arXiv:2605.05951v1 [cs.AI] 7 May 2026

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

Haoyun Tang[1*] **Haodong Cui**[2*] **Keyao Xu**[3]
Kun Wang[4†] **Zhandong Mei**[1†]
[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. We argue that this instability reflects two missing structures in planner-facing latents: history-conditioned memory for approximate Markov completeness, 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 $(\mathbf{q}, \mathbf{p})$ 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 four DeepMind Control Suite tasks, HaM-World reaches the highest Avg. AUC (117.9, $+9.5\%$), reduces long-horizon rollout error to 45% of a strong baseline model, and wins 11/12 $k \in \{3, 5, 7\}$ MSE cells. Under 12 OOD perturbations spanning dynamics shifts, action delay, and observation masking, HaM-World achieves the highest return in every condition, with average OOD-return gains of 10.2% on Finger Spin and 13.6% on Reacher Easy. Mechanism diagnostics further show bounded action-free Hamiltonian-energy drift, structured energy variation under policy rollouts, and coherent control-induced energy transfer, supporting the intended Soft-Hamiltonian dynamics design. 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 entangles configuration, momentum, and task semantics, causing multi-step prediction errors to grow without structural constraints during planning [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

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

Preprint.

as a Markov state; 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 Markov completeness and geometric consistency as two coupled stages of unified latent dynamics*. Memory supplies the conditioning needed for the latent dynamics to be approximately Markov, upon which Soft-Hamiltonian structure in $(\mathbf{q}, \mathbf{p})$ constrains error accumulation and stabilizes long-horizon rollouts. 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; (ii) **Mechanism-aware evaluation.** We evaluate return, $k \in \{3, 5, 7\}$ rollout MSE, 12 out-of-distribution (OOD) perturbations, and Hamiltonian mechanism analysis on four DeepMind Control Suite (DMC) tasks [12]; (iii) **Empirical gains.** HaM-World reaches the highest Avg. AUC (117.9, $+9.5\%$), reduces long-horizon rollout error to 45% of a strong baseline, wins 11/12 MSE cells, and obtains the highest return in every OOD condition, with 10.2% and 13.6% average OOD-return gains on Finger Spin and Reacher Easy, respectively.

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 breaks the Markov assumption and makes history-dependent representations 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 but compresses history through fixed 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 only as a history-conditioned input to the same latent dynamics, enabling approximate Markov completeness within the planner state rather than through a parallel rollout model or a planner-specific recurrent module outside the shared dynamics.

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, approximately Markov under partial observations, and directly useful for reward/value prediction and action search.

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, with no separate actor. The overall architecture is shown in Figure 1.

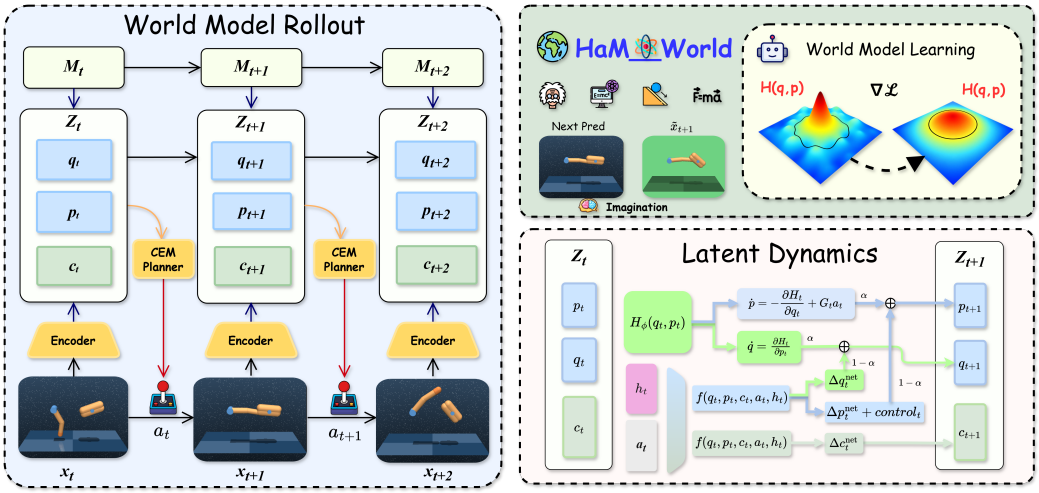


Figure 1: Architecture of HaM-World. Observations are encoded into Hamiltonian state $(\mathbf{q}_t, \mathbf{p}_t)$, context $\mathbf{c}_t$, and Mamba memory $\mathbf{h}_t$; a shared latent transition supports reward/value prediction and CEM planning inside one unified planner-facing interface without a separate actor.

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.

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

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

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.

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. 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. It does not define a parallel rollout: the model has one planner-facing latent dynamics, and memory fills in the Markov approximation under partial observability, action delay, and long horizons during imagined search rather than through an auxiliary recurrence hidden from CEM rollouts.

4.2 Mechanism-Oriented Stability Analysis

We package the stability argument as a local mechanism statement: memory makes the latent approximately Markov, while the Soft-Hamiltonian pair biases the planner-facing coordinates toward energy-organized rollout directions during repeated imagination.

Soft-Hamiltonian Stability Mechanism. For action-free rollouts, the explicit control drive vanishes and the Hamiltonian component is first-order energy-orthogonal on the canonical pair: $\nabla H(\mathbf{s}_t)^\top \xi_t = 0$ with $\xi_t = (\partial_{\mathbf{p}} H, -\partial_{\mathbf{q}} H)$. Appendix A.1 shows that the remaining drift is governed by alignment residual and curvature/discretization terms. This yields the diagnostic prediction tested later: no-action energy should drift slowly (Figure 3), whereas action work should appear as push-dependent contour crossing (Figure 4). For finite-horizon rollouts, Appendix A.2 gives $e_k \le \varepsilon(1 + L + \cdots + L^{k-1})$, with ε one-step error and L local expansion; memory acts on ε via history conditioning, while the Soft-Hamiltonian $(\mathbf{q}, \mathbf{p})$ pair acts on sources of L through energy-tangent dynamics and scaled residual/control/context channels.

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 for reproducibility.

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.

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

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 without a separate planner state.

5 Experimental Setup

We evaluate on Reacher Easy, Finger Spin, Cheetah Run, and Cartpole Swingup from the DeepMind Control Suite [12], using state observations, 100k environment steps, and 3 seeds. We report final return, AUC, $k \in \{3, 5, 7\}$ latent-rollout MSE, 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]). All methods share the same sample budget and evaluation protocol. 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 Hamiltonian mechanism analysis that connects the gains to the proposed memory-and-geometry design.

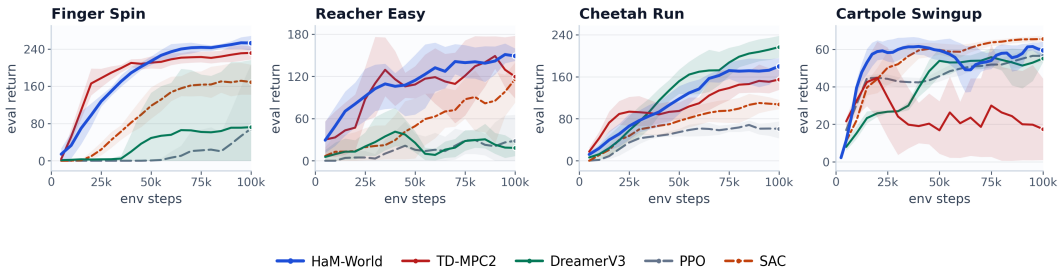


Figure 2: Return learning curves over four tasks (mean curve with seed envelope). HaM-World obtains the highest final return on Finger and Reacher and the second-best final return on Cheetah and Cartpole; when summarized by training AUC, it obtains the highest average score.

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.

Method	Finger Spin			Reacher Easy			Cheetah Run			Cartpole Swingup			Avg.
(I) Final return @ 100k env steps ↑ (Avg. = AUC over training)													
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), improving over TD-MPC2 by +9.5% (117.9 vs. 107.7), SAC by +66.3%, and DreamerV3 by +101.2%. 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). Thus, the gain is not a single endpoint artifact: it appears as higher training-area efficiency while remaining competitive across all four tasks.

Observation 2: planner-facing rollouts remain stable as k increases. 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 ($\leq 4\%$ 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). This ↓MSE pattern supports the intended stability budget: Soft-Hamiltonian geometry regularizes the repeated $(\mathbf{q}, \mathbf{p})$ rollout actually queried by CEM.

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).

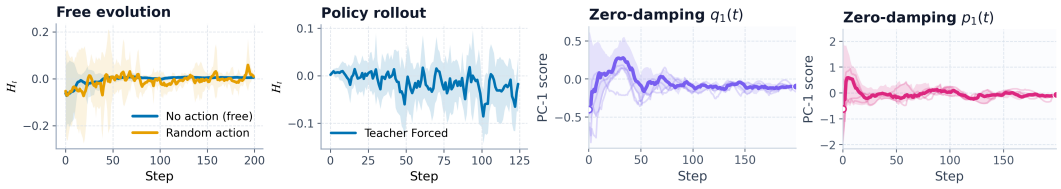
Observation 3: HaM-World retains performance under all 12 OOD perturbations. Table 2 shows that HaM-World obtains the highest return in every OOD column. On Finger Spin, its OOD average is 184.3±4.0, exceeding TD-MPC2 (167.3±7.8, +10.2%) and SAC (118.4±13.1, +55.7%),

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

while also leading under delay= 2 (74.4) and mask 30% (91.5). On Reacher Easy, it reaches 149.9±5.6 versus TD-MPC2's 132.0±37.2 (+13.6%), with much lower variance. These gains suggest that selective memory helps partial observability, while the Hamiltonian pair improves extrapolation under dynamics shifts in the same zero-shot protocol.

6.3 Hamiltonian Energy, Geometry, and Control Coupling



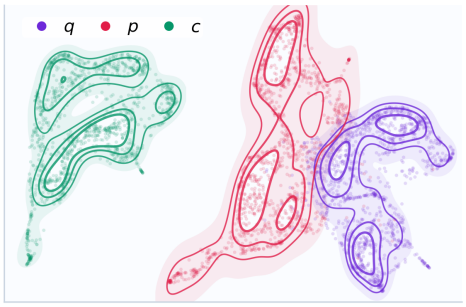
Free evolution | Policy rollout | Zero-damping $\tilde{q}_1(t)$ | Zero-damping $\tilde{p}_1(t)$

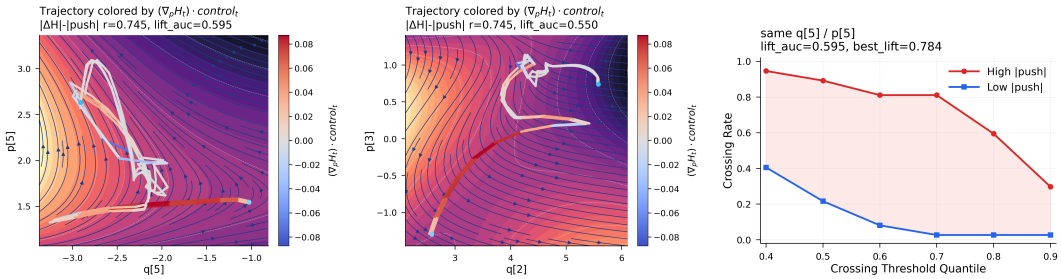
Figure 3: Cheetah Run Hamiltonian diagnostics. From left to right: Hamiltonian energy under zero external force versus random action, energy under policy rollout, and the two zero-damping Hamiltonian traces $\tilde{q}_1(t)$ and $\tilde{p}_1(t)$.

(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 the mean and shaded bands denoting ±std across episodes. The first panel of Figure 3 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 does not impose hard energy conservation. It is also consistent with the energy-stability analysis in Section 4.2: with the scheduled mixing coefficient used in the model and moderate alignment-residual/curvature terms, action-free drift is expected to remain at the percentage level.

(B) Policy rollouts show controlled energy variation. The second panel of Figure 3 tracks H_t along policy teacher-forced rollouts from 10 different initial states. Unlike the no-action free-evolution plot, policy execution continually applies control, so a flat conservation curve is not expected. Instead, H_t changes with the ongoing policy behavior while avoiding abrupt jumps, and the band remains bounded across initial states. Together with the first panel, this separates passive energy stability from policy-induced energy modulation. This behavior is consistent with the rollout-MSE results in Table 1: the Soft-Hamiltonian q/p dynamics can vary under control while still keeping finite-horizon prediction error low over the planning horizon.

(C) Geometry of the canonical and context subspaces. The last two panels of Figure 3 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 inset 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. The q/p overlap is expected because both coordinates are sampled from the same trajectory manifold; the useful signal is that they do not fully collapse into one mixed cloud or into $\mathbf{c}$ despite this overlap.





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

Figure 4: $\mathcal{H}$ 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.

(D) Visualizing Hamiltonian geometry and control coupling. Figure 4a 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 slices show a different viewpoint. Figure 4b projects the same rollout onto the non-conjugate slice (q_2, p_3). Because this is not a matched canonical pair, the projected streamlines need not align as cleanly as in (q_5, p_5). The key observation is nevertheless unchanged: larger $|\mathrm{push}|$ remains associated with stronger energy change and contour crossing, so the effect is not a visual artifact of one favorable slice or projection.

(F) High/low $|\mathrm{push}|$ contour-crossing rates. Figure 4c 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.

6.4 Result Summary

Across metrics, the evidence matches the hierarchy in Section 1. HaM-World has the highest Avg. AUC (117.9), wins 11 out of 12 cells in $k \in \{3, 5, 7\}$ MSE, and obtains the highest return under all 12 OOD conditions. The Cheetah Run diagnostics show near-conservation without action, structured energy variation during policy rollouts, and separated q/p/c geometry; the control-coupling slice further shows that action push modulates energy-layer crossing. Thus, selective memory supplies

input-side Markov completeness, while Soft-Hamiltonian geometry on $(\mathbf{q}, \mathbf{p})$ supplies output-side rollout consistency within the same planner-facing latent dynamics used by CEM. Taken together, the gains are consistent across control, rollout, and OOD stress tests, so they look structural rather than like a single-metric artifact of the benchmark setup or reporting choice.

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.

Table 3: Core ablations (3 seeds, mean±std). A1 removes geometric structure; A2/A3 alter memory.

	Return ↑		MSE@6 ↓		Reacher OOD ↑	
Variant	Cheetah	Finger	Cartpole	Reacher	mass×0.7	damp×2.0
Full HaM-World	$\mathbf{184.4}_{\pm 8.4}$	$\mathbf{254.0}_{\pm 14.1}$	$\mathbf{0.009}_{\pm 0.003}$	$\mathbf{0.0778}_{\pm 0.043}$	$\mathbf{159.7}_{\pm 3.9}$	$\mathbf{139.9}_{\pm 7.7}$
A1: w/o geom. struct.	$169.8_{\pm 13.5}$	$247.7_{\pm 12.5}$	$0.012_{\pm 0.003}$	$0.0852_{\pm 0.022}$	$154.4_{\pm 15.2}$	$131.9_{\pm 22.9}$
A2: Memory = None	$59.2_{\pm 11.2}$	$34.0_{\pm 23.8}$	$0.110_{\pm 0.068}$	$0.287_{\pm 0.075}$	$103.6_{\pm 5.1}$	$115.4_{\pm 9.0}$
A3: Memory = GRU	$121.1_{\pm 32.1}$	$235.6_{\pm 4.6}$	$0.026_{\pm 0.007}$	$0.148_{\pm 0.021}$	$157.6_{\pm 7.7}$	$138.4_{\pm 3.0}$

Ablation MSE@6 reports per-coordinate rollout MSE because the variants share the same latent size and evaluation pipeline, making the averaged metric a within-family ablation measure; Table 1 reports coordinate-summed latent error, so their scales differ. The ablations reveal a hierarchy. A2 (no memory) is most damaging: 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 $103.6/115.4$. Thus, under partial observability and action delay, memory is the dominant prerequisite in this setting. 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 weaker state completeness than selective SSM memory. A1 keeps the planner usable but weakens the geometric prior: the Reacher MSE gap is small, yet return and OOD scores still fall ($184 \to 170$, $254 \to 248$; $159.7 \to 154.4$, $139.9 \to 131.9$). Thus, memory makes the latent state usable, while Soft-Hamiltonian structure on $(\mathbf{q}, \mathbf{p})$ helps lift the ceiling on rollout robustness and OOD planning quality.

This also explains why A1 should be read conservatively: removing q/p/c does not remove capacity, the planner, or recurrence, so short local transitions can still be fitted. What it removes is the coordinate target on which Hamiltonian alignment and action-free energy regularization can organize the latent repeatedly reused by CEM during imagined planning.

8 Conclusion

HaM-World decomposes planner-facing latent dynamics into input-side Markov completeness through Mamba selective memory and output-side geometric consistency through a Soft-Hamiltonian bias on $(\mathbf{q}, \mathbf{p})$. On four DMC tasks, HaM-World reaches an Avg. AUC of 117.9, wins 11 out of 12 cells for $k \in \{3, 5, 7\}$ MSE, and obtains the highest return across all 12 OOD conditions. Ablations and geometry diagnostics support the same hierarchy: memory makes the latent state usable, while Soft-Hamiltonian structure in the Hamiltonian pair stabilizes longer rollouts and dynamics perturbations; $\mathbf{c}$ captures 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. This study uses state observations and four DMC tasks, so scaling to pixels, broader morphologies, and discontinuous contact remains open. The diagnostics support the learned interface but do not prove that the learned dynamics are globally Hamiltonian. Future work should test the same planner-facing q/p/c split from pixels and use stronger geometric validation to distinguish local energy organization from true Hamiltonian structure, especially in contact-rich domains where impacts break smooth-flow assumptions.

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, 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

The theoretical analysis in this appendix is not intended to provide a global convergence guarantee for the learned dynamics. Instead, it formalizes the local mechanism used in Section 4.2: the Hamiltonian component is energy-orthogonal at first order, the alignment residual and learned control drive account for non-conservative effects, and multi-step rollout error is governed by one-step model error and the local expansion of the learned transition. All statements are local to the finite rollout region visited by CEM and the evaluation policy.

Notation and assumptions. Let $\mathbf{s}_t = (\mathbf{q}_t, \mathbf{p}_t)$ denote the canonical pair. In the implementation, the network branch is softly aligned with the uncontrolled Hamiltonian direction, while the learned control drive is added separately to the $\mathbf{p}$ update. We therefore write

$$\Delta \mathbf{s}_t^{\text{net}} = \xi_\phi(\mathbf{s}_t) + \boldsymbol{\delta}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t),$$

where $\boldsymbol{\delta}_\phi$ is the remaining alignment residual after the Hamiltonian loss. The Soft-Hamiltonian update can then be written as

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \xi_\phi(\mathbf{s}_t) + \mathbf{u}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t) + (1-\alpha)\boldsymbol{\delta}_\phi(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t) \tag{8}$$

where

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

Here $\mathbf{u}_\phi$ is the explicit control drive produced by the control map in the code; because it is multiplied by $\mathbf{a}_t$, it vanishes for zero action. We assume that, on a compact rollout region $\mathcal{R}$, H_ϕ is smooth enough to admit the third-order Taylor remainder below, with $\|\nabla^2 H_\phi(\mathbf{s})\| \le L_H$. We also assume that the alignment residual and control increments are bounded by $\|\boldsymbol{\delta}_\phi\| \le M_\delta$ and $\|\mathbf{u}_\phi\| \le M_u$, and that the Taylor remainder is bounded by $C_H\|\Delta\mathbf{s}_t\|^3$ for the one-step increment $\Delta\mathbf{s}_t = \mathbf{s}_{t+1} - \mathbf{s}_t$. These are local assumptions for the finite CEM/evaluation rollouts, where actions and horizons are bounded, rather than global guarantees.

A.1 Energy Variation Decomposition

We first show why the Hamiltonian component alone does not change the learned energy at first order. Let $J = \left(\begin{smallmatrix} 0 & I \\ -I & 0 \end{smallmatrix}\right)$ be the canonical symplectic matrix, so that $\xi_\phi(\mathbf{s}) = J\nabla H_\phi(\mathbf{s})$. Since $J^\top = -J$,

$$\nabla H_\phi(\mathbf{s})^\top \xi_\phi(\mathbf{s}) = \nabla H_\phi(\mathbf{s})^\top J \nabla H_\phi(\mathbf{s}) = 0. \tag{9}$$

This is the first-order cancellation used in the main text. For one discrete step, Taylor expansion gives

$$\Delta\mathcal{H}_t = \nabla H_\phi(\mathbf{s}_t)^\top \Delta\mathbf{s}_t + \frac{1}{2}\Delta\mathbf{s}_t^\top \nabla^2 H_\phi(\mathbf{s}_t)\Delta\mathbf{s}_t + R_3(\Delta\mathbf{s}_t), \qquad |R_3(\Delta\mathbf{s}_t)| \le C_H\|\Delta\mathbf{s}_t\|^3. \tag{10}$$

Substituting Eq. (8) and using Eq. (9) yields

$$\Delta\mathcal{H}_t = \nabla H_\phi(\mathbf{s}_t)^\top\big(\mathbf{u}_\phi + (1-\alpha)\boldsymbol{\delta}_\phi\big) + \frac{1}{2}\Delta\mathbf{s}_t^\top \nabla^2 H_\phi(\mathbf{s}_t)\Delta\mathbf{s}_t + R_3(\Delta\mathbf{s}_t). \tag{11}$$

Thus, to first order, energy can change only through the alignment residual and the explicit control drive. The pure Hamiltonian component appears only in curvature and higher-order terms after discretization.

Using the boundedness assumptions and $\|\Delta\mathbf{s}_t\| \le \|\xi_\phi(\mathbf{s}_t)\| + M_u + (1-\alpha)M_\delta$, Eq. (11) implies

$$|\Delta\mathcal{H}_t| \le \|\nabla H_\phi(\mathbf{s}_t)\|\big(M_u + (1-\alpha)M_\delta\big) + \frac{1}{2}L_H\|\Delta\mathbf{s}_t\|^2 + C_H\|\Delta\mathbf{s}_t\|^3. \tag{12}$$

For action-free rollouts with $\mathbf{a}_t = 0$ and hence $M_u = 0$, this reduces to

$$\begin{aligned} |\Delta\mathcal{H}_t| \le{}& (1-\alpha)\|\nabla H_\phi(\mathbf{s}_t)\|M_\delta \\ &+ \frac{1}{2}L_H\big(\|\xi_\phi(\mathbf{s}_t)\| + (1-\alpha)M_\delta\big)^2 \\ &+ C_H\big(\|\xi_\phi(\mathbf{s}_t)\| + (1-\alpha)M_\delta\big)^3. \end{aligned} \tag{13}$$

Equation (13) explains the empirical energy-drift behavior in the first panel of Figure 3: in the no-action regime, the explicit control term vanishes and the remaining drift is controlled by $(1-\alpha)M_\delta$ plus curvature/discretization terms, which is why the passive-energy trace stays nearly flat. When actions are present, the additional term $\nabla H_\phi(\mathbf{s}_t)^\top \mathbf{u}_\phi$ corresponds to action work, matching the push-based diagnostics in Figure 4.

A.2 Error Propagation Analysis

We next make precise the finite-horizon error statement used in Section 4.2. Let T denote the environment-induced latent transition under a fixed action sequence and let $\widehat{T}_\phi$ denote the learned transition used by CEM. Consider rollouts

$$\mathbf{z}_{i+1} = T(\mathbf{z}_i), \qquad \widehat{\mathbf{z}}_{i+1} = \widehat{T}_\phi(\widehat{\mathbf{z}}_i), \qquad \widehat{\mathbf{z}}_0 = \mathbf{z}_0.$$

On the finite rollout region, assume a uniform one-step model error and local Lipschitz continuity:

$$\|\widehat{T}_\phi(\mathbf{z}) - T(\mathbf{z})\| \le \varepsilon, \qquad \|T(\mathbf{z}) - T(\mathbf{z}')\| \le L\|\mathbf{z} - \mathbf{z}'\|.$$

Then, for $e_i = \|\widehat{\mathbf{z}}_i - \mathbf{z}_i\|$,

$$\begin{aligned} e_{i+1} &= \|\widehat{T}_\phi(\widehat{\mathbf{z}}_i) - T(\mathbf{z}_i)\| \\ &\le \|\widehat{T}_\phi(\widehat{\mathbf{z}}_i) - T(\widehat{\mathbf{z}}_i)\| + \|T(\widehat{\mathbf{z}}_i) - T(\mathbf{z}_i)\| \\ &\le \varepsilon + Le_i. \end{aligned} \tag{14}$$

Unrolling Eq. (14) with $e_0 = 0$ gives

$$e_k \le \varepsilon \sum_{i=0}^{k-1} L^i = \begin{cases} \varepsilon k, & L = 1, \\ \varepsilon (L^k - 1)/(L - 1), & L \neq 1. \end{cases} \tag{15}$$

This bound is intentionally local: it does not claim that learned neural dynamics are globally contracting. Rather, it identifies the two quantities that matter for finite-horizon planning: the one-step error ε and the effective local expansion factor L. The rollout consistency loss directly reduces ε over the horizons used by CEM, while the Soft-Hamiltonian update constrains the canonical pair to evolve along an energy-organized vector field, reducing uncontrolled expansion in the planner-facing coordinates.

Finally, the memory state affects the premise of this bound. Under partial observability, a single observation latent may not determine the next transition, increasing the apparent one-step error. Mamba memory supplies a history-conditioned input $\mathbf{h}_t$ to the same transition $\widehat{T}_\phi$, making the planner-facing latent closer to Markov on the finite rollout region. In this view, memory reduces the effective ε, while the Soft-Hamiltonian q/p structure reduces sources of local expansion that contribute to the effective L: the dominant branch is tangent to learned energy levels, the residual is scaled by $(1 - \alpha)$ and penalized by $\mathcal{L}_{\text{ham}}$, and non-conservative effects are routed through explicit control and context channels rather than a single unconstrained latent update. The empirical improvements in Table 1 and Table 3 are consistent with this two-part mechanism.

B Implementation Details

B.1 Shared Protocol and Method Hyperparameters

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	Reacher Easy / Finger Spin / Cheetah Run / Cartpole Swingup
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

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 only 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. Unlike the GRU prior in the baselines, it only outputs the history-conditioned features needed by the shared latent dynamics and does not implement a separate latent-rollout path.

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. The losses are therefore not independent add-ons: they are chosen so that the planner-facing latent dynamics remains 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.

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

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:

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

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

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 β_{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.

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

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.

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

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 deliberately 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.

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

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.

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

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.

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

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.

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 $\dot{\mathbf{q}} = \partial\mathcal{H}/\partial\mathbf{p}$, $\dot{\mathbf{p}} = -\partial\mathcal{H}/\partial\mathbf{q} + g(\mathbf{q})\mathbf{a}$.

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 and compute temperature-weighted weights $w^{(j)} = \mathrm{softmax}(G^{(j)}/\tau)$
    $\boldsymbol{\mu}_i = \sum_j w^{(j)} \mathbf{a}^{(j)}$, $\boldsymbol{\sigma}_i^2 = \sum_j w^{(j)} (\mathbf{a}^{(j)} - \boldsymbol{\mu}_i)^2 + \epsilon$
  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

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 $(\Delta\mathbf{q}_t, \Delta\mathbf{p}_t)$ is taken from Eq. (3) and both Hamiltonian gradients are evaluated at $(\mathbf{q}_t, \mathbf{p}_t)$. Thus, when $\alpha \to 1$, the canonical branch follows the Hamiltonian vector field through forward Euler, with the learned control drive still added to $\mathbf{p}$; it is not a symplectic Euler or leapfrog integrator. We use the Hamiltonian vector field as a *local geometric bias*, not as a strict symplectic numerical solver. When $\alpha \to 0$, the pair transition reduces to a data-driven residual update plus the same control drive. Intermediate α balances task fitting with the Hamiltonian bias while keeping training and inference on the same first-order update rule.

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. We do not introduce an explicit G-invariant or Lie-equivariant architecture for this head. Instead, the physics prior is imposed through the action-conditioned Hamiltonian form motivated by controlled Hamiltonian models [4, 9]. With state-based inputs, rotational/translational symmetries of $(\mathbf{q}, \mathbf{p})$ are partly handled implicitly by the encoder and EMA target. $\mathcal{L}_{\text{ham}}$ further aligns the q/p residual update with $\partial\mathcal{H}/\partial(\mathbf{q}, \mathbf{p})$, acting as a soft invariance constraint.

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 is designed to isolate the output-side geometric hypothesis rather than the input-side memory hypothesis. It 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. Therefore the comparison between Full HaM-World and A1 asks whether, after memory has already supplied a usable history-conditioned state, assigning part of the latent to a Hamiltonian pair provides additional rollout headroom. Under the Full model, $\mathcal{L}_{\text{ham}}$ has a well-defined target: it aligns the residual update of $(\mathbf{q}, \mathbf{p})$ with a scalar energy field $\mathcal{H}_\phi(\mathbf{q}, \mathbf{p})$, while $\mathbf{c}$ remains available for actuation, contact, and dissipative information that should not be forced into the Hamiltonian pair. Under A1, this target disappears because there is no distinguished pair on which the Hamiltonian vector field can act; the same capacity must represent conservative, controlled, and dissipative factors in a single 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. Instead, A1 keeps relatively mild return degradation but loses some return and OOD headroom, matching the expected signature of losing a geometric regularizer: short-horizon control can still be learned, while longer imagined rollouts and dynamics shifts become less stable.

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 state dimension 128. The concatenated input $(\mathbf{z}_t, \mathbf{a}_t)$ is projected to the model dimension; each layer uses input-dependent state updates, readout, and gating, and the output $\mathbf{h}_t$ enters the Hamiltonian-pair core and context updater within the unified latent dynamics. The selective scan provides input-dependent state filtering, which is the key advantage over a standard GRU and explains the degradation in ablation A3 when Mamba is replaced by GRU.

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 the Markov state $\mathbf{z}_t$, while memory fills in missing state information under partial observability.

B.10 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.11 OOD Evaluation Conditions

Table 7: OOD evaluation conditions on Reacher Easy and Finger Spin. Each condition is evaluated zero-shot with 3 episodes per seed and 3 seeds, for 9 evaluations total.

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)

The main OOD table reports raw zero-shot return. For retention diagnostics in the released scripts, $\mathrm{Retention}(\%) = \mathrm{Return}_{\mathrm{OOD}}/\mathrm{Return}_{\mathrm{ID}} \times 100$ 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.12 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.

The point of this setup is to test action-free energy drift and action-induced energy variation under reduced damping; it is a diagnostic setting rather than a claim that the simulated environment becomes exactly conservative.

B.13 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

(β_1=0.9, β_2=0.999), and target-update coefficient τ=0.01 for the EMA encoder/projector/value heads. DreamerV3 uses deterministic RSSM dimension 256 and 16 $\times$ 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. Under the strict 100k-step sample budget, their sample efficiency is substantially lower than the above model-based methods, consistent with the model-based RL literature [2, 19].

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 one seed and 100k environment steps, HaM-World takes about 4.5 h, TD-MPC2 about 3.5 h, and DreamerV3 about 3.0 h. OOD evaluation takes less than 5 min per condition per seed. The reported main experiments complete in about 10–12 h on the 8-GPU machine using seed-level parallelism, for roughly 100 GPU-hours in total.

C Additional Results and Supplementary Analysis

Representative rollout keyframes. The main text uses Cheetah Run as the mechanism example; Figure 5 provides representative best-seed rollout keyframes for the evaluated tasks.

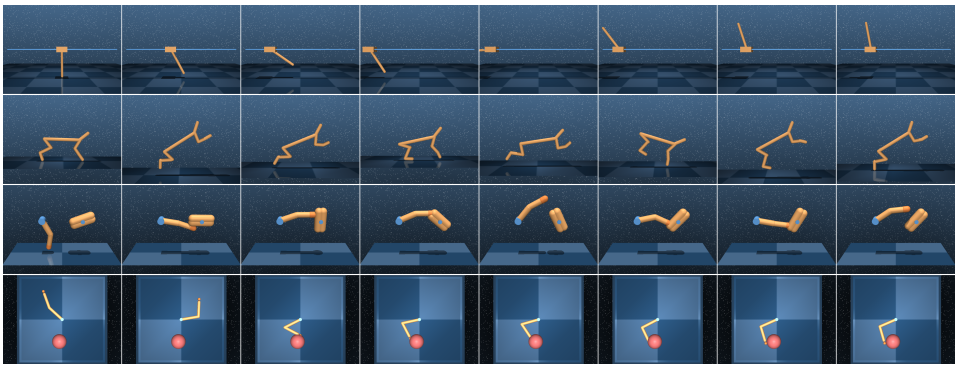


Figure 5: Representative keyframes from one best-seed evaluation rollout, with one row per task and uniformly sampled frames along the episode.