DreamSAC: Learning Hamiltonian World Models via Symmetry Exploration

Supplementary Material

6. Detailed Methodology

6.1. Work as Symmetry Breaking

Our symmetry-aware exploration aims to learn the internal Hamiltonian H_ϕ, which encodes the system's conservative dynamics and symmetries. By Noether's theorem, continuous symmetries in a physical system correspond to conserved quantities. For an autonomous (closed) Hamiltonian system, time-translation symmetry implies conservation of energy, meaning the Hamiltonian is constant along trajectories:

$$\begin{aligned}\frac{dH}{dt} &= \frac{\partial H}{\partial q}\dot{q} + \frac{\partial H}{\partial p}\dot{p} \\ &= \frac{\partial H}{\partial q}\left(\frac{\partial H}{\partial p}\right) + \frac{\partial H}{\partial p}\left(-\frac{\partial H}{\partial q}\right) = 0\end{aligned} \tag{8}$$

In such a closed system, observing passive evolution provides limited information about the underlying structure of H, as the system remains on a fixed energy level set.

To efficiently learn H across its entire domain, an agent must actively break this conservation. In our controlled setting (Eq. 11), the agent's action a_t acts as a non-conservative external force. The time evolution of the Hamiltonian in this open system becomes:

$$\begin{aligned}\frac{dH}{dt} &= \frac{\partial H}{\partial q}\dot{q} + \frac{\partial H}{\partial p}\dot{p} \\ &= \frac{\partial H}{\partial q}\left(\frac{\partial H}{\partial p}\right) + \frac{\partial H}{\partial p}\left(-\frac{\partial H}{\partial q} + g(q)a_t\right)\end{aligned} \tag{9}$$

Simplifying this yields a direct relationship between the change in the Hamiltonian and the external work done by the agent:

$$\frac{dH}{dt} = \left[\frac{\partial H}{\partial p}\right]^\top g(q)a_t = \dot{q}^\top F_{ext} = P_{ext} \tag{10}$$

where $F_{ext} = g(q)a_t$ is the effective external force and P_{ext} is the external power delivered to the system.

Equation (10) provides the theoretical justification for our intrinsic reward, $r_{sym} \approx |\Delta H|$. By maximizing $|H(Z_{t+1}) - H(Z_t)|$, the agent is intrinsically motivated to perform actions that maximize work done on or by the system. This actively steers the system away from its current energy level sets, effectively "breaking" the symmetries that hold for unforced trajectories and exploring new regions of the phase space essential for robustly learning the global structure of H_ϕ.

6.2. Hamiltonian Dynamics and Discretization

Our world model's latent dynamics follow the controlled Hamiltonian equations of motion. In continuous time, for a latent state $Z = (\boldsymbol{q}, \boldsymbol{p})$ and control action a_t, these are:

$$\begin{aligned}\dot{q} &= \frac{\partial H_\phi}{\partial p}(\boldsymbol{q}, \boldsymbol{p}) \\ \dot{p} &= -\frac{\partial H_\phi}{\partial q}(\boldsymbol{q}, \boldsymbol{p}) + g_\phi(q)a_t\end{aligned} \tag{11}$$

Dual Integration Strategy. Discretizing these equations requires careful consideration. While Symplectic Integrators are essential for long-term energy conservation, they can sometimes yield less stable gradients during the early phases of training deep generative models compared to standard numerical methods. To balance training stability with physical faithfulness during imagination, we employ a dual integration strategy:

- **World Model Training (Gradient Stability):** During the pretraining and adaptation phases, where we optimize the world model parameters ϕ via the ELBO (short-horizon predictions), we employ a standard explicit Euler integrator for maximum gradient stability and computational efficiency.
- **Imagination & Inference (Physical Conservation):** During actor-critic training (imagined rollouts) and evaluation, where long-horizon physical consistency is paramount, we switch to the explicit Symplectic Leapfrog Integrator.

Discussion of Limitations. We acknowledge that optimizing the vector field under Euler discretization while evaluating it with Leapfrog introduces a theoretical gap: the model is not strictly "trained to be symplectic." However, this is a deliberate design choice. Training directly through a multi-step Symplectic Integrator can lead to exploding gradients in the early stages of learning deep generative models. By using a small time step Δt, the discretization error between Euler and Leapfrog is minimized. Empirically, we find that the vector field learned via Euler approximation sufficiently captures the underlying continuous Hamiltonian dynamics. Crucially, with our sufficiently small integration step ($\Delta t = 0.1$), the learned dynamics approach the continuous ODE limit. This ensures that the Symplectic Integrator utilized during inference theoretically guarantees the preservation of symplectic structure and energy conservation, rendering it a structurally grounded choice rather than merely an empirical heuristic.

Explicit Leapfrog with Control. Despite using a general parameterized Hamiltonian $H_\phi(q, p)$, our network architecture ensures efficient computation of partial derivatives via automatic differentiation, allowing for an explicit approximation. We apply the control input a_t (assumed constant over Δt) during both momentum half-steps. The update from (q_t, p_t) to (q_{t+1}, p_{t+1}) is:

1. **Half-step Momentum Update:**

$$p_{t+1/2} = p_t + \frac{\Delta t}{2}\left(-\frac{\partial H_\phi}{\partial q}(q_t, p_t) + g_\phi(q_t)a_t\right) \quad (12)$$

2. **Full-step Position Update:**

$$q_{t+1} = q_t + \Delta t \cdot \frac{\partial H_\phi}{\partial p}(q_t, p_{t+1/2}) \quad (13)$$

3. **Full-step Momentum Completion:**

$$p_{t+1} = p_{t+1/2} + \tfrac{\Delta t}{2}\left(-\tfrac{\partial H_\phi}{\partial q}(q_{t+1}, p_{t+1/2}) + g_\phi(q_{t+1})a_t\right) \quad (14)$$

This formulation ensures that the external work done by the agent is correctly accounted for in the system's momentum change while preserving the symplectic structure of the internal dynamics during rollout.

6.3. Viewpoint-Robustness

To ensure the latent state Z_t captures underlying physical dynamics rather than nuisance viewpoint parameters, we employ a self-supervised contrastive loss, $\mathcal{L}_{vr}$. This loss explicitly incentivizes the encoder q_ϕ to be invariant to image transformations that preserve the physical state.

Augmentation Pipeline. We construct positive pairs (x_t^A, x_t^B) from a single observation x_t by applying a stochastic sequence of augmentations. To effectively simulate complex viewpoint changes and environmental variations, we utilize the following augmentations in order (all implemented using standard TorchVision transforms):

- **Random Resized Crop:** Simulates changes in camera distance and focus. We use a scale range of [0.8, 1.0] and an aspect ratio range of [0.75, 1.33].
- **Random Perspective:** Crucial for simulating drastic camera angle shifts. We use a distortion scale of 0.5 applied with a probability of 0.5.
- **Color Jitter:** Simulates lighting variations. We adjust brightness, contrast, saturation, and hue with factors of [0.4, 0.4, 0.4, 0.1] respectively, applied with probability 0.8.
- **Gaussian Blur:** Prevents reliance on high-frequency artifacts. We use a kernel size of 23(roughly 10% of image size) and a sigma range of [0.1, 2.0], applied with probability 0.5.

Loss Implementation Details. We adopt the InfoNCE loss. For a minibatch of size K, we use the other $2(K-1)$ augmented views within the same batch as negative samples. No external memory bank is used. For a positive pair (i, j), the loss is:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(Z_i, Z_j)/\tau)}{\sum_{k=1,k\neq i}^{2K} \exp(\text{sim}(Z_i, Z_k)/\tau)} \quad (15)$$

where $\text{sim}(u, v)$ is cosine similarity.

Temperature Parameter τ Analysis. The temperature parameter τ controls the sharpness of the distribution and the "hardness" of negative samples. We use $\tau = 0.07$ for all main experiments. Table 5 shows a sensitivity analysis on the 'FetchPush New View' task, demonstrating that our chosen value provides an optimal balance between learning discriminative features and training stability.

Preserving Physical Signal under Augmentation. A critical concern is whether strong visual augmentations might degrade the model's ability to identify fine-grained physical properties (*e.g.*, mass or friction) required for parametric generalization. We address this by strictly limiting our augmentation pipeline to **spatial and chromatic transformations** (*e.g.*, crop, perspective, color) while strictly forbidding temporal augmentations (*e.g.*, frame skipping, speed jitter). Since physical parameters like mass are inferred from the *temporal evolution* of the state (*i.e.*, acceleration under force), our strategy ensures that the temporal structure remains intact. The contrastive loss $\mathcal{L}_{vr}$ thus forces the encoder to discard static visual nuisances (viewpoint) while retaining the temporal dynamics information essential for the Hamiltonian world model to infer implicit physical parameters.

Table 5. Sensitivity analysis of the temperature parameter τ on Reacher Hard (Unseen View) mean reward (mean over 5 seeds).

Temperature τ	0.05	**0.07 (Ours)**	0.1	0.2	0.5
Mean Reward	312.4	**321.9**	308.7	245.3	112.8

6.4. Symmetry Exploration Reward Annealing

The intrinsic reward $r_{int,t}$ combines a standard novelty-based exploration bonus (r_{RND}) and our proposed physics-based symmetry probing bonus (r_{sym}). Since these two signals originate from different sources (prediction error vs. Hamiltonian difference) and fluctuate significantly during training, proper normalization is crucial.

Following the standard practice in Dreamer [17] and RND [4], we normalize each reward component using

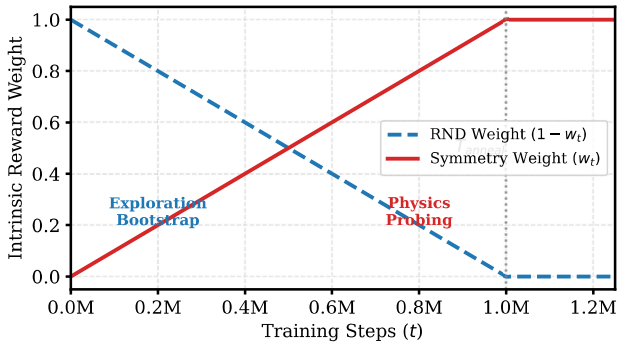

Figure 4. Visual illustration of the linear annealing schedule for an environment with $T_{anneal} = 10^6$.

its **running standard deviation**. We maintain a running standard deviation σ_i for each reward stream $i \in \{RND, sym\}$. The normalized rewards used in the annealing equation are calculated as:

$$\tilde{r}_{i,t} = \frac{r_{i,t}}{\sigma_{i,t}} \tag{16}$$

The final annealed reward is then computed as:

$$r_{int,t} = (1 - w_t) \cdot \tilde{r}_{RND,t} + w_t \cdot \tilde{r}_{sym,t} \tag{17}$$

This dynamic normalization ensures that the magnitude of the intrinsic rewards remains consistent (≈ 1.0) throughout the training process, preventing either component from dominating the learning signal due to scale differences.

Rationale for Annealing. At the beginning of training, the Hamiltonian world model H_ϕ is randomly initialized. Consequently, the symmetry probing reward $r_{sym,t} \propto |H_\phi(Z_{t+1}) - H_\phi(Z_t)|$ is extremely noisy and does not yet reflect true physical information gain. Relying solely on it initially can lead to degenerate exploration behaviors. To bootstrap the learning process, we initially rely on r_{RND}. We use a standard RND implementation that operates directly on pixel observations, providing a stable, model-agnostic diversity signal. This ensures broad coverage of the state space, collecting the initial data necessary to start training H_ϕ. As H_ϕ becomes more accurate, we linearly shift the exploration focus towards active symmetry probing.

Annealing Schedule. We employ a linear annealing schedule for the weight w_t:

$$w_t = \text{clip}\left(\frac{t}{T_{anneal}}, 0, 1\right) \tag{18}$$

The annealing duration T_{anneal} is a task-dependent hyperparameter, generally set longer for environments with more complex dynamics that require more initial data to stabilize H_ϕ. Table 6 lists the specific values used.

Table 6. Annealing duration T_{anneal} for different environments. We set longer annealing phases for environments with complex dynamics or high-dimensional state spaces (*e.g.*, Humanoid, Quadruped) to allow sufficient exploration via RND before switching to symmetry probing.

Environment	T_{anneal} **(Steps)**
GymFetch Manipulation	
FetchReach	2×10^5
FetchPush	5×10^5
FetchSlide	1×10^6
DeepMind Control Suite	
Acrobot Swingup	2×10^5
Reacher Hard	5×10^5
Hopper Hop	5×10^5
Walker Walk / Run	5×10^5
Cheetah Run	5×10^5
Quadruped Run / Escape	1×10^6
Humanoid Walk	1×10^6

7. Implementation Details

7.1. World Model Loss Functions

Our full world model objective (Eq. 5 in the main paper) is:

$$\mathcal{L}_{\text{total}}(\phi) = \sum_{t=1}^{T} \Big(\mathcal{L}_{\text{pred}}(\phi) + \beta_{\text{dyn}} \mathcal{L}_{\text{dyn}}(\phi) + \beta_{\text{rep}} \mathcal{L}_{\text{rep}}(\phi) + \gamma \mathcal{L}_{\text{vr}}(\phi) \Big)$$

As we introduce a novel Hamiltonian dynamics prior, p_ϕ, the standard KL divergence losses from [17] must be adapted. We define these components as follows:

Prediction Loss ($\mathcal{L}_{pred}$) This is the standard reconstruction loss, formulated as the log-likelihood of the observation x_t given the latent state Z_t and the recurrent state h_t:

$$\mathcal{L}_{pred} = -\mathbb{E}_{q_\phi(Z_t|x_t,h_t)} \left[\log p_\phi(x_t|Z_t, h_t)\right]$$

where $q_\phi(Z_t|x_t, h_t)$ is the posterior from the object-centric encoder, and $p_\phi(x_t|Z_t, h_t)$ is the decoder.

Dynamics and Representation Losses ($\mathcal{L}_{dyn}, \mathcal{L}_{rep}$) These are the two components of the KL divergence that force the latent state Z_t inferred from the image (the posterior q_ϕ) to match the state predicted by the Hamiltonian

dynamics (the prior p_ϕ). Our key modification is that our prior p_ϕ is Markovian on Z and does not depend on the recurrent state h_t.

The posterior q_ϕ is inferred from the current image x_t and history h_t. The prior p_ϕ is predicted from the previous latent state Z_{t-1} and action a_{t-1} using our Hamiltonian integrator (Eq. 4).

Following [17], we split the KL divergence and apply stop-gradients (sg) to create two distinct losses that train the encoder and the prior separately:

$$\mathcal{L}_{dyn} = KL\left[\text{sg}(q_\phi(Z_t|x_t, h_t)) \parallel p_\phi(Z_t|Z_{t-1}, a_{t-1})\right]$$

This loss trains the Hamiltonian prior p_ϕ (*i.e.*, H_ϕ) to correctly predict the latent state Z_t that the encoder inferred from the image.

$$\mathcal{L}_{rep} = KL\left[q_\phi(Z_t|x_t, h_t) \parallel \text{sg}(p_\phi(Z_t|Z_{t-1}, a_{t-1}))\right]$$

This loss trains the encoder q_ϕ to produce latent states Z_t that are consistent with the predictions of the (fixed) dynamics prior. Following DreamerV3 [17], we do not use fixed static weights for $\mathcal{L}_{dyn}$ and $\mathcal{L}_{rep}$. Instead, we employ KL Balancing to encourage the posterior to maintain sufficient entropy while pulling the prior towards it. The objective is computed as:

$$\mathcal{L}_{KL} = \beta_{dyn} KL[\text{sg}(q_\phi)||p_\phi] + \beta_{rep} KL[q_\phi||\text{sg}(p_\phi)] \quad (19)$$

We set the coefficients to $\beta_{dyn} = 0.5$ and $\beta_{rep} = 0.1$. This configuration allows the prior to learn the dynamics rapidly without collapsing the posterior's representation capacity, ensuring the Hamiltonian structure H_ϕ captures rich physical features from the start.

7.2. Network Architectures

7.2.1. Object-Centric Encoder (SAVi)

We utilize the Slot Attention for Video (SAVi) [24] architecture to decompose the visual scene into object-centric latent slots $Z_t = \{z_t^1, \ldots, z_t^N\}$.

- **Backbone:** A ResNet-18 (truncated) extracts a feature map of size $H' \times W' \times D_{enc}$ from the 64×64 input image. We add sinusoidal position embeddings to the feature map.
- **Slot Attention:** We use $N = 6$ slots for all environments (sufficient for the robot, objects, and background). The attention mechanism runs for 3 iterations per time step.
- **Slot Dimensions:** Each slot has a dimension of $D_{slot} = 128$.
- **Latent Projection:** The output of the slot attention is projected via an MLP to parameterize the mean and variance of the posterior Gaussian q_ϕ. Crucially, the slot dimension is split evenly into generalized coordinates and momenta: $z^i = [q^i, p^i]$, where $q, p \in \mathbb{R}^{64}$.

Table 7. Hyperparameters used for DreamSAC training.

Parameter	Value
Training Config	
Batch Size	50
Sequence Length (T)	64
Total Training Steps	2×10^6 (Pretrain) + 5×10^5 (Adapt)
Optimizer	AdamW
Learning Rate (World Model)	1×10^{-4}
Learning Rate (Actor-Critic)	3×10^{-5}
Grad Clip Norm	100.0
Loss Weights	
Prediction Loss Scale	1.0
Dynamics Loss Scale (β_{dyn})	0.5
Representation Loss Scale (β_{rep})	0.1
Viewpoint-Robustness Scale (γ)	1.0
InfoNCE Temperature (τ)	0.07
Exploration & Physics	
Intrinsic Reward Scale	1.0
Action Smoothness (λ_s)	0.01
Annealing Steps (T_{anneal})	See Table 6
Physics Integration Step (Δt)	0.1

7.2.2. Invariant Hamiltonian (Lie Transformer)

The internal Hamiltonian $H_\phi(Z_t)$ is parameterized as a Lie Transformer [20] to enforce $SE(3)$ invariance by construction.

- **Input:** The set of object slots Z_t is treated as a set of particles.
- **Group Structure:** We lift the inputs to the Lie Algebra of $SE(3)$.
- **Architecture:** The network consists of $L = 4$ Lie Self-Attention layers with 4 attention heads each. The embedding dimension is 128.
- **Output:** A final invariant pooling layer aggregates the features followed by an MLP to output a single scalar value: the Hamiltonian $\mathcal{H} \in \mathbb{R}$.

7.2.3. Input Matrix Network (g_ϕ)

The input matrix $g_\phi(q_t)$ determines how the action a_t influences the system's momentum. It is parameterized as a 3-layer MLP with 128 hidden units and ELU activations.

- **Input:** The generalized coordinates $q_t \in \mathbb{R}^{64}$ extracted from the object slots.
- **Output:** A matrix of dimension $64 \times |A|$, where $|A|$ is the action dimension. This matrix is reshaped to perform the element-wise product with the action vector in Eq. 11.
- **Initialization:** The final layer weights are initialized with a small scale (10^{-3}) to ensure the initial dynamics are close to autonomous evolution, stabilizing the early training of the Hamiltonian prior.

7.2.4. Decoder and Actor-Critic

- **Decoder:** A standard transposed convolutional network with 4 layers (kernels: 4×4, stride: 2) and ELU activations. It receives the concatenated slots and recurrent state.

- **Actor & Critic:** Both are MLPs with 4 hidden layers of 256 units and ELU activations. The Actor outputs a $\tanh$ Gaussian policy; the Critic outputs a scalar value estimate.

7.3. Hyperparameters

Table 7 summarizes the hyperparameters used across our experiments. We adhered closely to the default DreamerV3 parameters where possible to isolate the gains from our Hamiltonian contribution.

7.4. Training Pseudocode

The training process is distinctively split into an unsupervised curiosity-driven phase and a task-driven adaptation phase.

Algorithm 1 DreamSAC Training Procedure

Params: ϕ (World Model), θ (Actor), ψ (Critic)
Init: Buffer $\mathcal{B}$, random weights
▷ **Phase 1: Unsup. Pretraining**
while $t < T_{pretrain}$ **do**
  $w_t \leftarrow \min(t/T_{anneal}, 1)$
  Interact: Execute $a_t \sim \pi_\theta(h_t, Z_t)$, observe x_{t+1}
  $r_{int} \leftarrow (1 - w_t) r_{\text{RND}} + w_t |\Delta H_\phi|$
  Add (x_t, a_t, r_{int}) to $\mathcal{B}$
  Train: Sample batch $B \sim \mathcal{B}$
  Optimize ϕ via $\mathcal{L}_{total}$ (Eq. 5, incl. $\mathcal{L}_{vr}$)
  Imagine $\{Z_\tau\}$ via Ham. Integrator
  Update π_θ, v_ψ on imagined data
end while
▷ **Phase 2: Adaptation**
for task with reward r_{ext} **do**
  Opt A (ID):
  Freeze p_ϕ, q_ϕ
  Re-init & train $\pi_{\theta'}$ via r_{ext}
  Opt B (OOD):
  Freeze q_ϕ
  Finetune H_ϕ (LR 10^{-5}); Train $\pi_{\theta'}$
end for

8. Experimental Setup Details

8.1. Baseline Configurations and Oracle Definitions

To provide a comprehensive evaluation, we compare against different training protocols. It is crucial to distinguish between the extrapolation capabilities and the theoretical upper bound of the tasks:

- **DreamerV3+Policy (Oracle / Reference):** In Table 2 (Main Paper), the entries for "DreamerV3+Policy" represent the **Oracle performance**. For these specific entries, the model was trained *directly* on the target OOD environment (*e.g.*, the specific Unseen View or Unseen Gravity configuration) from scratch. This serves as an empirical **upper bound**, quantifying the maximum achievable reward if the agent were perfectly adapted to the target domain. The significant gap between this Oracle score (*e.g.*, $\sim$ 954 on Reacher Unseen View) and the adaptation scores (*e.g.*, $\sim$ 321 for DreamSAC) highlights the extreme difficulty of the zero-shot/few-shot extrapolation task compared to standard i.i.d. training.
- **DreamerV3+RND & DreamSAC (OOD Extrapolation):** In contrast, all other baselines and our method follow the strict OOD protocols defined below (Single-Parameter Shift or Distribution Extrapolation), where the agent has *never* seen the specific target configuration during the pre-training phase.

8.2. Environment Configurations

To rigorously test extrapolative generalization, we constructed specific OOD variants of standard DeepMind Control Suite and GymFetch tasks. We categorize these into Structural Generalization and Parametric Generalization tasks.

Structural Generalization (Visual & Configuration). These tasks test the model's ability to handle unseen visual perspectives and spatial configurations. For the standard training distribution (used by extrapolation models), we fix the camera azimuth at $\phi = 0°$ and elevation at $\theta = 15°$ with a single dynamic object. In the Unseen View OOD setting, we sample the camera azimuth uniformly from $[0°, 90°]$ to test viewpoint invariance. Note on Baselines: It is important to distinguish that the DreamerV3+Policy baseline reported in Table 2 serves as an Oracle: it was trained directly on the target views to establish an empirical upper bound. In contrast, DreamSAC and other baselines are evaluated in a strict zero-shot manner, having never encountered these viewing angles during pre-training. For the Unseen Object task, we increase the number of dynamic objects from 1 to 3, testing the slot attention's ability to instantiate new slots for physics interactions. Crucially, for the Unseen Goal task (in FetchReach and Reacher), we sample target positions that lie strictly outside the distance range encountered during training (*e.g.*, targets are generated in the outer 20% of the workspace radius, whereas training targets are confined to the inner 50%). This requires the agent to spatially extrapolate its motion planning policy to reach novel coordinates never visited before.

Parametric Generalization (Physical Laws). These tasks test the Hamiltonian model's ability to adapt to changes in the fundamental constants of the environment. We employ two distinct evaluation protocols to rigorously test different aspects of generalization. (1) For the Unseen Gravity, Unseen Friction, and GymFetch Heavy Block

tasks, we utilize a single-parameter shift (zero-shot extrapolation) protocol. Here, models are trained on a fixed standard configuration (*e.g.*, standard gravity $g = -9.81m/s^2$ or block mass $2kg$) and evaluated on a significantly shifted configuration (*e.g.*, gravity scaled by $1.5\times$, friction by $2.0\times$, or mass to $10kg$). This tests the model's ability to extrapolate physical laws from a single data point without prior exposure to parameter variations. (2) Crucially different is the protocol for the Walker and Cheetah Unseen Dist. tasks, where we adopt a distribution extrapolation protocol. Unlike the single-parameter shift, this task employs a rigorous train/test split strategy to evaluate robustness against compound domain shifts. We define a broad range of physical parameters (simultaneously perturbing torso mass, joint damping, and contact friction) and sample training environments exclusively from the lower 80% of this range (D_{train}). Evaluation is performed solely on the held-out upper 20% (D_{test}). This setup implies that all models—including the DreamerV3 baseline—are trained on the randomized D_{train} distribution, effectively making the baseline a domain randomization (DR) agent. Consequently, DreamSAC's superior performance on this task demonstrates that it has not merely memorized the training distribution (interpolation) but has learned the underlying functional form of the dynamics to generalize to unseen parameter ranges (extrapolation).

9. Additional Experimental Results

9.1. Analysis of Learned Physical Representations

A core hypothesis of DreamSAC is that the split latent representation $Z_t = (q_t, p_t)$ learns to encode underlying physical laws and symmetries, rather than mere visual statistics. We validate this through the qualitative analyses presented in Figure 3 of the main paper.

We first verify the conservation laws by analyzing the evolution of the learned internal Hamiltonian H_ϕ during a rollout. As shown in Figure 3b (Main Paper), the value of H_ϕ remains nearly constant (red dashed line) during a *zero-action* rollout. This empirically confirms that our model has successfully discovered the environment's underlying physical invariant (energy conservation) and satisfies the autonomous Hamiltonian dynamics condition $\dot{H} \approx 0$ without direct supervision. In contrast, during random action rollouts, H_ϕ fluctuates, reflecting the work done by external forces.

Furthermore, we investigate the physics-aware latent structure by visualizing the high-dimensional latent states $(\boldsymbol{q}, \boldsymbol{p})$ using t-SNE. Figure 3d (Main Paper) compares the latent distributions of the pre-trained model against the model fine-tuned on downstream tasks. For In-Distribution (ID) tasks where physical properties match the training set, the representations of the fine-tuned model and pre-trained model remain heavily mixed, indicating that the pre-trained physics prior is directly applicable. Conversely, for Out-of-Distribution (OOD) tasks with novel physical properties (*e.g.*, modified friction or gravity), the fine-tuned states form distinct clusters that clearly separate from the pre-training distribution. This separation demonstrates that the encoder q_ϕ has learned a physics-aware topology capable of distinguishing between familiar and novel dynamics based on interaction.

9.2. Extended Baseline Comparisons

We compared Symmetry Exploration against other intrinsic motivation baselines on the GymFetch **Heavy Block** OOD task, which requires precise physical adaptation.

Implementation of Baselines. To ensure a fair comparison and isolate the efficacy of our Symmetry Exploration strategy, we did not use the original pixel-based implementations of ICM [38] or Plan2Explore [45]. Instead, we re-implemented both baselines **on top of the exact same DreamerV3 backbone** (with Hamiltonian) used by DreamSAC. Specifically, they operate on the same latent features Z_t, share the same hyperparameters for the world model training, and use the same SAVi encoder. This guarantees that the performance gains reported below are solely driven by our physics-aware curiosity mechanism, rather than differences in the underlying generative model capacity.

- **ICM (Prediction Error):** Focuses on parts of the state space that are hard to predict. We found this often led the agent to get stuck in "stochastic traps" (*e.g.*, white noise), failing to learn the precise dynamics required to manipulate the heavy object.
- **Plan2Explore:** Maximizes information gain about the dynamics. While effective, it requires training an ensemble of dynamics models, which is computationally heavier than our single Hamiltonian method. Furthermore, it lacks the specific incentive to probe energy boundaries.

As shown in Table 8, DreamSAC significantly outperforms both baselines on the DMCS Walker-walk task with unseen gravity. This suggests that seeking energy changes ($r_{sym} \approx |\Delta H|$) is a more efficient heuristic for discovering physical parameters (like mass) than generic information gain, enabling the agent to adapt to the heavier object dynamics.

9.3. Hyperparameter Sensitivity Analysis

We analyze the impact of the action smoothness regularization weight λ_s (Eq. 6). A potential critique is that performance gains might stem solely from action smoothing. However, our ablation on Reacher-Hard (Table 9) refutes this.

The intrinsic reward $r_{sym} \approx |\Delta H|$ encourages the agent to maximize energy changes (work). Without regularization

Table 8. Mean Reward on the DMCS Walker-walk Unseen Gravity OOD task (Gravity $1\times \rightarrow 1.5\times$). Comparison ensures identical backbone architectures (with Hamiltonian).

Method	Mean Reward
DreamerV3+ICM	469.72
DreamerV3+Plan2Explore	379.28
DreamSAC (Ours)	**499.91**

($\lambda_s = 0$), the agent can trivially maximize this via high-frequency "jitter," which generates large numerical ΔH but lacks physical meaningfulness (effective work). As shown in Table 9, setting $\lambda_s = 0$ results in a reward of **306.5**. While this is lower than our peak performance, it remains significantly effective (far exceeding random policies), confirming that the Hamiltonian exploration mechanism itself is the primary driver of learning.

Introducing $\lambda_s = 0.01$ filters out this "jitter noise," allowing r_{sym} to accurately reflect coherent physical work, boosting the reward to **321.90**. Conversely, excessive smoothing ($\lambda_s = 0.1$) overly restricts the agent's ability to manipulate the system, dropping performance to 152.4. Thus, λ_s acts as a necessary signal-to-noise filter for the physics-based reward, rather than a standalone performance hack.

Table 9. Sensitivity analysis of Action Smoothness λ_s on **Walker-walk**. Note that even with $\lambda_s = 0$, the model maintains decent performance, indicating that the Hamiltonian prior is robust. The smoothing parameter primarily serves to filter out high-frequency jitter that creates false energy deltas.

λ_s Value	Mean Reward	Behavior
0.0 (No Reg.)	967.43	Jitter
0.01 (Ours)	**996.50**	Coherent
0.1 (High Reg.)	921.66	Over-smooth

10. Impact of Integrator Choice during Inference

A core design choice in DreamSAC is the *Dual Integration Strategy*, which employs a standard explicit Euler integrator during training for gradient stability while switching to a Symplectic Leapfrog integrator during imagination and inference to better preserve physical invariants. To validate that this discrepancy does not degrade predictive performance and indeed improves physical consistency, we conducted a comparative experiment on the `Acrobot` task. We evaluated the pre-trained DreamSAC model using both Euler and Leapfrog integrators over a long prediction horizon ($H = 100$). Our evaluation relies on two key metrics: the **Long-term Prediction MSE** to measure visual dynamics accuracy, and the **Energy Drift (σ_H)**, defined as the standard deviation of the learned Hamiltonian value $H_\phi(z_t)$ over a zero-action rollout, where a lower value indicates better adherence to the conservation of energy law.

As shown in Table 10, the results demonstrate that while both integrators achieve comparable predictive MSE with Leapfrog being slightly superior (0.198 vs. 0.215), the Symplectic Leapfrog integrator significantly outperforms Euler in terms of energy conservation. Specifically, the Euler integrator suffers from numerical dissipation, leading to a high energy drift ($\sigma_H = 0.128$), whereas the Leapfrog integrator maintains a nearly constant Hamiltonian ($\sigma_H = 0.015$). This confirms that our dual strategy successfully combines training stability with the long-term physical plausibility required for robust planning.

Table 10. Comparison of Inference Integrators on Acrobot ($H = 100$). While MSE remains similar, the Symplectic Leapfrog integrator (Ours) drastically reduces Energy Drift, confirming its ability to enforce physical conservation laws that explicit Euler fails to maintain over long horizons.

Metric	Euler Inference	Leapfrog Inference (Ours)
Prediction MSE (↓)	0.215	**0.198**
Energy Drift (σ_H, ↓)	0.128	**0.015**

10.1. Computational Efficiency Analysis

A potential concern with Hamiltonian-based models is the computational overhead of the symplectic integrator, which requires evaluating gradients of the Hamiltonian during the forward pass. We provide a breakdown of the training and inference costs in Table 11, measured on a single NVIDIA A100 GPU.

While DreamSAC introduces a $\sim 35\%$ increase in training wall-clock time per step due to the dual integration strategy (Euler for world model updates, Leapfrog for imagination), this is offset by its superior **sample efficiency**. DreamSAC typically converges to higher rewards with significantly fewer environment interaction steps compared to the baselines, making it more efficient in terms of total time-to-convergence for complex physical tasks.

Table 11. Computational cost comparison (normalized relative to DreamerV3).

Method	Training Time / Step	GPU Memory Usage
DreamerV3	$1.00\times$	$1.00\times$
DreamSAC (Ours)	$1.35\times$	$1.12\times$

11. Limitations and Future Work

While DreamSAC demonstrates robust extrapolative capabilities via symmetry discovery, an analysis through the

lenses of theoretical modeling, algorithmic stability, and computational scalability reveals key areas for future development.

Theoretical Boundaries of Conservative Modeling. Our framework currently models the world as a controlled Hamiltonian system, presupposing that the underlying dynamics are fundamentally conservative with external control. This assumption faces challenges in highly dissipative environments—such as movement through viscous fluids or soft-body deformations with internal friction—where energy is continuously dissipated. In our current formulation, the model must implicitly "overload" the control term $g(q)a_t$ to mimic friction as a negative force, effectively conflating system dynamics with actuation. A promising direction is to extend this formulation to the **Port-Hamiltonian System (PHS)** framework, which explicitly separates energy storage, energy dissipation (via Rayleigh functions), and external ports, offering a theoretically unified view of open physical systems. Furthermore, complex robotic interactions often involve non-holonomic constraints (*e.g.*, a rolling wheel preventing sideways sliding) which are difficult to capture purely via the potential energy shaping used in our current approach.

Numerical Stiffness and Discretization Gaps. Modeling hard contacts as stiff potential barriers within H_ϕ introduces significant numerical stiffness into the ordinary differential equations. During inference, if the symplectic integrator's time step Δt is not sufficiently infinitesimal, high-velocity impacts can lead to numerical instability or non-physical energy spikes (tunneling effects). Future iterations could integrate Differentiable Linear Complementarity Problems (LCP) or learned jump maps directly into the integration step to handle instantaneous momentum updates without requiring computationally expensive small time steps. Additionally, as discussed in Sec. 6.2, our dual integration strategy (training on Euler, imagining on Leapfrog) introduces a discretization gap: the vector field is optimized for one numerical scheme but evaluated on another. Developing stable methods for end-to-end symplectic training (*e.g.*, via implicit differentiation or adjoint sensitivity methods) remains a critical open challenge for deep generative models.

Computational and Representational Scalability. The reliance on a symplectic integrator imposes a computational overhead, requiring two evaluations of the Hamiltonian gradients per time step. This results in an inference cost approximately $1.5\times$ higher than standard GRU-based RSSMs, creating non-negligible latency for high-frequency real-time control (> 30 Hz). Future work could explore model distillation techniques to compress the learned Hamiltonian dynamics into faster, explicit predictors for deployment. Finally, our SAVi-based encoder assumes the scene decomposes into a fixed number of discrete slots, which limits applicability to unstructured environments containing liquids, cloth, or granular media. Integrating Grid-based Neural Physics or hierarchical representations with our Hamiltonian prior could extend extrapolative generalization to these more complex, physically unstructured domains.