}_t^\top \nabla^2 H_\phi(\mathbf{s}_t) \Delta \mathbf{s}_t + R_3(\Delta \mathbf{s}_t). \tag{14}$$

Substituting Eq. (12) and applying the first identity of Eq. (13) to cancel the ξ_ϕ term yields

$$\Delta \mathcal{H}_t = \underbrace{\nabla H_\phi(\mathbf{s}_t)^\top \mathbf{u}_\phi}_{\text{action work}} + \underbrace{(1-\alpha)\, \nabla H_\phi(\mathbf{s}_t)^\top \boldsymbol{\delta}_\phi}_{\text{alignment residual}} + \underbrace{\tfrac{1}{2} \Delta \mathbf{s}_t^\top \nabla^2 H_\phi(\mathbf{s}_t) \Delta \mathbf{s}_t + R_3(\Delta \mathbf{s}_t)}_{\text{curvature / discretization}}. \tag{15}$$

Under the stated conditions, Eq. (15) is exact to first order. To first order the learned energy can change through exactly two channels — the explicit control drive and the alignment residual — while the Hamiltonian component contributes only through curvature and higher-order discretization terms. This motivates the diagnostics in Section 6:

- In the action-free regime, $\mathbf{u}_\phi = 0$ and the surviving terms are scaled by $(1-\alpha)$ or are curvature-order, so the passive energy trace should stay nearly flat — tested in Figure 3a, where drift is about 1% on Cheetah Run.
- When actions are present, the action-work term is $\nabla H_\phi^\top \mathbf{u}_\phi$, i.e. the projection of the control drive onto the energy gradient. Energy change should therefore track $\text{push}_t = (\nabla_{\mathbf{p}} \mathcal{H}_t) \cdot \text{control}_t$ — tested in Figure 5, where $\text{corr}(\text{sign}\, \Delta\mathcal{H}, \text{push}) = 0.757$ and $\text{corr}(|\Delta\mathcal{H}|, |\text{push}|) = 0.745$.

Effect of clipping and projection. The implementation clips $\partial_{\mathbf{q}} H_\phi$ and $\partial_{\mathbf{p}} H_\phi$ separately and projects the next latent onto a norm ball (Appendix B.7). These operations need not preserve the raw field's energy orthogonality or zero divergence. When either is active, Eqs. (12) and (15) require an additional clipping/projection correction, which the Hamiltonian cancellation does not remove. The energy traces in Section 6.3 are measured on the implemented update, safeguards included. Appendix B.7 describes the discretization and its relation to symplectic integration.

Table 4: Overview of the experimental protocol. The upper block lists the environment and evaluation settings shared by all methods; the lower block lists key training-hyperparameter differences across methods.

Shared environment and evaluation protocol	
Tasks	*Core:* Reacher Easy / Finger Spin / Cheetah Run / Cartpole Swingup *Additional:* Walker Run / Hopper Hop
Environment	DeepMind Control Suite, flattened state observations
Action repeat	Finger Spin = 2; others = 4
Episode length	Finger / Cheetah = 500; Reacher / Cartpole = 200
Total interaction	100k env steps (3 seeds: 7/8/9)
Evaluation frequency	Every 5k steps, 3 episodes per evaluation
Reported metrics	Final return, AUC (mean $\pm$ std over seeds)

	HaM-World	TD-MPC2	DreamerV3
Latent dim	48 (8/8/32)	128	256 (det.)+16×16
Batch size	128	128	64
Seq. length	8	8	16
Grad. steps	2	1	1
Seed steps	5k	5k	5k
Train every	2	2	2
Planning horizon	6 (CEM)	5 (MPPI)	8 (imagination)
Learning rate	10^{-4}	3×10^{-4}	10^{-4}

PPO (on-policy actor-critic)		*SAC (off-policy actor-critic)*	
Actor / critic MLP	$[128, 128]$	Actor / critic MLP	$[128, 128]$
Rollout steps	1024	Replay capacity	300k
Minibatch size	128	Batch size	128
Update epochs	4	Seed steps	8k
Learning rate	2×10^{-4}	Train every	2 (grad steps $= 1$)
Discount γ	0.99	Learning rate (all)	3×10^{-4}
GAE λ	0.95	Target τ	0.01
Clip ratio	0.2	Init. temperature	0.1 (learnable)
Value / entropy coef	0.5 / 0.001	Target entropy	$-\dim(\mathcal{A})$
Grad clip / target KL	0.5 / 0.05	Actor / target update	every 1 step

B Implementation Details

B.1 Shared Protocol and Method Hyperparameters

DreamerV3 is implemented in the repository as a state-based DreamerV3 reimplementation. It retains the core mechanisms of discrete RSSM, symlog encoder/decoder, symlog+twohot reward/value regression, KL balancing, λ-returns, and percentile return scaling; it uses deterministic state dimension 256, discrete stochastic state 16×16, and imagination horizon 8. TD-MPC2 is a single-task online state-based TD-MPC2 implementation with latent dim 128, SimNorm groups 8, planner horizon 5, 256 candidates, and 32 elites. PPO and SAC use the configurations in the table above, from `ppo/configs/low_budget_compare_dmcontrol.yaml` and `sac/configs/low_budget_compare_dmcontrol.yaml`. They do not learn explicit latent dynamics and do not use a CEM/MPPI planner; they serve as model-free control references under the same 100k-step budget.

B.2 Network Architecture

The current implementation uses state-based inputs and an MLP encoder. By default, the encoder is a two-layer MLP of width 256 that maps flattened observations to a 48-dimensional latent. The trainer then splits it into 8-dimensional $\mathbf{q}$, 8-dimensional $\mathbf{p}$, and 32-dimensional $\mathbf{c}$. The projector output dimension is 64. The reward head, value head, policy prior, Hamiltonian-pair dynamics core, context updater, and Hamiltonian head use MLPs of width 128 or 256. The memory mechanism stacks two Mamba-style selective state-space layers with model/state dimension 128. It outputs the history-conditioned features used by the shared latent dynamics.

Table 5: Default network architecture and planner configuration for HaM-World.

Element	Default configuration
Encoder	MLP, hidden dims $[256, 256]$, latent dim 48
q/p/c split	$8/8/32$
Projector	hidden dim 128, projection dim 64
Memory	2-layer Mamba-style selective SSM, model/state dim 128
Hamiltonian / Aux / Energy heads	hidden dims $[128, 128]$
Planner	horizon 6, iterations 6, candidates 128, elite 16

B.3 Training Losses: Definitions and Rationale

Eq. (4) groups the implementation losses into three design roles. Prediction and value losses make the latent state useful for control, rollout losses train the same finite-horizon transitions used by the planner, and geometric losses softly bias the canonical coordinates toward Soft-Hamiltonian behavior without imposing hard conservation. Together they keep the planner-facing latent dynamics predictive, reward-aware, and structurally stable.

Representation alignment.

$$\mathcal{L}_{\text{repr}} = \tfrac{1}{T} \sum_t \|g_\phi(\hat{\mathbf{z}}_{t+1}) - \text{sg}(\bar{g}_{\bar{\phi}}(\bar{\mathbf{z}}_{t+1}))\|_2^2.$$

Here $g_\phi / \bar{g}_{\bar{\phi}}$ are the online/EMA-target projectors and the target path is stop-gradient. This JEPA-style objective aligns predicted latents with slowly moving target latents without reconstructing observations. The reason for using it is practical: pixel-level or raw-state reconstruction can force the latent to preserve nuisance variation, while the planner only needs a compact predictive state. EMA targets also stabilize bootstrapping by preventing the transition model and its target from moving simultaneously.

One-step latent dynamics.

$$\mathcal{L}_{\text{dyn}} = \tfrac{1}{T} \sum_t \|\hat{\mathbf{z}}_{t+1} - \text{sg}(\mathbf{z}_{t+1})\|_2^2.$$

This term anchors the learned transition to the online encoder at the next step. It is the local consistency counterpart of $\mathcal{L}_{\text{roll}}$: without it, multi-step rollout loss would have to correct both immediate transition mismatch and long-horizon accumulation; without rollout loss, one-step consistency alone could still accumulate errors under planning.

Reward and value heads.

$$\mathcal{L}_{\text{reward}} = \tfrac{1}{T} \sum_t \text{CE}_{\text{2hot}}\left(\hat{r}_\phi^{\text{logits}}(\hat{\mathbf{z}}_{t+1}), r_t\right).$$

The reward head is trained on the predicted next latent because the planner evaluates candidate actions through imagined next states. The value head supplies the terminal estimate for horizon-limited CEM. Following DreamerV3, scalar rewards and values are represented with symlog two-hot bins, and the value head is stabilized with an EMA slow target:

$$\mathcal{L}_{\text{value}}^{\text{ce}} = -\tfrac{1}{T} \sum_t \mathbf{p}_t^{\lambda\top} \log \text{softmax}\big(\hat{V}_\phi^{\text{logits}}(\mathbf{z}_t)\big), \tag{16}$$

$$\mathcal{L}_{\text{value}}^{\text{slow}} = -\tfrac{1}{T} \sum_t \text{sg}[\bar{\mathbf{p}}_t]^\top \log \text{softmax}\big(\hat{V}_\phi^{\text{logits}}(\mathbf{z}_t)\big), \tag{17}$$

where $\mathbf{p}_t^\lambda$ is the λ-return target encoded over two-hot bins and $\bar{\mathbf{p}}_t$ is the EMA value-target distribution at $\mathbf{z}_t$. In the implementation, $\mathcal{L}_{\text{value}} \equiv \mathcal{L}_{\text{value}}^{\text{ce}} + \beta_{\text{slow}} \mathcal{L}_{\text{value}}^{\text{slow}}$ with $\beta_{\text{slow}}{=}1$. The two-hot target improves robustness to return scale, and the slow target reduces value-target oscillation during online training.

Policy prior for CEM warm start.

$$\mathcal{L}_{\text{policy}} = \tfrac{1}{T} \sum_t \|\pi_\phi(\mathbf{z}_t) - \mathbf{a}_t\|_2^2.$$

The auxiliary policy is not the final decision rule. It is a deterministic action prior trained by behavior cloning and used to warm-start CEM trajectories, reducing planner search variance when the action dimension or horizon is nontrivial.

Multi-step rollout consistency.

$$\mathcal{L}_{\text{roll}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} \|\hat{\mathbf{z}}_{s+k}^{(s)} - \text{sg}(\mathbf{z}_{s+k})\|_2^2, \tag{18}$$

where $\hat{\mathbf{z}}_{s+k}^{(s)}$ denotes the k-step predicted latent rolled out from $\mathbf{z}_s$ using $\mathbf{h}_s$ as the initial memory condition. This is the loss most directly matched to planning: CEM queries the model for several steps, so training only a one-step predictor would create exposure bias. The stop-gradient target keeps the encoder as the reference coordinate system, while rerolling from multiple positions s makes the transition robust to different rollout prefixes.

B.4 Structured Regularizers: Definitions and Rationale

The geometric loss

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{ham}}\mathcal{L}_{\text{ham}} + \lambda_{\text{en}}\mathcal{L}_{\text{energy}} + \lambda_{\text{sa}}\mathcal{L}_{\text{sa}} + \lambda_{\text{temp}}\mathcal{L}_{\text{temp}} + \lambda_{\text{dec}}\mathcal{L}_{\text{dec}} + \lambda_{\text{c}}\mathcal{L}_{\text{c}}$$

implements soft biases on the canonical $(\mathbf{q}, \mathbf{p})$ dynamics and the context variable $\mathbf{c}$. These terms are low-weighted: their role is to shape the learned dynamics toward the intended decomposition, while the prediction, reward, value, and rollout losses remain responsible for task fit.

Hamiltonian alignment.

$$\mathcal{L}_{\text{ham}} = \left\|\Delta\mathbf{q}_t^{\text{net}} - \partial_{\mathbf{p}}\mathcal{H}_t\right\|_2^2 + \left\|\Delta\mathbf{p}_t^{\text{net}} + \partial_{\mathbf{q}}\mathcal{H}_t\right\|_2^2.$$

This term aligns the network branch with the Hamiltonian vector field; the learned control drive is added separately in the $\mathbf{p}$ update. It does not make the system strictly Hamiltonian, but gives the learnable q/p update a directional bias so that the energy geometry remains visible even when $\alpha < 1$ and residual dynamics are active.

Action-free energy regularization.

$$\mathcal{L}_{\text{energy}} = \mathbb{E}_{t:\|\mathbf{a}_t\|<\epsilon}\left[(\mathcal{H}(\mathbf{q}_{t+1}, \mathbf{p}_{t+1}) - \mathcal{H}(\mathbf{q}_t, \mathbf{p}_t))^2\right].$$

This term is applied only near the action-free regime, where the intended behavior is small energy drift. It is not used to prevent action-induced energy transfer; random or policy actions should be able to move the system across energy levels. This distinction is important because control affects how the q/p dynamics evolves, while $\mathbf{c}$ supplies contextual information about semantic and non-conservative factors.

Small-action smoothness.

$$\mathcal{L}_{\text{sa}} = \mathbb{E}_{t:\|\mathbf{a}_t\|<\epsilon}\left[\|\Delta\mathbf{q}_t\|^2 + \|\Delta\mathbf{p}_t\|^2\right].$$

This regularizer discourages spurious q/p motion when the action magnitude is negligible. It complements $\mathcal{L}_{\text{energy}}$: energy regularization controls changes in H, while small-action smoothness controls the latent displacement itself.

Temporal and statistical role separation.

$$\mathcal{L}_{\text{temp}} = \|\Delta\mathbf{q}_t\|^2 - \rho_{\text{temp}}\|\Delta\mathbf{p}_t\|^2, \qquad \mathcal{L}_{\text{dec}} = \left\|\tfrac{1}{B-1}(\mathbf{Q}-\bar{\mathbf{q}}\mathbf{1}^\top)^\top(\mathbf{P}-\bar{\mathbf{p}}\mathbf{1}^\top)\right\|_F^2. \tag{19}$$

$\mathcal{L}_{\text{temp}}$ uses ρ_{temp}=0.5 in the released configs and acts as a lightweight q-slow/p-fast bias: $\mathbf{q}$ should behave more like a slowly varying configuration coordinate, while $\mathbf{p}$ should carry faster momentum-like changes. It is not optimized in isolation; prediction and rollout losses prevent the signed bias from dominating the transition. $\mathcal{L}_{\text{dec}}$ suppresses batch-level correlation between $\mathbf{q}$ and $\mathbf{p}$, reducing the risk that the canonical coordinates collapse into duplicated copies.

Context sparsity.

$$\mathcal{L}_{\text{c}} = \mathbb{E}\left[|\Delta\mathbf{c}_t|\right].$$

The context variable is meant to provide complementary semantic and non-conservative information, not to absorb all fast dynamics. Sparsity in $\Delta\mathbf{c}_t$ encourages $\mathbf{c}$ to change only when useful for prediction or control, leaving the canonical coordinates to carry the structured q/p evolution.

B.5 Complete 14-Term Loss and Weight Table

The training logger records the optimized loss fields plus value subdiagnostics. Table 6 maps each field to its paper symbol, conceptual group, default weight, and warmup schedule.

Table 6: Optimized loss fields and value subdiagnostics, with conceptual groups, default weights, and warmup schedule.

Implementation field (logger)	Paper symbol	Conceptual group	Default weight	Warmup
`repr_loss`	$\mathcal{L}_{\text{repr}}$	repr	1.0	–
`dyn_loss`	$\mathcal{L}_{\text{dyn}}$	dyn	1.0	–
`roll_loss`	$\mathcal{L}_{\text{roll}}$	roll	0.5	30%–60%
`reward_loss`	$\mathcal{L}_{\text{reward}}$	rew	1.0	–
`value_loss`	$\mathcal{L}_{\text{value}}$	val	0.5	–
`value_ce_loss`	$\mathcal{L}^{\text{ce}}_{\text{value}}$	val diagnostic	inside val	–
`value_slow_loss`	$\mathcal{L}^{\text{slow}}_{\text{value}}$	val diagnostic	inside val	–
`policy_prior_loss`	$\mathcal{L}_{\text{policy}}$	pol (β_{p})	0.1	–
`sa_loss`	$\mathcal{L}_{\text{sa}}$	geo	0.05	30%–60%
`energy_loss`	$\mathcal{L}_{\text{energy}}$	geo	0.01	30%–60%
`hamiltonian_loss`	$\mathcal{L}_{\text{ham}}$	geo	0.05	–
`temp_loss`	$\mathcal{L}_{\text{temp}}$	geo	0.01	–
`decouple_loss`	$\mathcal{L}_{\text{dec}}$	geo	0.01	–
`c_sparse_loss`	$\mathcal{L}_{\text{c}}$	geo	0.001	–

The conceptual groups in Eq. (4) relate to the table as follows: $\mathcal{L}_{\text{value}}$ in the main text equals $\mathcal{L}^{\text{ce}}_{\text{value}} + \beta_{\text{slow}}\mathcal{L}^{\text{slow}}_{\text{value}}$, with default β_{slow}=1; $\mathcal{L}_{\text{geo}}$ in the main text is the weighted sum of `sa+energy+hamiltonian+temp+decouple+c_sparse`. Rollout, small-action, and energy terms are warmed up because they depend on an already meaningful latent coordinate system; applying them too strongly at initialization can over-constrain the model before reward and one-step dynamics have stabilized. The Hamiltonian, temporal, decoupling, and context-sparsity weights are kept small from the start, serving as weak shaping biases rather than dominant objectives.

A single global weight configuration is used unchanged across all tasks, seeds, and OOD conditions; no per-task sweep was performed. $\mathcal{L}_{\text{repr}}, \mathcal{L}_{\text{dyn}}, \mathcal{L}_{\text{reward}}$ are primary at weight 1; $\mathcal{L}_{\text{roll}}, \mathcal{L}_{\text{value}}$ sit at 0.5; $\mathcal{L}_{\text{policy}}$ at 0.1 assists the CEM proposal and is not the decision rule; and every geometric term lies in 10^{-3}–5×10^{-2}, two to three orders of magnitude below the task-fitting terms, so they shape the dynamics without competing with it.

B.6 Training Pseudocode

B.7 Soft-Hamiltonian Pair Discretization and Train/Test Consistency

Eq. (3) gives the discrete update form for the Hamiltonian pair in HaM-World. We describe how it is used consistently during training and inference, and how it corresponds to the continuous-time controlled Hamiltonian form $\dot{\mathbf{q}} = \partial\mathcal{H}/\partial\mathbf{p}$, $\dot{\mathbf{p}} = -\partial\mathcal{H}/\partial\mathbf{q} + g(\mathbf{q})\mathbf{a}$.

Discretization scheme. The implementation uses the first-order explicit update in Eq. (3) as the discrete transition:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \Delta\mathbf{q}_t, \qquad \mathbf{p}_{t+1} = \mathbf{p}_t + \Delta\mathbf{p}_t, \tag{20}$$

where both Hamiltonian gradients are evaluated at $(\mathbf{q}_t, \mathbf{p}_t)$. Eq. (3) gives the core update before safeguards. The implementation clips each Hamiltonian-gradient block to norm 5 before mixing, then projects the concatenated next latent $[\mathbf{q}_{t+1}, \mathbf{p}_{t+1}, \mathbf{c}_{t+1}]$ onto the norm-50 ball. With these safeguards inactive, $\alpha \to 1$ gives a forward-Euler step along the Hamiltonian field plus the control drive, whereas $\alpha \to 0$ gives a network update plus control. Intermediate α balances task fitting with the Hamiltonian bias. This is a *local geometric bias*, not a symplectic Euler or leapfrog solver.

Training updates, evaluation rollouts, and CEM imagination use the same explicit core and clipping/projection safeguards, with both gradients taken at the current $(\mathbf{q}_t, \mathbf{p}_t)$ and a fixed per-checkpoint α.

Discretization choice. Standard explicit symplectic Euler splitting and Velocity Verlet exploit a *separable* Hamiltonian $H(\mathbf{q}, \mathbf{p}) = T(\mathbf{p}) + V(\mathbf{q})$. Our learned scalar field over concatenated $(\mathbf{q}, \mathbf{p})$ has no such structure, so this explicit splitting is unavailable. General symplectic Euler remains applicable to a nonseparable Hamiltonian, but typically requires an implicit solve rather than an explicit staggered update.

Algorithm 1 HaM-World Data Collection and CEM Planning

Require: environment $\mathcal{E}$, replay buffer capacity N, CEM horizon H, iterations I, candidate count N_c, elite count N_e, EMA coefficient τ

Initialize $\mathcal{D} \leftarrow \emptyset$; initialize all network parameters ϕ and set $\bar{\phi} \leftarrow \phi$; set $\mathbf{h}_0 \leftarrow \mathbf{0}$
for each environment step t **do**
  $\mathbf{z}_t = E_\phi(o_t) = [\mathbf{q}_t, \mathbf{p}_t, \mathbf{c}_t]$ ▷ online encoder
  $\mathbf{h}_t = \mathrm{Memory}_\phi(\mathbf{z}_t,\ \mathbf{a}_{t-1},\ \mathbf{h}_{t-1})$ ▷ Mamba memory update
  Initialize action distribution $(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$
  for $i = 1, \ldots, I$ **do** ▷ CEM iteration
    Sample $\{\mathbf{a}_{t:t+H-1}^{(j)}\}_{j=1}^{N_c} \sim \mathcal{N}(\boldsymbol{\mu}_{i-1}, \boldsymbol{\sigma}_{i-1}^2)$
    for each candidate j **do**
      Roll out imagination to obtain $\hat{\mathbf{z}}_{t+1:t+H}^{(j)}$ using the transition in Algorithm 2
      $G^{(j)} = \sum_{k=0}^{H-1} \gamma^k \hat{r}(\hat{\mathbf{z}}_{t+k}^{(j)}) + \gamma^H \bar{V}(\hat{\mathbf{z}}_{t+H}^{(j)})$
    end for
    Select top-N_e elites; compute $w^{(j)} = \mathrm{softmax}(G^{(j)}/\tau)$
    $\boldsymbol{\mu}_i = \sum_j w^{(j)} \mathbf{a}^{(j)}$, $\boldsymbol{\sigma}_i^2 = \sum_j w^{(j)} (\mathbf{a}^{(j)} - \boldsymbol{\mu}_i)^2 + \epsilon$
  end for
  Execute $\mathbf{a}_t \leftarrow \boldsymbol{\mu}_I[0]$; add $\{(o_t, \mathbf{a}_t, r_t, o_{t+1})\}$ to $\mathcal{D}$
  if $|\mathcal{D}| \geq$ update threshold **then**
    Run Algorithm 2 (model update)
  end if
end for

Algorithm 2 HaM-World Model Update Step (called whenever an update is triggered)

Require: batch $\{(o_{1:T}, \mathbf{a}_{1:T}, r_{1:T})\}$, sequence length T, scheduled mixing coefficient $\alpha(s)$, rollout length K

$\mathbf{z}_{1:T} = E_\phi(o_{1:T})$; $\bar{\mathbf{z}}_{1:T} = \mathrm{sg}(\bar{E}_{\bar{\phi}}(o_{1:T}))$ ▷ online / target latent
$\mathbf{h}_0 \leftarrow \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  $\mathbf{h}_t = \mathrm{Memory}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_{t-1})$ ▷ Mamba memory
  $(\Delta\mathbf{q}_t^{\mathrm{net}},\ \Delta\mathbf{p}_t^{\mathrm{net}}) = f_{\mathrm{core}}(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$ ▷ Hamiltonian-pair core
  $\mathcal{H}_t = H_\phi(\mathbf{q}_t, \mathbf{p}_t)$; $\mathbf{G}_t = G_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$; obtain $\nabla_{\mathbf{q}}\mathcal{H}_t,\ \nabla_{\mathbf{p}}\mathcal{H}_t$ ▷ Hamiltonian head
  $\hat{\mathbf{q}}_{t+1} = \mathbf{q}_t + (1-\alpha)\Delta\mathbf{q}_t^{\mathrm{net}} + \alpha\, \nabla_{\mathbf{p}}\mathcal{H}_t$
  $\hat{\mathbf{p}}_{t+1} = \mathbf{p}_t + (1-\alpha)\Delta\mathbf{p}_t^{\mathrm{net}} - \alpha \nabla_{\mathbf{q}}\mathcal{H}_t + \mathbf{G}_t\mathbf{a}_t$ ▷ Hamiltonian-pair update
  $\hat{\mathbf{c}}_{t+1} = \mathbf{c}_t + f_c(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$ ▷ semantic / context update
  $\hat{\mathbf{z}}_{t+1} = [\hat{\mathbf{q}}_{t+1},\ \hat{\mathbf{p}}_{t+1},\ \hat{\mathbf{c}}_{t+1}]$
end for
// Base losses
$\mathcal{L}_{\mathrm{repr}} = \frac{1}{T}\sum_t \|g_\phi(\hat{\mathbf{z}}_{t+1}) - \mathrm{sg}(\bar{g}_{\bar{\phi}}(\bar{\mathbf{z}}_{t+1}))\|_2^2$
$\mathcal{L}_{\mathrm{dyn}} = \frac{1}{T}\sum_t \|\hat{\mathbf{z}}_{t+1} - \mathrm{sg}(\mathbf{z}_{t+1})\|_2^2$ ▷ target is the online encoder, stop-gradient
$\mathcal{L}_{\mathrm{rew}} = \mathrm{CE}_{\mathrm{2hot}}(\hat{r}_\phi^{\mathrm{logits}}(\hat{\mathbf{z}}_{t+1}), r_t)$ ▷ reward acts on predicted next latent
Compute λ-return targets with $\bar{V}_{\bar{\phi}}$; set $\mathcal{L}_{\mathrm{val}} = \mathcal{L}_{\mathrm{val}}^{\mathrm{ce}} + \mathcal{L}_{\mathrm{val}}^{\mathrm{slow}}$ ▷ value acts on current latent
From each position s, reroll for K steps using the snapshot memory state $\mathbf{h}_s$:
$\mathcal{L}_{\mathrm{roll}} = \frac{1}{|\mathcal{S}|}\sum_s \sum_{k=1}^{K} \|\hat{\mathbf{z}}_{s+k}^{(s)} - \mathrm{sg}(\mathbf{z}_{s+k})\|_2^2$
// Value subterms and policy prior
Compute $\mathcal{L}_{\mathrm{val}}^{\mathrm{ce}}, \mathcal{L}_{\mathrm{val}}^{\mathrm{slow}}$ using Eqs. (16)–(17); compute $\mathcal{L}_{\mathrm{policy}} = \|\pi_\phi(\mathbf{z}_t) - \mathbf{a}_t\|^2$
// Structured regularizers (enabled according to the warmup schedule)
Compute $\mathcal{L}_{\mathrm{sa}}, \mathcal{L}_{\mathrm{energy}}, \mathcal{L}_{\mathrm{ham}}, \mathcal{L}_{\mathrm{dec}}, \mathcal{L}_{\mathrm{temp}}, \mathcal{L}_{\mathrm{c}}$
$\mathcal{L} = \sum_{k\in\mathcal{K}} w_k(s)\, \mathcal{L}_k$ ▷ $\mathcal{K}$ = implemented weighted optimization terms in Table 6
Backpropagate, clip gradients, and update ϕ
$\bar{\phi} \leftarrow \tau\bar{\phi} + (1-\tau)\phi$ ▷ EMA update for encoder, projector, and value head

Symplecticity is a property of the *entire* one-step map. Alongside the Hamiltonian field, our implemented transition contains the learned residual, the control port $\mathbf{G}_\phi\mathbf{a}$, memory conditioning through $\mathbf{h}_t$, the context channel f_c, Hamiltonian-gradient clipping, and latent norm projection. Using a symplectic method for one branch alone would not guarantee symplecticity of the composite update. The construction is therefore a local geometric bias. A separable or explicitly mechanical parameterization would permit explicit symplectic splitting of the Hamiltonian branch; whether that improves planning remains open.

Scheduled mixing coefficient. In the released implementation, α starts from 0.1 and, in the paper configs, increases linearly after 30% of training until it reaches at most 0.5. This schedule is applied during both training updates and planner rollouts through the same model step. Separately, rollout, small-action, and energy loss weights are warmed up from 30% to 60% of training.

Invariance of the Hamiltonian head. $\mathcal{H}_\phi$ is parameterized as a two-layer MLP with hidden dimension 128 and SiLU activation. It takes $(\mathbf{q}, \mathbf{p})$ as input and outputs a scalar, with no equivariance constraints; the physics prior enters through the action-conditioned Hamiltonian form motivated by controlled Hamiltonian models [4, 9], and $\mathcal{L}_{\text{ham}}$ aligns the q/p residual update with $\partial\mathcal{H}/\partial(\mathbf{q}, \mathbf{p})$.

B.8 Choosing the q/p/c Split Dimensions

The default split is $\dim(\mathbf{q})=\dim(\mathbf{p})=8$ and $\dim(\mathbf{c})=32$, for a total latent dimension of 48. This choice is based on three considerations. First, DMC state dimensions are at most 24, so a $8+8=16$-dimensional canonical subspace is sufficient to cover low-dimensional configuration/velocity symplectic pairs. Second, the 32-dimensional $\mathbf{c}$ is the largest coordinate group, avoiding excessive restriction of semantic and non-conservative context. Third, the total dimension is much smaller than TD-MPC2's 128-dimensional latent, reflecting the parameter efficiency of the q/p/c split. The necessity of this split is validated in Section 6 and in ablation A1, which removes q/p/c.

A1 isolation argument. A1 preserves the planner, recurrent memory, training horizon, latent size, reward/value objectives, and optimization budget, but replaces the role-separated transition with an unstructured latent transition and removes the Soft-Hamiltonian q/p bias. Under the Full model, $\mathcal{L}_{\text{ham}}$ aligns the residual update of $(\mathbf{q}, \mathbf{p})$ with a scalar energy field $\mathcal{H}_\phi(\mathbf{q}, \mathbf{p})$, while $\mathbf{c}$ carries actuation, contact, and dissipative information; under A1 there is no distinguished pair on which the Hamiltonian vector field can act, and the same capacity represents conservative, controlled, and dissipative factors in one mixed coordinate system. If the improvements were only due to recurrence, model size, or short-horizon reward fitting, A1 should remain close to the Full model once memory is retained. It does not: A1 has lower return and OOD scores, while short-horizon control is still learned.

B.9 Mamba Selective Memory

The memory mechanism Memory_ϕ stacks two lightweight Mamba-style selective SSM layers [8, 57], with model dimension 128 and recurrent-state dimension 128. The concatenated input $(\mathbf{z}_t, \mathbf{a}_t)$ is projected to the model dimension, and the output $\mathbf{h}_t$ enters f_{core}, G_ϕ, and f_c simultaneously within the unified latent dynamics.

Exact layer specification. Each of the two layers consists of: LayerNorm on the layer input; input-dependent projections producing the step size, state-update, and readout parameters from the current input; a learned diagonal decay for the recurrent state; a sigmoid-gated readout; and a residual feed-forward block with $2\times$ hidden expansion. This lightweight Mamba-style variant uses a 128-dimensional recurrent state without temporal convolution or additional state expansion. Input-dependent step sizes and state/readout projections operate within a diagonal recurrence.
Memory does not participate in encoding the latent state $\mathbf{z}_t$, which is produced directly by the encoder. It only provides historical conditioning for the latent dynamics. This design ensures that the reward/value heads and planner always consume $\mathbf{z}_t$, while memory supplies the history summary $\mathbf{h}_t$ that the transition needs under partial observability.

B.10 Alternative Memory Mixers

Section 7 argues that Mamba is chosen for input-dependent selectivity, controlled long-range decay, and a fixed-size state compatible with batched rollouts. A Transformer sequence mixer has greater raw capacity than a selective SSM but fits the state-transition interface worse. Our memory conditions the transition at *every* step of *every* imagined rollout, which imposes three requirements: a **fixed-size runtime state**, because CEM unrolls N_c candidates $\times$ H steps per real step and attending over a

token history would recompute attention at each imagined step at a cost scaling with candidates $\times$ horizon; $O(1)$ **incremental updates**, since each real step appends one observation whereas a Transformer must reprocess or cache-manage the whole context; and an output that is **injectable as a single vector** into f_{core}, G_ϕ, and f_c, which is the native form of a recurrent state. A controlled experiment replacing only the memory module confirms this:

Table 7: Replacing only the memory module (final return at 100k env steps, 3 seeds). All other components, objectives, and budgets are held fixed.

Memory	Finger Spin	Reacher Easy	Cheetah Run	Cartpole Swingup
Mamba (ours)	$\mathbf{254.0}_{\pm 14.1}$	$\mathbf{150.6}_{\pm 7.5}$	$\mathbf{184.4}_{\pm 8.4}$	$\mathbf{58.9}_{\pm 3.8}$
Transformer	$91.89_{\pm 46.18}$	$144.67_{\pm 16.57}$	$60.65_{\pm 10.73}$	$51.25_{\pm 6.63}$

Mamba is higher on 4/4 tasks, with the largest gaps on Finger Spin and Cheetah Run.

B.11 CEM Planner Details

The CEM objective is $J(\mathbf{a}_{t:t+H-1}) = \sum_{k=0}^{H-1} \gamma^k \hat{r}_{t+k} + \gamma^H \hat{V}(\hat{\mathbf{z}}_{t+H})$. The implementation uses horizon H=6, iterations I=6, N_c=128 candidate action sequences per iteration, and top-N_e=16 elites. The mean/variance are updated by weighting scores $G^{(j)}$ with a softmax at temperature τ=0.5. The initial mean is warm-started by the deterministic policy prior when enabled, and each CEM iteration also includes 32 noisy policy trajectories; the Gaussian sampling scale uses init std 0.4 and min std 0.05. At evaluation the executed action is $\boldsymbol{\mu}_I[0]$; during training, exploration noise with std 0.3 is added after planning.

B.12 OOD Evaluation Conditions

Table 8: OOD evaluation conditions on Reacher Easy and Finger Spin. Each condition is evaluated zero-shot with 10 episodes per seed and 3 seeds, for 30 episodes total. Reported mean$\pm$std is computed across the three seed-level mean returns.

Task	Category	Condition	Implementation
Reacher Easy	Dynamics	ID	default parameters
		mass $\times$0.7	link mass scale 0.7
		mass $\times$1.3	link mass scale 1.3
		damp $\times$0.5	joint damping scale 0.5
		damp $\times$2.0	joint damping scale 2.0
		act $\times$0.7	actuator strength 0.7
		act $\times$1.3	actuator strength 1.3
Finger Spin	Dynamics	ID	default parameters
		fric $\times$0.5	finger-object friction 0.5
		fric $\times$1.5	finger-object friction 1.5
		mass $\times$1.3	spinner mass 1.3
		mass $\times$1.5	spinner mass 1.5
	Partial obs.	delay = 2	action-execution delay of 2 control steps
		mask 30%	randomly mask 30% of observation dimensions at every step (set to 0)

B.13 Normalized OOD Retention

Table 9 gives normalized retention for all five methods.

Table 9: Normalized OOD retention, $\text{Retention} = 100 \times \overline{R}_{\text{OOD}}/\overline{R}_{\text{ID}}$, computed from the ID final returns of Table 1 and the 6-condition OOD averages of Table 2; the macro column weights both tasks equally. As ratios of reported means these carry no recoverable std.

	Finger Spin		Reacher Easy		Macro (12 cond.)	
Method	Retention	Abs. ret.	Retention	Abs. ret.	Retention	Abs. ret.
HaM-World	72.6%	**184.3**	99.5%	**149.9**	86.0%	**167.1**
TD-MPC2	72.0%	167.3	**125.6%**	132.0	**98.8%**	149.7
SAC	68.2%	118.4	77.7%	92.4	73.0%	105.4
DreamerV3	**76.3%**	55.2	50.5%	9.3	63.4%	32.3
PPO	67.3%	47.3	68.3%	12.9	67.8%	30.1

DreamerV3 reaches 76.3% on Finger Spin against our 72.6%, from absolute returns of 55.2 versus 184.3; TD-MPC2 reaches 125.6% on Reacher Easy against our 99.5%, from 132.0 versus 149.9. HaM-World obtains the highest absolute return in all 12 conditions, with task-dependent relative retention.
The main OOD table reports raw zero-shot return. For retention diagnostics in the released scripts, $\text{Retention}(\%) = \text{Return}_{\text{OOD}}/\text{Return}_{\text{ID}} \times 100$ is computed per seed using the corresponding ID return before averaging, so differences in ID baselines across seeds do not contaminate the retention value.

B.14 Hamiltonian Validation Setup

Figure 3 uses a low-damping passive validation regime for the Hamiltonian diagnostics:

- **Kick.** After `env.reset()`, inject uniformly random angular velocities of ± 5 rad/s into all joints. For Finger Spin, the same step prevents the task from freezing completely under zero force after a zero-velocity reset on the table.
- **Zero damping.** Set all joint `damping` values in the mjModel to 0, reducing passive dissipation during free evolution.
- **Episode.** Run 10 episodes each for no-action and random-action, with 200 steps per episode.

This reduced-damping setting isolates action-free energy drift from action-induced energy variation.

B.15 Reproducibility Protocol

The current release configuration uses total steps 100k, seed steps 5k, batch size 128, sequence length 8, gradient steps 2, discount 0.99, learning rate 10^{-4}, gradient clipping 10.0, AdamW optimizer with $(\beta_1{=}0.9, \beta_2{=}0.999)$, and target-update coefficient $\tau{=}0.01$ for the EMA encoder/projector/value heads. DreamerV3 uses deterministic RSSM dimension 256 and 16×16 discrete latent, with symlog+twohot regression for reward/value heads. TD-MPC2 uses latent dim 128 with SimNorm(8), planner horizon 5, and 256 MPPI candidates. PPO / SAC are model-free actor-critic baselines.

Hardware and runtime. All training is performed on a server with $8\times$ NVIDIA GeForce RTX 3090 GPUs; each GPU has 24 GiB of memory (`nvidia-smi` reports 24576 MiB). The software stack uses NVIDIA driver 570.211.01 and CUDA 12.8. Each seed occupies one GPU. For a single-seed run of 100k environment steps, estimated training times are about 4.5 h for HaM-World, 3.5 h for TD-MPC2, and 3.0 h for DreamerV3. OOD evaluation takes less than 5 min per condition per seed. Summing these per-run estimates over all reported model-based runs (main and additional tasks, ablations A1–A6, the Transformer memory variant, and the MPPI variant: 150 runs) gives roughly 630 GPU-hours on RTX 3090; the model-free baselines add a smaller amount. The full project, including preliminary and failed runs, required more compute than this.

C Rollout Horizon, Planner Horizon, and Decision Quality

This appendix reports the horizon behaviour promised in Section 6, including the extended open-loop range and its relation to closed-loop planning.

C.1 Rollout Error Beyond the Planning-Relevant Range

Table 1 reports $k \in \{3, 5, 7\}$ because that is what each method's native decision procedure consumes. Table 10 extends the comparison to $k = 10, 15, 20$: rollouts are open-loop from a replay-encoded $\mathbf{z}_t$ with no observation injected, scored by endpoint squared-error sum over three seeds. DreamerV3 is omitted because it is worst at every k (rising from 7.20 to 80.20).

Table 10: Open-loop rollout MSE up to k=20 (3 seeds, lower is better). **Bold** marks the better of the two methods at each k. Our error starts far lower but grows faster, and is overtaken at a task-dependent crossover.

Task	Method	k=3	k=5	k=7	k=10	k=15	k=20	Crossover
Finger	HaM-World	**0.72**	**1.43**	**2.53**	4.30	7.70	11.55	k=10
	TD-MPC2	3.48	3.58	3.61	**3.67**	**3.78**	**3.90**	
Reacher	HaM-World	**1.28**	**2.36**	3.68	5.75	9.65	14.05	k=7
	TD-MPC2	3.34	3.38	**3.42**	**3.48**	**3.58**	**3.68**	
Cheetah	HaM-World	**1.21**	**1.88**	**2.74**	**4.10**	6.65	9.45	k=15
	TD-MPC2	4.28	4.40	4.47	4.57	**4.74**	**4.91**	
Cartpole	HaM-World	**0.77**	**1.30**	**1.97**	**3.05**	**5.05**	7.30	k=20
	TD-MPC2	4.66	4.79	4.87	4.99	5.16	**5.34**	

HaM-World is lowest on 4/4, 4/4, 3/4, 2/4 and 1/4 tasks at $k = 3, 5, 7, 10, 15$ respectively, and on 0/4 at $k = 20$. Our error starts far lower but has a steeper slope; TD-MPC2 stays nearly flat on an already high plateau of $\sim$4. At the horizons the planners use ($k \leq 7$) our error is lower; beyond them it grows faster, and the next subsection tests whether open-loop error at k=20 matters for closed-loop control.

C.2 Does a Longer Planning Horizon Help?

A closed-loop planner re-anchors on a real observation at every environment step, so open-loop accuracy at k=20 is never required by the controller. To test what horizon actually helps, we vary only the CEM horizon H at fixed 100k-step checkpoints, leaving weights untouched. We sweep our own planner. TD-MPC2's MPPI horizon is also a test-time setting (H=3 in the official configuration [19], H=5 in our runs), and DreamerV3 acts through its learned actor with no test-time planning horizon, its imagination horizon (H=8 here; 15 officially) entering only actor–critic training. The planning-based methods therefore differ from it in kind: both HaM-World (H=6) and TD-MPC2 (H=5) query their model for a short action sequence and re-plan from a real observation at every step, so short horizons are what these planners consume. Table 1 compares them at $k \in \{3, 5, 7\}$, which covers those horizons, and our advantage over DreamerV3 extends well beyond them: at k=15 our rollout error is still below DreamerV3's at k=7 on all four tasks.

Table 11: Return as a function of our CEM planning horizon at fixed 100k checkpoints. Latency is relative to the native horizon. Longer planning is not uniformly better.

Task	Native H / return	Best H / return	H=20 return	Δ vs. native	Latency at H=20
Cartpole	6 / 58.9	12 / **63.62**	61.74	+4.82%	4.61×
Cheetah	6 / 184.4	14 / **196.16**	168.13	−8.82%	3.31×
Finger	6 / **254.0**	6 / **254.0**	36.00	−85.83%	3.24×
Reacher	6 / **150.6**	6 / **150.6**	126.00	−16.33%	3.31×

There is a foresight–accuracy trade-off: a larger H sees farther but compounds prediction error and multiplies latency. The best horizon lies in 6–14 on all four tasks, and at H=20 only Cartpole remains above its native setting while the others fall by 8.8%–85.8% at 3.2–4.6× the planning cost. The latency factor is a property of sampling-based planning at a fixed sample budget and applies equally to MPPI. Closed-loop control therefore consumes rollout accuracy at roughly 3–12 steps; the baselines' native horizons (5 and 8) lie in this range, which Table 1 measures and where our error is lower in 11 of 12 cells.

C.3 From Rollout Accuracy to Decision Quality

Lower rollout error is only useful if it changes which action the planner selects. The transmission is: bounded rollout error ⇒ reward/value heads queried at more accurate states ⇒ smaller score error ⇒ the *relative ordering* of candidates survives (a score error ϵ implies regret at most 2ϵ) ⇒ a near-optimal first action ⇒ replanning accumulates these into higher return. Note that the planner does not need correct scores, only correct ordering.

We test each link with a matched-candidate diagnostic: 600 scenarios (50 reproducible simulator states per task per seed) in which both models start from an *identical* state and history and score an *identical* set of 64 candidate action sequences.

Table 12: Matched-candidate diagnostic over 600 scenarios with identical states and identical 64-candidate sets. (a) score error, (b) whether that error inverts the ranking, (c) realized return when the first action is executed.

	Metric	HaM-World	TD-MPC2
(a)	Top-1 selection regret ↓	**0.3263**	0.6722
	Candidate reward MAE@6 ↓	**0.1851**	0.6071
(b)	Top-1 hit rate ↑	**33.17%**	15.00%
	Top-5 hit rate ↑	**54.67%**	37.17%
	Spearman ρ (score vs. true order) ↑	**0.4244**	0.2570
(c)	Avg. return over diagnostic states ↑	**122.82**	96.75
–	Planning latency per step (ms) ↓	25.39	**8.20**

Group (a) measures score error from two complementary angles. *Top-1 selection regret* is the true return of the set's actual best candidate minus that of the model's rank-1 candidate, giving the decision cost of the error; *reward MAE* compares each candidate's imagined cumulative reward with what the simulator realizes for the same actions, giving the error in score value. The first jointly reflects dynamics and reward/value prediction, since the rollout determines which states are queried and the heads determine their scores; the second isolates reward prediction. Group (b) asks whether those errors actually invert the ordering, via hit rate and rank correlation over all 64 candidates. Group (c) reports what the resulting choices accumulate to when the first action of each plan is executed.
The chain holds link by link: starting from lower rollout MSE (four-task mean 1.74/2.73 at k=5/7 versus 4.04/4.09), both score errors are smaller — and they move together, with Pearson correlation 0.474 between regret and reward MAE — yielding fewer ordering inversions and higher realized return, consistent with Table 1. The cost is planning latency, where we are roughly 3× slower per step than TD-MPC2.

C.4 Sensitivity to the Choice of Planner

Table 4 gives each method its native decision procedure (CEM for us, MPPI for TD-MPC2, imagined actor–critic for DreamerV3) under a unified budget, seeds, and protocol: MPPI is tuned around TD-MPC2's latent and value head, so substituting CEM would evaluate an untuned variant, and DreamerV3's imagined actor–critic is integral to its algorithm.
The Full-model CEM OOD entries below reuse the reference evaluation in Table 2. To probe sensitivity within our own method, we retrained HaM-World end-to-end with MPPI for the full 100k steps, so that MPPI participates in data collection, training, and deployment, with architecture and budget held fixed.

Table 13: End-to-end planner substitution within HaM-World (3 seeds). MPPI participates in data collection, training, and deployment.

Planner variant	Cheetah ret. ↑	Finger ret. ↑	Cartpole MSE ↓	Reacher MSE ↓	Mass OOD ↑	Damp. OOD ↑
HaM-World + CEM	**184.4**±8.4	**254.0**±14.1	**0.009**±0.003	**0.0778**±0.043	**158.5**±5.9	**139.7**±9.3
HaM-World + MPPI	166.1±11.4	231.7±16.2	0.019±0.005	0.104±0.021	143.2±8.9	124.6±9.2

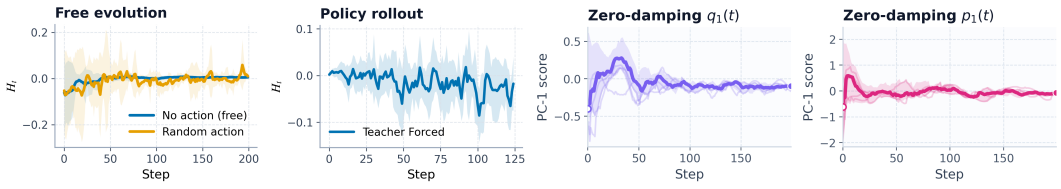
The CEM branch attains higher ID and OOD return and lower rollout MSE. Because MPPI also changes the data-collection and training distributions, this measures the end-to-end effect of the planner within HaM-World.

D Additional Results and Supplementary Analysis

D.1 Hamiltonian Energy, Geometry, and Control Coupling

This appendix gives the full mechanism diagnostics summarized in Section 6.3. Figures 3–5 collect all panels; the discussion in paragraphs (A)–(F) then refers back to them.

(A) The energy network is approximately stable and $\mathcal{H}$ matches the design expectation. The displayed mechanism analysis uses Cheetah Run as the running example. After reset, we inject uniformly random joint angular velocities in the range ± 5 rad/s, set joint damping to 0, and let the system freely evolve for 200 steps; each regime contains 10 episodes, with solid curves denoting



(a) Free evolution (b) Policy rollout (c) Zero-damping $\tilde{q}_1(t)$ (d) Zero-damping $\tilde{p}_1(t)$

Figure 3: Cheetah Run Hamiltonian diagnostics. **(a)** Hamiltonian energy under zero external force versus random action; **(b)** energy under policy rollout; **(c,d)** the two zero-damping Hamiltonian traces $\tilde{q}_1(t)$ and $\tilde{p}_1(t)$. Cheetah Run is used as the representative display.

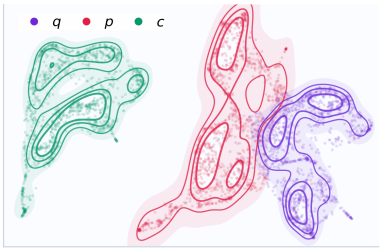
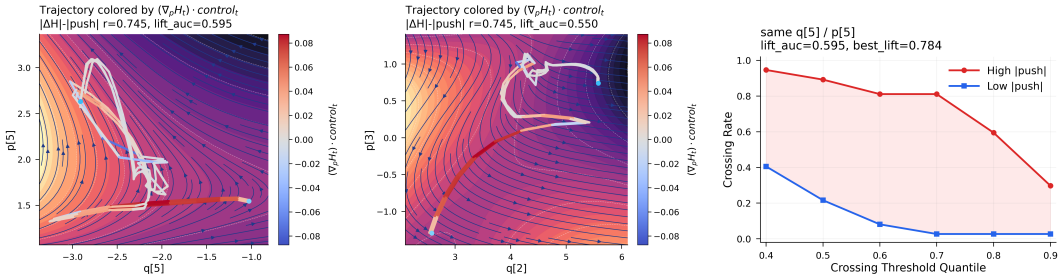


Figure 4: UMAP of the Cheetah Run latent coordinates: $\mathbf{q}$ and $\mathbf{p}$ form elongated manifolds with partial overlap, while $\mathbf{c}$ stays a more compact context cloud.



(a) $i{=}j{=}5$: streamline alignment (b) (q_2, p_3), $i{\neq}j$: deviation (c) High vs. Low $|\mathrm{push}|$ crossing rate

Figure 5: $\mathcal{H}$ contours and control coupling. The same diagnostic explains how external control moves rollouts across energy layers: high $|\mathrm{push}| = |\nabla_p \mathcal{H} \cdot \mathrm{control}|$ consistently yields larger equipotential crossing rates across thresholds.

the mean and shaded bands denoting $\pm$std across episodes. Figure 3a compares the same $\mathcal{H}$ head under no-action and random-action conditions. The no-action curve remains nearly flat within episodes (about 1% drift on Cheetah Run), with a narrow band across initial states. After random actions are introduced, the curve rises and widens, matching the physical expectation that external action continuously injects work into the system. This contrast is the key check: the learned scalar behaves differently under passive evolution and action-driven work, even though training imposes no hard energy conservation — consistent with Section 4.2, where action-free drift is expected at the percentage level.

(B) Policy rollouts show controlled energy variation. Figure 3b tracks H_t along policy teacher-forced rollouts from 10 different initial states. Policy execution continually applies control, so a flat conservation curve is not expected; instead H_t tracks the ongoing behavior without abrupt jumps, with a bounded band across initial states. Together with the first panel this separates passive energy stability from policy-induced modulation, consistent with Table 1: the q/p dynamics can vary under control while keeping finite-horizon error low.

(C) Geometry of the canonical and context subspaces. Figures 3c and 3d probe the Cheetah Run canonical traces behind the energy behavior. In the no-action, zero-damping setting, the randomized kick first produces a transient, but the uncontrolled Cheetah soon settles into a low-motion grounded posture; accordingly, $\tilde{q}_1(t)$ and $\tilde{p}_1(t)$ decay to small variations instead of drifting away, matching the passive rollout. The UMAP projection in Figure 4 gives a complementary view of the same latent

organization: $\mathbf{q}$ and $\mathbf{p}$ form elongated manifolds with partial overlap, while $\mathbf{c}$ stays as a more compact context cloud. $\mathbf{q}$ and $\mathbf{p}$ overlap partially, as expected for coordinates sampled from one trajectory manifold, but remain distinct from each other and from $\mathbf{c}$.

(D) Visualizing Hamiltonian geometry and control coupling. Figure 5a visualizes the learned Hamiltonian on a canonical slice (q_5, p_5): color denotes $\mathcal{H}(q,p)$, white curves are energy contours, and blue streamlines show $(\dot{q}, \dot{p}) = (\partial\mathcal{H}/\partial p, -\partial\mathcal{H}/\partial q)$. Teacher-forced rollout projections are colored by $\mathrm{push}_t = (\nabla_p \mathcal{H}_t) \cdot \mathrm{control}_t$. Local directions often follow the Hamiltonian flow; small $|\mathrm{push}|$ steps tend to move tangentially along contours, whereas large $|\mathrm{push}|$ steps more often cross energy layers. This is exactly the controlled-Hamiltonian story: tangential motion preserves the learned energy level locally, while control projected onto $\nabla_p \mathcal{H}$ performs work and changes energy. The quantitative correlations support the visual pattern, with $\mathrm{corr}(\mathrm{sign}\,\Delta\mathcal{H}, \mathrm{push}) = 0.757$ and $\mathrm{corr}(|\Delta\mathcal{H}|, |\mathrm{push}|) = 0.745$.

(E) Non-conjugate slice. Figure 5b projects the same rollout onto the non-conjugate slice (q_2, p_3). Because this is not a matched canonical pair, the projected streamlines are not expected to align; the panel checks that the control–energy association is not specific to one slice. The correlations $\mathrm{corr}(\mathrm{sign}\,\Delta\mathcal{H}, \mathrm{push}) = 0.757$ and $\mathrm{corr}(|\Delta\mathcal{H}|, |\mathrm{push}|) = 0.745$ are computed on the full latent state and are therefore slice-independent; the slice-specific contour-crossing lift remains above chance on (q_2, p_3) (lift_auc $= 0.550$) and is higher on the conjugate pair (0.595), as the canonical structure predicts.

(F) High/low $|\mathrm{push}|$ contour-crossing rates. Figure 5c quantifies this effect by thresholding contour crossings and splitting timesteps by $|\mathrm{push}|$. The threshold sweep avoids relying on a single contour resolution. As the threshold increases, both crossing rates decrease, but the High-$|\mathrm{push}|$ curve stays above the Low-$|\mathrm{push}|$ curve (lift_auc $= 0.595$, best lift $= 0.784$), confirming that stronger control projection along $\nabla_p \mathcal{H}$ makes contour crossings more likely.

D.2 Additional Tasks: Walker Run and Hopper Hop

The four core tasks underrepresent sustained coordination, momentum, and repeated contact, so we add two tasks under the identical protocol (100k env steps, seeds 7/8/9, same budget and evaluation pipeline). Learning curves are not reproduced for these two tasks; we report final return and the Avg. AUC pooled over all seed-runs of the core suite and of all six tasks (AUC computed as in Table 1).

Table 14: Additional tasks (mean±std, 3 seeds), same protocol and budget as Table 1: final return at 100k env steps, and the Avg. AUC pooled over all seed-runs of the core suite (Table 1) and of all six tasks.

Final return @ 100k ↑	HaM-World	TD-MPC2	DreamerV3	SAC	PPO
Walker Run	**151.99**±33.27	101.85±29.72	<u>104.57</u>±22.93	56.54±21.08	19.13±3.16
Hopper Hop	**34.46**±8.93	13.97±7.76	11.30±0.27	<u>14.70</u>±1.22	4.70±0.21
Avg. AUC ↑, core suite (4 tasks)	**117.9**	<u>107.7</u>	58.6	70.9	30.3
Avg. AUC ↑, all six tasks	**91.9**	<u>83.3</u>	51.7	52.8	22.7

HaM-World is best on both, with gains over TD-MPC2 of $+49\%$ on Walker Run and $+147\%$ on Hopper Hop. Combined with Table 1, this gives first place on four of six tasks and second on the remaining two. Pooled over all 18 seed-runs per method, the six-task Avg. AUC is 91.9 against 83.3 for TD-MPC2 ($+10.3\%$), so the core-suite sample-efficiency advantage carries over. These results extend the evaluation to control requiring sustained coordination and repeated contact, supporting the combined memory and geometric design.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction scope the claims to six state-based DMC tasks, planning-relevant rollout horizons, and absolute OOD return. Sections 6–7 and Appendix C report supporting results and the long-horizon crossover.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion discusses the limits of state-based simulation, the absence of pixel and robotic-manipulation experiments, and the first-order local nature of the Soft-Hamiltonian prior.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 4.2 and Appendix A state the local mechanism, unrolled core update, and first-order energy decomposition for steps without gradient clipping or latent norm projection, with the required smoothness assumption. Appendices A.3 and B.7 explain the safeguards and the scope of the derivation. No global convergence or finite-horizon prediction-error guarantee is claimed.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5 and Appendix B specify the tasks, 100k-step budget, seeds 7/8/9, architectures, objectives, planner, optimization settings, and pseudocode. Appendix B.12 specifies 10 OOD episodes per seed per condition, with results summarized across three seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public repository linked in the abstract provides the implementation, baseline agents, configurations, training and evaluation scripts, result exports, and installation and execution instructions. The environments are public DeepMind Control Suite tasks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix B give the training and evaluation settings, including optimizer, planner, seed protocol, and OOD perturbations. Appendix B.5 states that a single global loss-weight configuration is used unchanged across tasks, seeds, and OOD conditions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The principal result tables report mean and standard deviation across three seeds, as stated in their captions. Normalized retention is explicitly a ratio of reported means, and the extended rollout-error table reports three-seed means rather than significance tests (Appendices B.13 and C).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware and runtime paragraph in Appendix B specifies the GPU type and memory, one GPU per seed, estimated runtimes for each individual run, and an estimated total of roughly 630 GPU-hours for the reported model-based runs; it also discloses that the full project, including preliminary and failed runs, required more compute than reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The work uses standard simulated-control benchmarks and involves no human subjects, sensitive personal data, or deployed decision systems. It follows the NeurIPS Code of Ethics in this research setting.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: More sample-efficient simulated-control learning and structured latent diagnostics may support more efficient and interpretable model-based RL research. Controllers transferred to safety-critical systems may still fail under distribution shift; OOD return is a robustness diagnostic, not a safety certification.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released assets concern simulated continuous-control research and do not include scraped personal data or high-risk pretrained generative models requiring controlled release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The benchmark (DeepMind Control Suite, MuJoCo) and method (TD-MPC2, DreamerV3) sources are cited, and their licences – Apache-2.0 and MIT respectively – are stated in Appendix B. Baseline reimplementations are described there, and the released code is distributed under the repository's licence.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the reference implementation, with installation and execution instructions, configurations, and result exports in the linked repository. Appendix B documents the architecture, objectives, planner, pseudocode, and evaluation protocol; no new human-derived dataset is released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research uses simulated-control environments and involves neither crowdsourcing nor human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved, so IRB or equivalent approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not an important, original, or non-standard component of the method or experiments. Writing, editing, or formatting assistance does not affect the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.