LEARNING SUCCESSOR-MEASURE WORLD MODELS FOR ZERO-SHOT CONTROL FROM PIXELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models typically support decision-making by predicting future observations or latent states, with behavior adapted to a task through planning or policy optimization. For zero-shot control, we instead ask whether a world model can directly organize the long-horizon visitation induced by different behaviors, so that a new reward can select among them without additional policy optimization. We propose SWM, a successor-measure world model for zero-shot visual control. Rather than learning step-by-step predictive dynamics, SWM learns policy-conditioned future occupancy in a compact pretrained visual latent space. SWM uses a frozen DINO encoder as a task-agnostic visual basis and trains a lightweight adaptor, together with forward and backward successor-measure representations, to organize visual observations into a space where future occupancy can be factorized and reused across tasks. At test time, a new task specified by reward-labeled observations is projected onto the learned backward basis, directly selecting a policy from the pretrained behavior family without additional policy optimization or online interaction. Experiments on pixel-based Control tasks show that SWM consistently improves over reward-free predictive world-model baselines and achieves higher mean return than CNN-FB on all sixteen evaluated tasks. Importantly, the learned latent space preserves control-relevant pose information and retains robustness under visual perturbations, suggesting that frozen DINO features can serve as a reusable basis for occupancy-based world modeling when aligned through successor-measure factorization.

1 INTRODUCTION

Modeling and planning over future dynamics is central to decision-making (Sutton, 1991). World models provide a powerful mechanism for this purpose by learning predictive dynamics, either in pixel space or in a learned latent space, and optimizing behavior through planning or policy optimization inside the learned model (Ha & Schmidhuber, 2018; Hafner et al., 2020; 2021; Hansen et al., 2022a; Hafner et al., 2023; Hansen et al., 2024). While general, they may allocate capacity to redundant visual details that are not useful for control, and test-time planning may exploit errors in the learned dynamics. However, in many control problems, the most useful notion of the future is not a visually accurate prediction of all future frames or states but rather which states an agent is likely to visit and whether those states are useful, reachable, and relevant to a downstream task. This distinction is especially important in visual control, where an agent must learn control-relevant representations from image trajectories while retaining the ability to adapt to new tasks without additional policy training or online interaction.

Successor representations and successor measures provide a principled foundation for this view by representing the discounted future occupancy under a policy (Dayan, 1993b). At a high level, they model where a behavior is likely to go over the long horizon, and separate this policy-dependent future visitation from task-dependent rewards. As a result, once the successor measure is learned, new tasks can be handled by projection rather than by training a new policy from scratch. If such an occupancy representation can be learned from visual trajectories, it would provide a direct route to control-relevant and reusable world models: image observations could be mapped to a latent space where policies are evaluated by the futures they visit, and new visual tasks could be specified through reward-labeled observations without additional policy optimization. This brings successor-measure learning into the domain of visual world models, but with a different modeling target:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

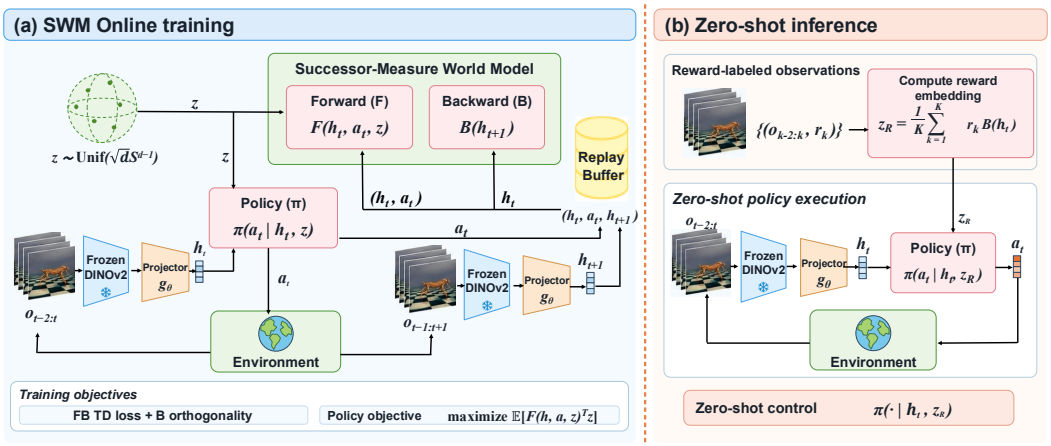

Figure 1: **SWM.** A frozen DINOv2 encoder and a lightweight projector g_θ map observations into a latent space $\mathcal{H}$ on which forward–backward successor-measure factorization is learned.

rather than predicting future pixels or latent states step by step, the model can learn a latent space in which policy-conditioned future occupancy can be reused and aligned with downstream control. Existing pixel-based FB implementations leave open whether a successor-measure world model can instead be built on a reusable pretrained visual representation, while retaining robustness to nuisance visual variation without relearning perception in every target domain. Realizing this requires more than preserving visual information: the representation must support factorized policy-conditioned occupancy and downstream reward projection.

In this work, we propose **S**uccessor-measure **W**orld **M**odel (SWM), a successor-measure world model that builds on pretrained visual embeddings to learn reusable, control-relevant future occupancy from image trajectories. SWM occupies a different design point between predictive model-based RL and direct visual planning: instead of using the model as a simulator for imagined policy learning or test-time action optimization, it learns an occupancy model together with a family of latent-conditioned policies, as shown in Figure 1. Given image observations, SWM builds on frozen DINO features (Oquab et al., 2023) as a task-agnostic visual basis. Using such features for successor-measure control introduces two distinct challenges. First, a single image does not directly expose motion, motivating short-term temporal context. Second, even with temporal context, a generic pretrained representation need not be readily usable by the FB objective; indeed, we find that applying FB directly to the same stacked frozen DINO features leads to substantially weaker zero-shot control. SWM addresses these challenges separately by aggregating DINO features across recent observations and learning a lightweight projector that adapts the frozen representation for successor-measure learning. In this latent space, SWM uses the forward–backward successor-measure objective (Touati & Ollivier, 2021a) to factorize policy-conditioned future occupancy: the forward representation encodes the futures associated with a candidate behavior, while the backward representation provides a basis for projecting new rewards. Unlike model-based RL with predictive world models (Ha & Schmidhuber, 2018; Hafner et al., 2020) and test-time MPC or CEM planning (Hansen et al., 2022a; Hansen et al.; Zhou et al., 2024), SWM does not train a new policy through imagined rollouts or optimize actions against predicted visual futures. Instead, it performs zero-shot adaptation by reward projection over a learned future-occupancy representation.

We evaluate SWM on pixel-based control tasks in the DeepMind Control (DMC) Suite Tassa et al. (2018): Walker, Quadruped, Cheetah, and Jaco Arm Laskin et al. (2021). SWM improves both control performance and target-domain training efficiency: it achieves the highest mean zero-shot return on all sixteen tasks while requiring at least $5\times$ less target-domain pretraining compute than its closest competitor CNN-FB, with the forward and backward representations and test-time task inference all operating on pixels alone. It also outperforms predictive world-model baselines adapted to our reward-free pretraining and no-finetuning downstream setting. The resulting policy also retains a strong advantage under most visual perturbations, showing that the frozen visual prior remains useful after FB-specific adaptation. More importantly, controlled ablations reveal that raw DINO

features can retain substantial decodable pose and motion information while still failing to support effective FB control, separating generic state recoverability from successor-measure control utility.

Our contribution is to recast visual world modeling for zero-shot control as successor-measure learning in a pretrained visual feature space. Concretely, SWM (i) constructs a visual successor-measure control space from frozen DINO features by separating temporal context from lightweight FB-driven adaptation, (ii) jointly learns a forward–backward occupancy factorization and a latent-conditioned policy family directly in this learned visual space, and (iii) uses the same learned space to project reward-labeled observations into zero-shot policies, without test-time planning or policy optimization. Our analysis further shows that this visual realization is non-trivial: raw pretrained features can retain substantial control-relevant information while remaining ineffective for FB control.

2 Related Work

Visual representations and latent-space world models. Visual control has increasingly moved from raw-pixel prediction to latent representations learned jointly with the policy Hafner et al. (2020; 2024; 2023); Hansen et al.; 2022b), pretrained for control Xiao et al. (2022); Majumdar et al. (2024); Nair et al. (2022), or, more recently, derived from frozen visual foundation models Oquab et al. (2023); Zhou et al. (2024); Destrade et al. (2025); Maes et al. (2026). Most latent-space world models remain predictive: Dreamer-style methods Hafner et al. (2020; 2024; 2023) improve policies through imagined rollouts, TD-MPC-style methods Nicklas Hansen (2026); Hansen et al.; 2022b) plan with learned latent dynamics, and DINO-WM Zhou et al. (2025) and JEPA Destrade et al. (2025) move the prediction target into the latent space of pretrained encoders. Thus, even when prediction happens in a pretrained feature space, the learned object is still a model of future observations or latent features, rather than a representation of future visitation that can be directly paired with a new reward. In contrast, SWM learns a representation and policy family jointly, so that new rewards select policies by projection rather than by test-time planning.

Generative world models for zero-shot control. A second route scales generative world models, using pretrained video or video-action generators for zero-shot behavior Ye et al. (2026); Chen et al. (2025); Liao et al. (2025). This is complementary to our setting but requires large-scale video pretraining and generative inference. We instead ask whether reward-conditioned zero-shot control can be obtained with a lightweight successor-measure model on top of a frozen visual backbone.

Successor measures and behavior foundation models. Successor measures and successor features represent the discounted future occupancy induced by policies Dayan (1993a); Blier et al. (2021); Borsa et al. (2018); Barreto et al. (2018); Jajoo et al. (2026). The forward–backward framework factorizes this occupancy with a z-indexed policy family Touati & Ollivier (2021a); Touati et al. (2023): a new reward is projected onto the backward representation and directly selects a pretrained policy, without downstream policy optimization or model-predictive planning. Recent behavior foundation models based on this idea Tirinzoni et al. (2025); Pirotta et al. (2024); Agarwal et al. (2024); Sikchi et al. (2025b); Li et al. (2025); Sikchi et al. (2025a) primarily operate on Markov states, privileged features, or proprioception. Visual zero-shot RL has also been studied in the offline setting. For example, TD-JEPA Bagatella et al. (2025) learns latent predictive representations from fixed reward-free datasets. In contrast, we study online reward-free visual pretraining, where the agent collects its own experience while learning the reusable policy family. Within this setting, pixel-based FB can be trained with a CNN encoder from scratch Touati et al. (2023), yielding a visual representation specialized to the training environment. SWM instead uses frozen DINO features as the visual basis and learns a lightweight FB-trained projector on top of them.

3 Preliminaries

In this section, we introduce preliminaries and terminology used in later sections. We build on prior work on successor measures, behavior optimization from successor measures, and compact visual representations. For conciseness, we discuss only the components most directly relevant to SWM.

Successor Measure. Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mu, \gamma)$ be a reward-free discounted Markov decision process (MDP) characterizing a sequential decision-making problem. $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P(\mathrm{d}s' \mid s, a)$ the transition kernel describing the stochastic next state obtained by taking action a in state s, μ a distribution over initial states, and $\gamma \in [0, 1)$ the discount factor. We write

$\Pr(\cdot \mid s_0, a_0, \pi)$ and $\mathbb{E}[\cdot \mid s_0, a_0, \pi]$ for the probability and expectation operators over state-action sequences $(s_t, a_t)_{t\geq 0}$ starting from (s_0, a_0) under a policy π, with $s_t \sim P(\cdot \mid s_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot \mid s_t)$. For each policy π and each $(s_0, a_0) \in \mathcal{S} \times \mathcal{A}$, the *successor measure* $M^\pi(s_0, a_0, \cdot)$ over $\mathcal{S}$ captures the cumulative discounted visitation at each future state can be represented as:

$$M^\pi(s_0, a_0, X) := \sum_{t\geq 0} \gamma^t \Pr(s_{t+1} \in X \mid s_0, a_0, \pi), \qquad \forall X \subseteq \mathcal{S}. \tag{1}$$

For the measurable subset $X \subseteq \mathcal{S}$, the successor measure $M^\pi(s_0, a_0, X)$ represents the discounted expected number of future visits to X, and satisfies a measure-valued Bellman Eq. Blier et al. (2021):

$$M^\pi(s, a, X) = P(X \mid s, a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a),\, a'\sim\pi(\cdot|s')}[M^\pi(s', a', X)], \qquad \forall X \subseteq \mathcal{S}. \tag{2}$$

We define $\rho^\pi(X) := (1-\gamma)\, \mathbb{E}_{s\sim\mu,\, a\sim\pi(\cdot|s)}[M^\pi(s, a, X)]$ as the discounted stationary distribution of π. Given M^π, the Q^π for any task-specific reward function $r : \mathcal{S} \to R$ can be written as

$$Q^\pi_r(s, a) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r(s_{t+1}) \,\middle|\, s_0 = s,\ a_0 = a,\ \pi\right] = \int_{s'\in\mathcal{S}} r(s')\, M^\pi(s, a, \mathrm{d}s'). \tag{3}$$

Eq. 3 cleanly separates Q^π_r into (1) the successor measure, which captures policy-induced occupancy, and (2) the task-specific reward. As a consequence, learning M^π enables zero-shot policy evaluation: Q^π_r for any reward r can be evaluated without further training.

The forward-backward (FB) framework. The FB representation provides a low-rank decomposition of the successor measure, $M^\pi \approx \int_{s^+\in X} F^\pi(s, a)^\top B(s^+)\, \rho(\mathrm{d}s^+)$, where ρ is a state distribution and $F^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and $B : \mathcal{S} \to \mathbb{R}^d$ are the *forward* and *backward* embeddings. This decomposition yields $Q^\pi_r(s, a) = F^\pi(s, a)^\top z$, where $z = \mathbb{E}_{s\sim\rho}[B(s)\, r(s)]$ is the task vector that projects r onto the backward basis. Touati & Ollivier (2021b) extend this approach to a family of policies, with both F and π indexed by the same z. This yields the following unsupervised pretraining objective:

$$\begin{cases} M^{\pi_z}(s, a, X) \approx \displaystyle\int_{s^+\in X} F(s, a, z)^\top B(s^+)\, \rho(\mathrm{d}s^+), & \forall s \in \mathcal{S},\ a \in \mathcal{A},\ X \subseteq \mathcal{S},\ z \in \mathcal{Z}, \\ \pi(s, z) = \arg\max_a F(s, a, z)^\top z, & \forall (s, a) \in \mathcal{S} \times \mathcal{A},\ z \in \mathcal{Z}. \end{cases} \tag{4}$$

The factorization in Eq. 4 can be learned without task rewards through a temporal-difference objective. Given a replay transition $(s, a, s') \sim \mathcal{D}$, an independently sampled query state $s^+ \sim \rho$, a policy index $z \sim p_{\mathcal{Z}}$, and $a' \sim \pi(\cdot \mid s', z)$, the standard FB objective is

$$\mathcal{L}_{\mathrm{FB}} = \mathbb{E}\Big[\big(F(s, a, z)^\top B(s^+) - \gamma\, \bar{F}(s', a', z)^\top \bar{B}(s^+)\big)^2\Big] \;-\; 2\, \mathbb{E}\big[F(s, a, z)^\top B(s')\big] + \lambda_B\, \mathcal{L}_{\mathrm{orth}}, \tag{5}$$

where $\bar{F}$ and $\bar{B}$ are target networks and $\mathcal{L}_{\mathrm{orth}}$ regularizes the backward representation (Appendix A.3). The first term is the bootstrapped successor-measure TD objective, while the second anchors occupancy at the observed next state. For continuous control, Eq. 4 is implemented with a conditional actor trained jointly with the FB representations:

$$\mathcal{L}_\pi = -\mathbb{E}_{\substack{s\sim\rho,\, z\sim p_{\mathcal{Z}}, \\ a\sim\pi(\cdot|s,z)}} \big[F(s, a, z)^\top z\big]. \tag{6}$$

Eq. 4 also clarifies what a visual representation must support: it must make discounted future occupancy factorizable and it must provide a coordinate system in which unseen rewards can be represented. While the original FB formulation operates on a Markov state space $\mathcal{S}$, visual control requires the same objective to be extracted from raw observations.

4 METHOD

Overview and problem formulation. We consider online reward-free pretraining for zero-shot visual control. During pretraining, the agent interacts with the environment from image observations and learns a family of behaviors without access to downstream task rewards. At test time, a new task is specified only through reward-labeled replay observations, and all learned parameters remain frozen. SWM builds this behavior family on top of the frozen DINO. A short history of DINO features provides the visual state, a lightweight projector adapts this representation for successor-measure learning, and the FB objective learns policy-conditioned future occupancy together with a conditional policy family. Once pretraining is complete, a new reward is projected onto the learned backward representation to select a policy from this family, requiring neither policy fine-tuning nor test-time trajectory optimization. Fig. 1 summarizes the training and inference pipeline.

4.1 Successor-Measure Modeling in Visual Representation Space

The FB framework in Sec. 3 defines a reusable interface between future occupancy and downstream rewards. The remaining question for visual control is what representation this occupancy should be modeled over. For a policy π_z, its policy-conditioned future occupancy is approximated as

$$M^{\pi_z}(h, a, \mathcal{X}) \approx \int_{h^+ \in \mathcal{X}} F(h, a, z)^\top B(h^+)\, \rho_{\mathcal{H}}(\mathrm{d}h^+), \qquad \forall\, \mathcal{X} \subseteq \mathcal{H}, \tag{7}$$

where $\rho_{\mathcal{H}}$ denotes the replay distribution in the learned representation space. Consequently, for a downstream reward r defined on these represented observations,

$$Q_r^{\pi_z}(h, a) \approx F(h, a, z)^\top z_r, \qquad z_r = \mathbb{E}_{h^+ \sim \rho_{\mathcal{H}}}\left[r(h^+) B(h^+)\right]. \tag{8}$$

Thus, once an appropriate visual state h is available, the standard FB machinery already provides the desired reward-to-policy interface. The main representation-learning problem in SWM is instead to construct such a state from frozen visual features: it must expose the temporal information needed for control while remaining amenable to successor-measure learning.

4.2 SWM: From Frozen Visual Features to Successor-Measure Control

We start from a frozen DINO encoder rather than learning perception from scratch on each control domain. DINO is pretrained with large-scale self-supervised visual objectives and provides transferable perceptual features (Oquab et al., 2023); keeping the backbone frozen allows this visual prior to be reused without adapting the foundation model to the target environment. A single DINO feature, however, does not fully specify the dynamic state relevant to control. Two observations with nearly identical pose can correspond to different velocities and therefore induce different future trajectories. We therefore make short-term temporal information available before learning the successor representation. Each image is encoded independently by the frozen backbone, and the CLS features from the three most recent observations are concatenated and projected into the control space:

$$c_t = \mathrm{CLS}(\phi_{\mathrm{DINO}}(o_t)), \qquad \tilde{c}_t = [c_{t-2}, c_{t-1}, c_t], \qquad h_t = g_\theta(\tilde{c}_t) \in \mathcal{H}. \tag{9}$$

Therefore, the DINO backbone remains a per-frame frozen encoder, while temporal context is introduced only after feature extraction through stacking the last three frames.

Making temporal information available is only the first part of the problem. Generic pretrained features are not optimized for the FB objective: the information they contain need not be organized in a form that is readily usable for factorizing policy-conditioned future occupancy. Whether the stacked DINO features are already sufficient for FB control is therefore an empirical question. As shown in Table 2, applying FB directly to these raw features yields substantially weaker zero-shot control. SWM instead introduces a lightweight projector g_θ between the frozen DINO and the successor-measure model, allowing the representation to be adapted through the FB objective, where g_θ consists of LayerNorm followed by a linear projection from $\mathbb{R}^{3d_{\mathrm{DINO}}}$ to $\mathcal{H}$. The visual backbone remains fixed, so all control-specific representation learning is confined to this small projector.

The next question is what should shape this adaptation. Since the purpose of g_θ is to make the frozen representation usable for successor-measure modeling, we let the FB objective itself provide the representation-learning signal. Specifically, the standard FB loss in Eq. 5 jointly updates g_θ, the forward representation $F(h_t, a_t, z)$, and the backward representation $B(h^+)$, where h^+ denotes the future visual representation used to index successor occupancy (Appendix A.2). Both sides of the factorization therefore operate on the same learned visual control space $\mathcal{H}$. In contrast, the actor objective in Eq. 6 updates the conditional policy $\pi(\cdot \mid h_t, z)$ without propagating actor gradients into g_θ. The projector is consequently shaped by the successor-measure objective rather than by direct actor supervision. Full training objective is provided in Appendix A.

This coupling defines how the representation is learned, but reward-free pretraining must also expose the model to sufficiently diverse futures before any downstream task is known. SWM obtains this experience from the same latent-conditioned policy family that FB is learning. Policy indices $z \sim p_{\mathcal{Z}}$ are sampled uniformly in direction on the sphere of radius $\sqrt{d}$, and the corresponding policy $a_t \sim \pi(\cdot \mid h_t, z)$ interacts with the environment to occupy the replay buffer $\mathcal{D}$. A new index is initialized at episode reset and subsequently resampled according to the exploration schedule. Replay transitions are then used to update the FB representation and the actor under freshly sampled policy indices. In

this way, the policy family generates the behavioral diversity from which its own future-occupancy representation is learned. The FB updates use bootstrapped targets with exponential-moving-average copies $\bar{F}$ and $\bar{B}$ (Lillicrap et al., 2019). We do not maintain a separate target projector: target features are produced by the current projector and treated with stop-gradient in the target branch. Query observations required by the FB objective are sampled in-batch from replay. These choices affect the optimization of the factorization but leave the frozen DINO backbone unchanged throughout pretraining.

4.3 Zero-Shot Policy Inference

Reward-free pretraining associates each policy index z with a behavior and its corresponding future occupancy. A downstream reward can therefore identify a direction in this learned behavior space without changing either the visual representation or the policy family. Given K reward-labeled replay observations together with their temporal context, we compute

$$\hat{z}_R = \frac{1}{K}\sum_{k=1}^{K} r_k B(h_k), \qquad h_k = g_\theta(CLS(o_{k-2:k})), \qquad z_R = \sqrt{d}\,\frac{\hat{z}_R}{\max(\|\hat{z}_R\|_2, \epsilon)}. \tag{10}$$

The normalization places z_R on the same radius-$\sqrt{d}$ sphere used for policy indices during pretraining. The resulting vector directly selects a member of the pretrained policy family, and the agent executes the mean action of $\pi(\cdot \mid h_t, z_R)$ throughout evaluation. Zero-shot adaptation therefore consists only of projecting reward-labeled observations through the learned backward representation and using the resulting task vector to select behavior. No model parameter is updated, and no trajectory optimization or additional policy learning is performed at test time.

5 Experiments

Our evaluation focuses on online reward-free pretraining, where each method collects its own experience from pixels under the same environment interaction budget. Within this setting, we focus on three questions: **1)** Can a frozen visual foundation model be coupled with successor-measure learning to support zero-shot control directly from pixels? **2)** Does the resulting policy remain effective under test-time visual perturbations, indicating that the benefit of the pretrained representation is not restricted to the training rendering? **3)** Why is the learned projector necessary, and what control-relevant information remains accessible after projecting the frozen DINO features? Together, these experiments evaluate both the effectiveness of SWM and the role of representation adaptation in making frozen visual features usable for successor-measure control.

We evaluate SWM on sixteen pixel-based continuous-control tasks across Walker, Quadruped, and Cheetah from the DeepMind Control Suite (Tassa et al., 2018), and Jaco Arm (Laskin et al., 2021). During pretraining, all agents interact with the environment from pixels without access to downstream task rewards. At evaluation time, each task is specified by the same set of reward-labeled observations, and no method updates its policy, visual representation, or latent dynamics. Environment and implementation details are provided in Appendix B.

Baselines and variants. The comparison is designed to separate three questions. Dreamer-v3 and TD-MPC2 test whether reward-free predictive world models can support downstream control when their pretrained visual representation and latent dynamics are kept frozen. They are pretrained from the same reward-free pixel interaction budget as SWM; for each downstream task, we fit a small reward predictor using the same K reward-labeled observations available to SWM and perform model-predictive control in the frozen latent dynamics, see Appendix E.4. CNN-FB instead retains the FB objective but learns its visual encoder from scratch, thereby isolating the effect of introducing a frozen pretrained visual prior. Full baseline configurations and augmentations are provided in Appendix E.2. Finally, SWM-raw removes the learned projector g_θ and applies FB learning directly to frozen DINO features, allowing us to distinguish information already present in DINO from information that can be successfully used by the successor-measure objective.

5.1 Zero-Shot Control after Reward-Free Pixel Pretraining

Zero-shot control performance. We first evaluate whether the frozen DINO representation used by SWM can support effective successor-measure learning and zero-shot control after reward-free

Table 1: **Zero-shot control performance after reward-free pretraining.** Episode returns are averaged over 5 seeds, and higher is better. Orange shading denotes SWM, blue shading marks the strongest baseline for each task, and bold values indicate the best overall performance.

	Task	Dreamer	CNN-FB	TD-MPC2	SWM
Walker	stand	243.49 ± 16.17	407.89 ± 68.83	354.13 ± 67.12	$\mathbf{460.56 \pm 74.26}$
	flip	44.80 ± 3.03	189.54 ± 40.19	190.46 ± 42.18	$\mathbf{299.79 \pm 55.62}$
	walk	45.63 ± 2.77	337.66 ± 77.67	251.83 ± 37.64	$\mathbf{360.09 \pm 71.43}$
	run	38.84 ± 4.34	101.52 ± 13.73	69.24 ± 13.06	$\mathbf{125.33 \pm 17.70}$
	Average	93.19	259.15	216.42	**311.44**
Quadruped	jump	141.00 ± 55.01	497.48 ± 97.14	408.72 ± 96.40	$\mathbf{571.12 \pm 20.63}$
	walk	110.29 ± 4.09	350.53 ± 54.24	358.51 ± 77.29	$\mathbf{382.04 \pm 46.35}$
	stand	213.44 ± 3.05	732.50 ± 64.28	812.36 ± 104.83	$\mathbf{820.96 \pm 102.80}$
	run	87.19 ± 32.23	323.65 ± 21.53	256.78 ± 52.61	$\mathbf{458.42 \pm 50.89}$
	Average	137.98	476.04	459.09	**558.14**
Cheetah	walk backward	123.41 ± 86.29	142.71 ± 68.04	134.86 ± 29.52	$\mathbf{157.80 \pm 57.41}$
	run backward	29.16 ± 3.22	31.83 ± 16.00	24.37 ± 6.15	$\mathbf{38.40 \pm 13.13}$
	walk	319.61 ± 97.05	390.88 ± 135.74	221.64 ± 73.58	$\mathbf{586.04 \pm 117.44}$
	run	64.44 ± 20.41	70.19 ± 29.63	54.29 ± 21.46	$\mathbf{156.84 \pm 23.52}$
	Average	134.16	158.90	108.79	**234.77**
Jaco	reach top right	36.20 ± 6.17	55.32 ± 25.96	14.56 ± 10.40	$\mathbf{77.29 \pm 35.66}$
	reach top left	36.42 ± 7.51	61.42 ± 27.35	10.95 ± 18.22	$\mathbf{71.83 \pm 30.20}$
	reach bottom right	22.00 ± 9.66	35.85 ± 14.53	6.89 ± 2.65	$\mathbf{40.28 \pm 15.25}$
	reach bottom left	39.70 ± 11.75	44.88 ± 12.80	18.54 ± 7.50	$\mathbf{67.36 \pm 20.78}$
	Average	33.58	49.37	12.74	**64.19**

pretraining from pixels. Table 1 shows a consistent pattern. SWM outperforms CNN-FB on all sixteen downstream tasks. At the domain level, the average return improves by 20.2% on Walker, 17.2% on Quadruped, 47.7% on Cheetah, and 30.0% on Jaco.

Crucially, the advantage is not confined to static or slowly varying objectives. On Walker, SWM improves both *flip* and *run*, while the largest relative gains occur on Cheetah, where successful control depends strongly on fast body dynamics. The same direction of improvement also holds across all four Jaco reaching tasks. Thus, replacing a task-trained visual encoder with frozen pretrained DINO encoder does not merely preserve FB control: after adaptation through g_θ, it improves zero-shot performance across both locomotion and manipulation tasks in our evaluation. On the three DMC domains, SWM also exceeds Dreamer-v3 and TD-MPC2 in domain-average zero-shot return. These results indicate that frozen DINO features can be coupled directly with successor representations to provide an effective task-conditioning interface under the no finetuning evaluation protocol, without requiring downstream adaptation of a predictive world model.

Computational efficiency. Across the 2M-frame reward-free pretraining schedule, SWM requires at least **5**$\times$ fewer target-domain FLOPs (Appendix F) than CNN-FB while achieving higher zero-shot return. Our accounting includes frozen DINO inference for newly collected observations while reusing cached CLS features during replay updates, and excludes the external pretraining cost of DINO. For the Bernoulli-sampled backward-representation mixing used by FB, we report the expected compute under the configured mixing probability.

5.2 Robustness to Visual Perturbations

The previous results establish stronger control under the rendering distribution seen during pretraining. We next ask whether this advantage is robust to changes in visual appearance, while keeping the underlying dynamics and task objective fixed. We consider color, background, and camera shifts, together with their composition. Each perturbation is sampled once per episode and held fixed across frames. Individual perturbations are evaluated at easy and hard levels, while the combined condition sequentially applies the easy camera, background, and color shifts. Perturbation parameters are provided in Appendix C.2. For each downstream task, we infer z_R once from clean reward-labeled replay samples and keep the same task vector fixed across all perturbation conditions.

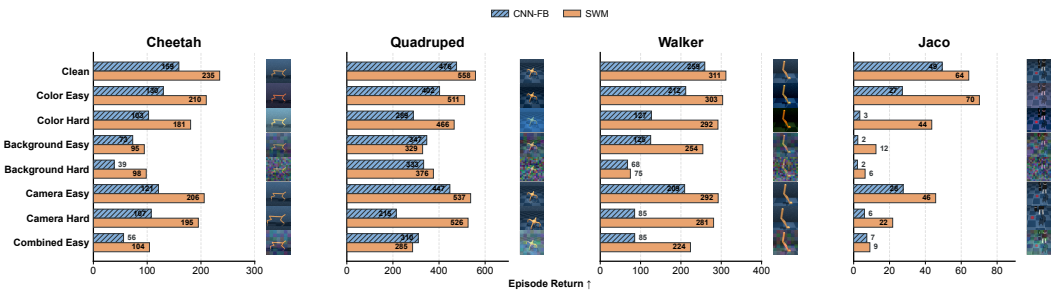

Figure 2: **The SWM advantage largely persists under test-time appearance shifts.** Mean episode return for CNN-FB and SWM under clean rendering, color, background, and camera perturbations, and their composition, across Cheetah, Quadruped, Walker, and Jaco. SWM achieves higher return in 30 of the 32 displayed domain–condition comparisons.

Figure 2 shows that the performance advantage of SWM largely persists under these visual shifts. SWM achieves higher mean return than CNN-FB in 26 of the 28 perturbed domain–condition comparisons. The only exceptions are Quadruped under Background Easy and Combined Easy. The separation often becomes pronounced under severe color and camera changes. On Walker, for example, return under Color Hard increases from 127 for CNN-FB to 292 for SWM, while Camera Hard increases from 85 to 281. On Quadruped, the corresponding comparisons are 289 versus 466 and 215 versus 526. Absolute return alone, however, does not establish stronger robustness, since SWM also starts from a higher clean return. We therefore normalize each perturbed return by the corresponding clean return. Across the 28 perturbed settings, SWM achieves higher retention than CNN-FB in 22 cases. The largest differences occur under hard color and camera shifts: averaged across domains, SWM retains 80.5% and 75.4% of its clean performance, compared with 45.3% and 39.4% for CNN-FB, respectively. See full retention results in Appendix D.1.

Together, the absolute returns and normalized retention show that the robustness advantage of SWM is not explained solely by its higher clean performance. Rather, the advantage persists after the pretrained features are mapped through g_θ and used by the FB policy, suggesting that the learned control representation preserves useful invariances from the frozen visual backbone.

5.3 Why Is the Learned Projector Necessary?

The robustness results show that useful invariances from DINO survive in the final policy, but they do not explain why the learned projector g_θ is required. We therefore separate two questions that are easily conflated: whether that representation is readily usable by the FB objective, and whether control-relevant information is present in the frozen DINO representation. To isolate the projector, we compare the full model (denoted SWM-CLS) with SWM-raw. Both use exactly the same frozen DINO encoder and three-frame temporal context; the only difference is that SWM-raw applies FB directly to the concatenated DINO features without g_θ.

Table 2: **Effect of g_θ.** Domain-average return.

Domain	SWM-raw	SWM-CLS
Walker	53.06	**311.44**
Quadruped	93.25	**558.14**
Cheetah	2.57	**234.77**
Jaco	2.64	**64.19**

Information in DINO is not sufficient for FB control. Table 2 shows that removing the projector causes a large drop in zero-shot control across every domain, with SWM-raw nearly collapsing on Cheetah and Jaco. This failure does not imply that the frozen DINO features lack control-relevant state information. As shown by the probes below, substantial pose and motion information remains recoverable from SWM-raw. The contrast therefore motivates a more specific question: does the projector improve FB control by making physical state variables uniformly more decodable, or does it change control utility in a way that is not captured by generic state probes?

Projection does not uniformly improve state decodability. We distinguish two groups of control-relevant state variables. The *visible-pose* target contains configuration variables reflected directly in the rendered observation, such as joint positions, torso height, uprightness, and center-of-mass height, while excluding global translation and velocity. The *motion* target contains dynamic

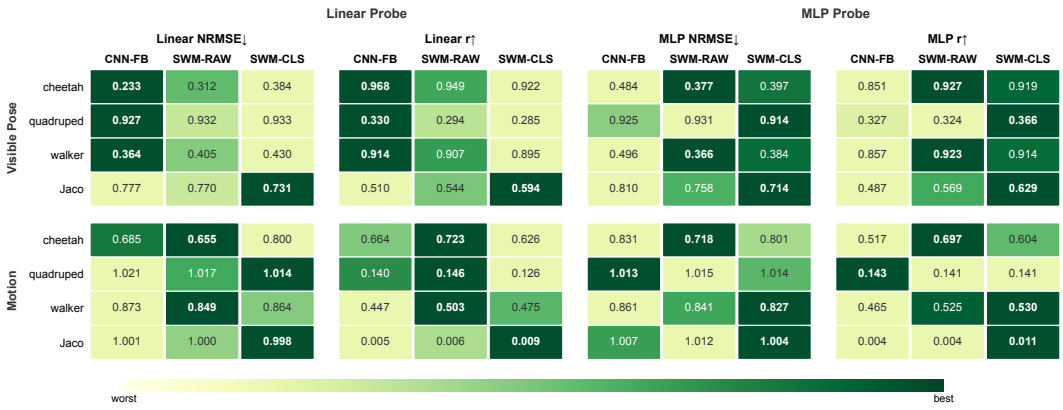

Figure 3: **Strong FB control is not explained by generic state decodability alone.** Linear and MLP probes recover visible-pose and motion variables from CNN-FB, SWM-raw, and SWM-CLS. Lower NRMSE and higher Pearson r are better. Raw DINO features already retain substantial control-relevant information, while projection changes recoverability non-uniformly across domains, target groups, and probe classes.

quantities such as velocities and angular momentum. We fit both linear Ridge and nonlinear MLP probes and report normalized RMSE and Pearson correlation on held-out episodes. The linear probe measures information that is directly accessible through a linear readout, whereas the MLP probe additionally captures information recoverable through a nonlinear decoder. Full target definitions and probing details are provided in Appendix B, with complete results in Appendix D.

Figure 3 reveals a substantially more nuanced pattern than a uniform increase in state decodability. For visible pose, CNN-FB is strongest under the linear probe on the three locomotion domains, while the DINO representations become competitive under nonlinear decoding. Most importantly, SWM-CLS does not consistently improve pose recovery over SWM-raw. Thus, the large control improvement produced by g_θ cannot be explained by a uniform increase in visible-state decodability. Motion recovery is similarly heterogeneous. On Cheetah, SWM-raw achieves stronger motion decoding than SWM-CLS under the nonlinear probe, yet its zero-shot return is only 2.57 compared with 234.77 for SWM-CLS. Thus, better recovery of dynamic state variables does not translate directly into better FB control. SWM-CLS remains competitive on Walker, while motion recovery is weak on Quadruped and near zero on Jaco for all methods.

The contrast between Table 2 and Figure 3 therefore separates state recoverability from control utility. SWM-raw can expose substantial pose and motion information, and can even outperform the projected representation on some probes, while still failing to train an effective FB policy. Conversely, learning g_θ yields large improvements in zero-shot control without uniformly improving generic state decodability. This suggests that the main benefit of the projector is not to maximize recoverability of every physical state variable, but to make the pretrained visual features substantially more usable by the successor-measure objective and the resulting policy.

6 Conclusion

We presented SWM, a successor-measure world model that represents the future through policy-conditioned long-horizon occupancy rather than step-wise predictive dynamics. SWM realizes this directly from pixels by combining frozen DINO features with short-term temporal context and lightweight FB-driven adaptation, enabling new rewards to select policies without downstream optimization or test-time planning. Across sixteen locomotion and manipulation tasks, SWM achieves higher mean zero-shot return than CNN-FB with substantially less target-domain compute, while remaining robust under visual shifts. Our analysis further shows that raw pretrained features can retain substantial decodable state information yet remain ineffective for FB control. Together, these results show that successor-measure world models can be built on reusable pretrained visual perception, with frozen DINO providing an effective instantiation in our experiments.

AI USE STATEMENT

In this work, we used generative AI tools to assist with method implementation and interpretation of results. We did not use generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, or design or provide feedback on research methodology or experiments. Assistance with translation, dataset cleaning or reformatting, qualitative or thematic data analysis was not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, and edit the manuscript to improve readability. We reviewed all AI-assisted work. We used LLM-generated figure components and drafts as intermediate material, while the final figures were organized and redrawn by the authors. We used Codex to implement the Dreamer-v3 and TD-MPC baselines based on the original codebases, including removing the reward signal during training and training a learnable reward head during inference, as described in the paper. After drafting, we used LLMs to suggest rephrasings for readability; the authors reviewed each suggestion and only incorporated edits they approved.

We take responsibility for the final content of this work, including text, claims, code, figures, and other artifacts produced with the aid of generative AI.

REFERENCES

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an rl agent. *arXiv preprint arXiv:2411.19418*, 2024.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning, 2025. URL https://arxiv.org/abs/2510.00739.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning, 2018. URL https://arxiv.org/abs/1606.05312.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint, 2021. URL https://arxiv.org/abs/2101.07123.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators, 2018. URL https://arxiv.org/abs/1812.07626.

Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T. Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, Vincent Sitzmann, and Yilun Du. Large video planner enables generalizable robot control, 2025. URL https://arxiv.org/abs/2512.15840.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993a. doi: 10.1162/neco.1993.5.4.613.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993b.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with jepa world models, 2025. URL https://arxiv.org/abs/2601.00844.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

N Hansen, X Wang, and H Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning, PMLR*, 2022a.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022b.

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. URL https://openreview.net/forum?id=Oxh5CstDJU.

Pranaya Jajoo, Harshit Sikchi, Siddhant Agarwal, Amy Zhang, Scott Niekum, and Martha White. Regularized latent dynamics prediction is a strong baseline for behavioral foundation models, 2026. URL https://arxiv.org/abs/2603.15857.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark, 2021. URL https://arxiv.org/abs/2110.15191.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and Guanya Shi. Bfm-zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning, 2025. URL https://arxiv.org/abs/2511.04131.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie envisioner: A unified world foundation platform for robotic manipulation, 2025. URL https://arxiv.org/abs/2508.05635.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

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

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence?, 2024. URL https://arxiv.org/abs/2303.18240.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL https://arxiv.org/abs/2203.12601.

Xiaolong Wang Nicklas Hansen, Hao Su. Learning massively multitask world models for continuous control, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qnWtw3l0jb.

Harshit Sikchi, Siddhant Agarwal, Pranaya Jajoo, Samyak Parajuli, Caleb Chuck, Max Rudolph, Peter Stone, Amy Zhang, and Scott Niekum. Rlzero: Direct policy inference from language without in-domain supervision, 2025a. URL https://arxiv.org/abs/2412.05718.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models, 2025b. URL https://arxiv.org/abs/2504.07896.

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

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, 2018. URL https://arxiv.org/abs/1801.00690.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models, 2025. URL https://arxiv.org/abs/2504.11054.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021a.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards, 2021b. URL https://arxiv.org/abs/2103.07945.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist?, 2023. URL https://arxiv.org/abs/2209.14935.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control, 2022. URL https://arxiv.org/abs/2203.06173.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021. URL https://arxiv.org/abs/2107.09645.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=D5RNACOZEI.

Appendix Contents

A FB Learning in the Visual Representation Space . . . 14
A.1 Visual History Representation . . . 14
A.2 Successor Density and the Role of $B(h^+)$ . . . 14
A.3 Population and In-Batch FB Objectives . . . 15
A.4 Gradient Routing, Actor Learning, and Reward Projection . . . 16
B Environment and Evaluation Details . . . 16
B.1 Task Suite . . . 16
B.2 Jaco Reaching Tasks . . . 16
B.3 Zero-Shot Evaluation and Statistics . . . 17
B.4 Representation Probe Protocol . . . 17
C Visual Robustness Perturbations . . . 18
C.1 Evaluation Protocol . . . 18
C.2 Perturbation Implementation . . . 18
D Additional Experimental Results . . . 19
D.1 Complete Robustness Results . . . 19
D.2 Projector Ablation . . . 19
D.3 Complete Probe Results . . . 19
E Implementation Details . . . 20
E.1 Training Procedure . . . 20
E.2 Visual Front Ends and DINO Feature Caching . . . 20
E.3 Optimization and Domain-Specific Settings . . . 21
E.4 Predictive World-Model Baselines . . . 22
F Target-Domain Compute Accounting . . . 23
F.1 Training Schedule . . . 23
F.2 Frozen-DINO Compute . . . 23
F.3 Backward-Derived Policy-Index Mixing . . . 24

A FB Learning in the Visual Representation Space

A.1 Visual History Representation

For each rendered observation o_t, the frozen DINOv2-Base backbone produces its final-layer CLS token

$$c_t = \mathrm{CLS}\left(\phi_{\mathrm{DINO}}(o_t)\right) \in \mathbb{R}^{768}. \tag{11}$$

We maintain the three most recent frozen features in temporal order,

$$\tilde{c}_t = [c_{t-2}; c_{t-1}; c_t] \in \mathbb{R}^{2304}, \tag{12}$$

and map this history into the visual control space through

$$h_t = g_\theta(\tilde{c}_t) = W\ \mathrm{LN}(\tilde{c}_t) + b \in \mathcal{H}, \qquad \mathcal{H} = \mathbb{R}^{512}. \tag{13}$$

At the beginning of an episode, unavailable history entries are filled with the reset feature. The first history is therefore $[c_0, c_0, c_0]$, followed by $[c_0, c_0, c_1]$ after the first transition.

DINO remains frozen throughout reward-free pretraining. Both the forward and backward representations operate on the same learned visual space,

$$F(h, a, z) \in \mathbb{R}^{d_z}, \qquad B(h^+) \in \mathbb{R}^{d_z}, \tag{14}$$

and the actor conditions only on h and the policy index z.

A.2 Successor Density and the Role of $B(h^+)$

For a latent-conditioned policy π_z, define its successor measure over the learned visual representation space as

$$M^{\pi_z}(h, a, \mathcal{X}) = \sum_{t \geq 0} \gamma^t \Pr\left(h_{t+1} \in \mathcal{X} \mid h_0 = h,\, a_0 = a,\, \pi_z\right), \qquad \mathcal{X} \subseteq \mathcal{H}. \tag{15}$$

Our convention starts future occupancy at the successor representation h_{t+1}.

The corresponding measure satisfies

$$\begin{aligned} M^{\pi_z}(h, a, \mathcal{X}) = {} & P_{\mathcal{H}}(\mathcal{X} \mid h, a) \\ & + \gamma \mathbb{E}_{\substack{h' \sim P_{\mathcal{H}}(\cdot \mid h, a) \\ a' \sim \pi(\cdot \mid h', z)}} \left[M^{\pi_z}(h', a', \mathcal{X})\right]. \end{aligned} \tag{16}$$

Writing the successor measure as a density with respect to the replay distribution $\rho_{\mathcal{H}}$,

$$M^{\pi_z}(h, a, \mathrm{d}h^+) = m_z(h, a, h^+) \rho_{\mathcal{H}}(\mathrm{d}h^+), \tag{17}$$

FB approximates

$$m_z(h, a, h^+) \approx F(h, a, z)^\top B(h^+). \tag{18}$$

The two factors play different roles. $F(h, a, z)$ represents future occupancy induced from the current state–action pair by the policy indexed by z, whereas $B(h^+)$ provides the basis coordinate associated with the visual state at which that occupancy is queried. The backward map therefore receives the future/query representation h^+ rather than the current representation h.

For a downstream reward $r : \mathcal{H} \to \mathbb{R}$, integrating the reward against this occupancy yields

$$\begin{aligned} Q_r^{\pi_z}(h, a) &= \int_{\mathcal{H}} r(h^+) M^{\pi_z}(h, a, \mathrm{d}h^+) \\ &\approx F(h, a, z)^\top z_r, \end{aligned} \tag{19}$$

where

$$z_r = \mathbb{E}_{h^+ \sim \rho_{\mathcal{H}}}\left[r(h^+) B(h^+)\right]. \tag{20}$$

Thus B provides both the query-state basis used to represent successor occupancy and the basis onto which unseen rewards are projected.

A.3 Population and In-Batch FB Objectives

Under the next-state occupancy convention, the population successor-density objective can be written, up to an overall constant, as

$$\begin{aligned}\mathcal{L}_{\mathrm{SM}} =& \mathbb{E}\left[\left(m_z(h,a,h^+) - \gamma \bar{m}_z(h',a',h^+)\right)^2\right] \\ & - 2\mathbb{E}\left[m_z(h,a,h')\right], \qquad a' \sim \pi(\cdot \mid h', z).\end{aligned} \tag{21}$$

The squared term performs Bellman regression at replay-distributed query states, while the linear term inserts the immediate occupancy contribution at the successor observed in the replay transition.

In implementation, a minibatch $\{(h_i, a_i, h_i')\}_{i=1}^b$ supplies both transition rows and future-query columns. We set

$$h_j^+ = h_j' \tag{22}$$

and construct

$$M_{ij}^{(k)} = F_k(h_i, a_i, z_i)^\top B(h_j^+), \qquad k \in \{1,2\}. \tag{23}$$

Each row i corresponds to a current tuple (h_i, a_i, z_i) and each column j to a candidate future visual representation. On the diagonal, $h_i^+ = h_i'$ is exactly the observed successor of transition i. For $i \neq j$, transition i is paired with the next state of another replay draw. These cross-transition pairs provide an in-batch product-distribution estimate of the query expectation and are not interpreted as unreachable negative states.

For

$$a_i' \sim \pi(\cdot \mid h_i', z_i), \tag{24}$$

the target matrices are

$$\bar{M}_{ij}^{(k)} = \bar{F}_k(h_i', a_i', z_i)^\top \bar{B}(h_j^+), \tag{25}$$

and the twin-head bootstrap target is

$$\bar{M}_{ij} = \min_{k\in\{1,2\}} \bar{M}_{ij}^{(k)}. \tag{26}$$

Let

$$\delta_i = \gamma d_i, \tag{27}$$

where d_i is the replayed environment discount. The implemented FB objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{FB}} = & \frac{1}{2b(b-1)} \sum_{k=1}^{2} \sum_{i\neq j} \left(M_{ij}^{(k)} - \delta_i \bar{M}_{ij}\right)^2 \\ & - \frac{1}{b} \sum_{k=1}^{2} \sum_{i} M_{ii}^{(k)} \\ & + \lambda_B \left[\frac{1}{b(b-1)} \sum_{i\neq j} G_{ij}^2 - \frac{2}{b} \sum_{i} G_{ii}\right],\end{aligned} \tag{28}$$

where

$$G_{ij} = B(h_i^+)^\top B(h_j^+). \tag{29}$$

The first two lines implement the in-batch successor-density estimator. The final line regularizes the backward representation toward an approximately orthogonal basis.

Although this estimator has an observed-pair/cross-pair structure, it is not an InfoNCE or classification objective: off-diagonal entries are regressed to bootstrapped successor-density targets rather than forced toward zero.

Both policy indices and backward embeddings are normalized to radius $\sqrt{d_z}$, with $d_z = 50$. Normalization fixes their scale while the Gram regularizer discourages correlations between backward directions.

Table 3: **Evaluation domains and downstream tasks.** Time limit and base control timestep refer to the environment before action repeat.

Domain	Tasks	Horizon	Base Δt	Repeat
Walker	stand, walk, run, flip	25 s	0.025	2
Quadruped	stand, walk, run, jump	20 s	0.020	2
Cheetah	walk, run, walk backward, run backward	10 s	0.010	2
Jaco	top-left, top-right, bottom-left, bottom-right	10 s	0.040	2

A.4 Gradient Routing, Actor Learning, and Reward Projection

The projector g_θ is optimized by the FB objective. Since both $F(h_i, a_i, z_i)$ and $B(h_j^+)$ consume online projected visual features, gradients reach the projector through both sides of the factorization, including the backward regularizer. Target scores are evaluated under stop-gradient.

The actor instead consumes a detached visual representation. We define

$$Q_z(h, a) = \min_{k \in \{1,2\}} F_k(h, a, z)^\top z \tag{30}$$

and optimize

$$\mathcal{L}_\pi = -\mathbb{E}_{\substack{h \sim \rho_\mathcal{H},\, z \sim p_\mathcal{Z} \\ a \sim \pi(\cdot|h,z)}} \left[Q_z(\mathrm{sg}(h), a) \right]. \tag{31}$$

Thus the control-specific visual representation is shaped by successor-measure learning rather than direct actor gradients.

After reward-free pretraining, all parameters remain frozen. Given K reward-labeled visual histories $\{(o_{k-2:k}, r_k)\}_{k=1}^K$, we compute

$$c_k = \mathrm{CLS}(\phi_{\mathrm{DINO}}(o_k)), h_k = g_\theta([c_{k-2}; c_{k-1}; c_k])$$

and estimate

$$\hat{z}_R = \frac{1}{K} \sum_{k=1}^{K} r_k B(h_k). \tag{32}$$

The task vector is normalized to the same radius used during reward-free pretraining:

$$z_R = \sqrt{d_z} \frac{\hat{z}_R}{\max(\|\hat{z}_R\|_2, \epsilon)}. \tag{33}$$

The selected policy $\pi(\cdot \mid h_t, z_R)$ is then executed without parameter updates, downstream policy learning, or test-time trajectory optimization.

B Environment and Evaluation Details

B.1 Task Suite

We pretrain one reward-free policy family per domain and evaluate the same pretrained model on four downstream rewards, yielding sixteen tasks.

The resulting policy decision intervals are 0.05 s for Walker, 0.04 s for Quadruped, 0.02 s for Cheetah, and 0.08 s for Jaco. Walker, Quadruped, and Cheetah contain 500 policy decisions per episode, whereas Jaco contains 125.

B.2 Jaco Reaching Tasks

The four Jaco tasks share the same dynamics and differ only in Cartesian target position:

$$\begin{aligned} q_{\mathrm{TL}} &= (-0.09,\ \ 0.09, 0.001), \quad q_{\mathrm{TR}} = (\ \ 0.09,\ \ 0.09, 0.001), \\ q_{\mathrm{BL}} &= (-0.09, -0.09, 0.001), \quad q_{\mathrm{BR}} = (\ \ 0.09, -0.09, 0.001). \end{aligned} \tag{34}$$

The coordinates are expressed in world coordinates. Top/bottom correspond to positive/negative world y, and left/right to negative/positive world x.

Let p denote the tool-center position and q the target. The reward is implemented as

$$\texttt{rewards.tolerance}\left(\|p - q\|_2, \texttt{bounds} = (0, 0.05), \texttt{margin} = 0.05\right). \tag{35}$$

B.3 Zero-Shot Evaluation and Statistics

During reward-free pretraining, agents never observe the downstream task rewards used for evaluation. After pretraining, every task is specified by $K = 5120$ reward-labeled visual histories. The inferred task vector remains fixed throughout evaluation and all learned SWM parameters remain frozen.

Each task is evaluated for ten episodes per training seed. Main results are reported over five training seeds. The task mean is obtained by first averaging the evaluation episodes associated with each trained agent and then averaging the resulting values across seeds.

All reported standard deviations use the sample standard deviation (Bessel-corrected) across seeds. For values $x_1, \ldots, x_n$,

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}. \tag{36}$$

B.4 Representation Probe Protocol

Shared physical-state banks. CNN-FB, SWM-raw, and SWM-CLS are evaluated on shared physical-state banks so that probe comparisons use exactly the same simulator states. Banks are collected with uniformly random actions from `walker_walk`, `quadruped_walk`, `cheetah_run`, and `jaco_reach_top_left`.

Each bank contains 50 parent episodes and 128 retained samples per episode, giving 6400 samples. We split by parent episode rather than individual frames: 30 episodes are used for training, 10 for validation, and 10 for testing, giving 3840/1280/1280 samples. Only samples with complete $[t-2, t-1, t]$ history are retained.

For Jaco bank construction, trajectories are extended to 129 decisions only to obtain 128 complete three-frame histories. Control evaluation itself uses the standard 125-decision episode.

Representations. CNN-FB uses its learned three-frame CNN representation. SWM-raw uses the 2304-dimensional concatenation of three frozen DINO CLS features. SWM-CLS uses the corresponding learned 512-dimensional projected representation. SWM-raw and SWM-CLS therefore differ only by the application of g_θ on the same underlying DINO history.

Linear probe. The linear probe is multi-output Ridge regression with an unpenalized intercept. Inputs are centered using the training split but are not whitened. We search

$$\alpha \in \{10^{-6}, 10^{-5}, \ldots, 10^{6}\} \tag{37}$$

and select α using normalized validation error. Exact ties are resolved in favor of the larger regularization coefficient.

Nonlinear probe. The MLP probe has architecture

$$D \to 512 \to 512 \to d_{\text{target}} \tag{38}$$

with ReLU hidden activations. Targets are standardized using the training-set mean and population standard deviation. The network is trained with AdamW for ten epochs using learning rate 10^{-3}, weight decay 10^{-4}, and batch size 1024. No scheduler, gradient clipping, or early stopping is used. MLP probe seeds are $0, 1, 2$.

Metrics. For target coordinate ℓ,

$$\mathrm{NRMSE}_\ell = \frac{\sqrt{\frac{1}{n_{\text{test}}}\sum_i (\hat{y}_{i\ell} - y_{i\ell})^2}}{\mathrm{SD}_{\text{train}}(y_\ell)}. \tag{39}$$

Pearson correlation is

$$r_\ell = \frac{(y_\ell - \bar{y}_\ell)^\top (\hat{y}_\ell - \bar{\hat{y}}_\ell)}{\|y_\ell - \bar{y}_\ell\|_2 \|\hat{y}_\ell - \bar{\hat{y}}_\ell\|_2}. \tag{40}$$

Coordinate-level metrics are averaged equally over valid targets.

Table 4: **Physical-state targets used for representation probing.** Dimensions denote visible-pose / motion target dimensionality.

Domain	Dim.	Visible pose	Motion
Cheetah	8/10	`physics.data.qpos[1:]` (8)	velocity (9), torso speed (1)
Walker	16/10	orientations (14), torso height (1), uprightness (1)	velocity (9), horizontal velocity (1)
Quadruped	29/20	non-root joint positions (16), uprightness (1), toe positions (12)	non-root joint velocities (16), torso velocity (3), speed norm (1)
Jaco	18/9	arm-joint sine/cosine (12), finger positions (3), pinch-site XYZ (3)	arm-joint velocities (6), finger-joint velocities (3)

Table 5: **Visual perturbation parameter ranges.** All values are sampled uniformly once per episode.

Perturbation	Easy	Hard
Contrast	$[0.75, 1.25]$	$[0.45, 1.75]$
Brightness	$[-25, 25]$	$[-60, 60]$
RGB channel scale	$[0.80, 1.20]$	$[0.45, 1.65]$
Background α	$[0.12, 0.25]$	$[0.28, 0.45]$
Texture grid	8×8	16×16
Camera zoom	$[1.02, 1.12]$	$[1.10, 1.35]$
Camera shift x/y	$[-0.08, 0.08]$	$[-0.18, 0.18]$
Combined	camera $\rightarrow$ background $\rightarrow$ color	—

C Visual Robustness Perturbations

C.1 Evaluation Protocol

Robustness evaluation modifies only rendered observations at test time. Environment dynamics, task rewards, action repeat, episode horizon, policy parameters, and task vectors remain unchanged.

For every downstream task, z_R is inferred once from clean reward-labeled histories and reused for all perturbation conditions. Each task–condition pair is evaluated for 50 episodes in total, we select 5 seeds and collect 10 episodes for each seed. Perturbation parameters and random textures are sampled once at episode reset and remain fixed across all frames.

C.2 Perturbation Implementation

Color. For rendered RGB image x, sampled contrast c, brightness b, and RGB gain vector g, the transformation is

$$x' = \mathrm{clip}\left([(x - 127.5)c + 127.5] \odot g + b, 0, 255\right). \tag{41}$$

Background. A random RGB texture is sampled on an 8×8 grid for Easy and a 16×16 grid for Hard, resized to the observation size, and blended over the rendered image using the sampled α. Simulator state and dynamics are unchanged.

Camera. Camera perturbations operate in image space. The sampled zoom determines crop size, while horizontal and vertical offsets shift the crop location by the specified fraction of image width and height. The crop is clipped to valid image bounds and resized back to 224×224.

Combined Easy applies easy camera, background, and color transformations sequentially in that order.

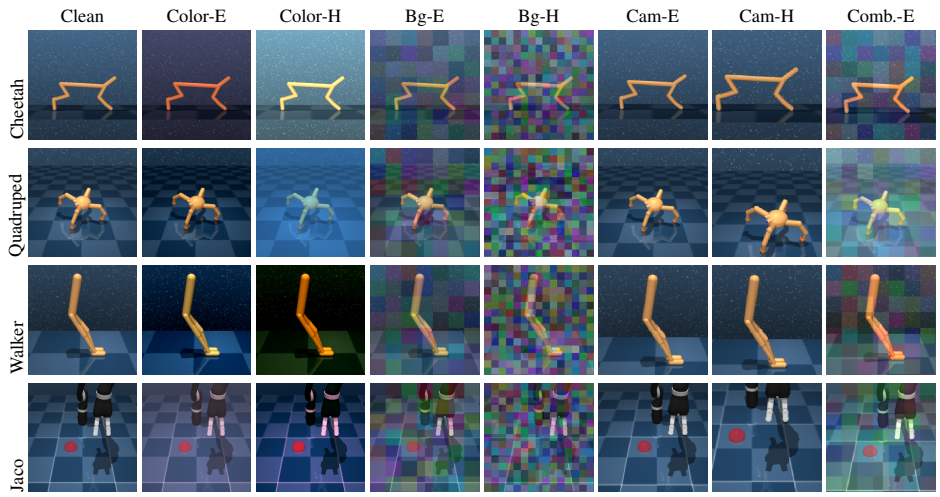

Figure 4: **Examples of all visual perturbation conditions.** Rows correspond to evaluation domains and columns to the clean rendering and seven test-time perturbations.

Table 6: **Absolute return under visual perturbations.** Higher is better.

Domain	Model	Clean	Color-E	Color-H	Bg-E	Bg-H	Cam-E	Cam-H	Comb.-E
Cheetah	CNN-FB	158.90	130.08	102.59	73.08	38.73	121.02	107.43	56.06
	SWM	**234.77**	**209.88**	**180.72**	**94.67**	**98.27**	**206.02**	**195.13**	**104.26**
Quadruped	CNN-FB	476.04	402.26	288.55	**346.79**	333.21	447.05	214.74	**310.07**
	SWM	**558.14**	**510.94**	**465.69**	328.78	**375.63**	**537.12**	**526.02**	285.28
Walker	CNN-FB	259.15	211.89	126.94	125.01	67.65	209.18	85.21	85.36
	SWM	**311.44**	**303.42**	**291.79**	**254.33**	**74.69**	**292.34**	**280.93**	**223.67**
Jaco	CNN-FB	49.37	27.16	3.43	2.30	2.11	27.59	6.02	7.34
	SWM	**64.19**	**70.13**	**43.55**	**12.44**	**6.38**	**45.73**	**21.76**	**9.04**

D Additional Experimental Results

D.1 Complete Robustness Results

Across the 28 non-clean comparisons, SWM achieves higher absolute return in 26 and higher retention in 22. Averaged across domains, hard-color retention is 80.5% for SWM versus 45.3% for CNN-FB, while hard-camera retention is 75.4% versus 39.4%.

D.2 Projector Ablation

SWM-raw and SWM-CLS use the same frozen DINO encoder and identical three-frame temporal context. SWM-raw removes both LayerNorm and the linear projector.

The large control gap is not accompanied by a uniform increase in generic state decodability, motivating the analysis below.

D.3 Complete Probe Results

The probe ranking differs substantially from the control ranking. In Cheetah, for example, SWM-raw has higher nonlinear motion correlation than SWM-CLS (0.697 versus 0.604), despite a zero-shot return of 2.57 versus 234.77. Motion recovery is weak on Quadruped and near zero on Jaco for all three representations.

Table 7: **Clean-normalized retention under visual perturbations.** Values are percentages of clean return. Higher is better.

Domain	Model	Color-E	Color-H	Bg-E	Bg-H	Cam-E	Cam-H	Comb.-E
Cheetah	CNN-FB	81.9	64.6	**46.0**	24.4	76.2	67.6	35.3
	SWM	**89.4**	**77.0**	40.3	**41.9**	**87.8**	**83.1**	**44.4**
Quadruped	CNN-FB	84.5	60.6	**72.8**	**70.0**	93.9	45.1	**65.1**
	SWM	**91.5**	**83.4**	58.9	67.3	**96.2**	**94.2**	51.1
Walker	CNN-FB	81.8	49.0	48.2	**26.1**	80.7	32.9	32.9
	SWM	**97.4**	**93.7**	**81.7**	24.0	**93.9**	**90.2**	**71.8**
Jaco	CNN-FB	55.0	6.9	4.7	4.3	55.9	12.2	**14.9**
	SWM	**109.3**	**67.8**	**19.4**	**9.9**	**71.2**	**33.9**	14.1

Table 8: **Effect of the learned projector g_θ.** Domain-average zero-shot return.

Domain	SWM-raw	SWM-CLS
Walker	53.06	**311.44**
Quadruped	93.25	**558.14**
Cheetah	2.57	**234.77**
Jaco	2.64	**64.19**

E Implementation Details

E.1 Training Procedure

Algorithm 1 summarizes online reward-free pretraining and zero-shot inference.

E.2 Visual Front Ends and DINO Feature Caching

SWM uses `facebook/dinov2-base` with 224×224 rendered images. Each frame produces a 768-dimensional CLS feature. SWM-CLS applies LayerNorm(2304) followed by a bias-enabled Linear$(2304, 512)$ layer. No output nonlinearity is used. SWM-raw removes both LayerNorm and the linear projection.

CNN-FB uses the same 224×224 image resolution with three-frame stacking, RandomShiftsAug with padding 4, and four 32-channel 3×3 convolutions with strides $[2, 1, 1, 1]$ followed by ReLU. This is the implementation taken from DrQ Yarats et al. (2021).

CNN-FB visual encoder. As summarized in Table 10, CNN-FB uses a trainable CNN with a three-frame observation history. We concatenate three consecutive 224×224 RGB frames along the channel dimension, yielding an input of size $9 \times 224 \times 224$. The encoder consists of four convolutional layers, each with 32 output channels, a 3×3 kernel, no padding, and a ReLU activation. The first layer uses stride 2, and the remaining layers use stride 1, producing a feature map of size $32 \times 105 \times 105$. We then apply adaptive average pooling to obtain a $32 \times 35 \times 35$ feature map and flatten it into a 39,200-dimensional visual representation. This pooling step preserves the representation dimensionality of the original 84×84 encoder while allowing 224×224 inputs. The dimension reported in Table 10 refers to this flattened encoder output, before the subsequent FB network layers.

Frozen DINO feature caching. The frozen DINO backbone is evaluated only when the environment produces a new RGB observation. For o_t, its 768-dimensional CLS feature c_t is computed once and inserted into a length-three feature deque. The two previous features are reused to form

$$\tilde{c}_t = [c_{t-2}; c_{t-1}; c_t]. \tag{42}$$

The replay buffer stores these frozen three-frame feature histories together with transition metadata rather than storing the projected h_t as its observation.

During replay optimization, the stored DINO features are reused directly and the DINO backbone is not run again. The trainable projector g_θ is recomputed on sampled histories at every update.

Table 9: **Full representation-probing results.** Lower NRMSE and higher Pearson r are better.

Property	Domain	Representation	Lin. NRMSE↓	Lin. r ↑	MLP NRMSE↓	MLP r ↑
Visible pose	Cheetah	CNN-FB	0.233	0.968	0.484	0.851
		SWM-raw	0.312	0.949	0.377	0.927
		SWM-CLS	0.384	0.922	0.397	0.919
	Quadruped	CNN-FB	0.927	0.330	0.925	0.327
		SWM-raw	0.932	0.294	0.931	0.324
		SWM-CLS	0.933	0.285	0.914	0.366
	Walker	CNN-FB	0.364	0.914	0.496	0.857
		SWM-raw	0.405	0.907	0.366	0.923
		SWM-CLS	0.430	0.895	0.384	0.914
	Jaco	CNN-FB	0.777	0.510	0.810	0.487
		SWM-raw	0.770	0.544	0.758	0.569
		SWM-CLS	0.731	0.594	0.714	0.629
Motion	Cheetah	CNN-FB	0.685	0.664	0.831	0.517
		SWM-raw	0.655	0.723	0.718	0.697
		SWM-CLS	0.800	0.626	0.801	0.604
	Quadruped	CNN-FB	1.021	0.140	1.013	0.143
		SWM-raw	1.017	0.146	1.015	0.141
		SWM-CLS	1.014	0.126	1.014	0.141
	Walker	CNN-FB	0.873	0.447	0.861	0.465
		SWM-raw	0.849	0.503	0.841	0.525
		SWM-CLS	0.864	0.475	0.827	0.530
	Jaco	CNN-FB	1.001	0.005	1.007	0.004
		SWM-raw	1.000	0.006	1.012	0.004
		SWM-CLS	0.998	0.009	1.004	0.011

Table 10: **Visual front ends used in the main comparison.**

Model	History	Encoder	FB representation
SWM-CLS	3	frozen DINOv2-B CLS	$2304 \to 512$
SWM-raw	3	frozen DINOv2-B CLS	2304-D
CNN-FB	3	trainable CNN	39,200-D

Therefore each newly rendered observation incurs one DINO forward pass, replaying a transition incurs no additional DINO inference, and three-frame temporal context does not require three new DINO evaluations per environment step.

E.3 Optimization and Domain-Specific Settings

Policy indices are sampled as

$$\tilde{z} \sim \mathcal{N}(0, I_{d_z}), \qquad z = \sqrt{d_z}\frac{\tilde{z}}{\|\tilde{z}\|_2}, \qquad d_z = 50. \tag{43}$$

During online collection, the exploration index is resampled every 300 agent steps. Each replay row is independently selected with probability 0.5 for backward-derived policy-index mixing. Hindsight-future mixing is disabled.

Target F and B parameters are updated as

$$\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}, \qquad \tau = 0.01. \tag{44}$$

There is no separate target DINO backbone or target projector.

Algorithm 1 SWM Reward-Free Pretraining and Zero-Shot Inference

Require: Frozen DINO encoder ϕ, projector g_θ, forward maps F_1, F_2, backward map B, actor π, replay $\mathcal{D}$

Initialize $g_\theta, F_1, F_2, B, \pi$
Initialize target networks $\bar{F}_1, \bar{F}_2, \bar{B}$

Reward-free interaction
for each environment step t **do**
  $c_t \leftarrow \mathrm{CLS}(\phi(o_t))$
  $\tilde{c}_t \leftarrow [c_{t-2}; c_{t-1}; c_t]$
  $h_t \leftarrow g_\theta(\tilde{c}_t)$
  Sample or reuse $z_{\mathrm{expl}} \sim p_{\mathcal{Z}}$
  $a_t \sim \pi(\cdot \mid h_t, z_{\mathrm{expl}})$
  Execute a_t and observe the next image
  Store frozen feature transition and action in $\mathcal{D}$
end for

FB and actor updates
for each gradient update **do**
  Sample $\{(\mathbf{c}_i, a_i, \mathbf{c}'_i, d_i)\}_{i=1}^{b} \sim \mathcal{D}$
  $h_i \leftarrow g_\theta(\mathbf{c}_i)$, $h'_i \leftarrow g_\theta(\mathbf{c}'_i)$
  Sample normalized policy indices z_i
  Apply configured backward-derived-z mixing
  $a'_i \sim \pi(\cdot \mid h'_i, z_i)$
  Set query representations $h_j^+ \leftarrow h'_j$
  for $k \in \{1, 2\}$ **do**
    $M_{ij}^{(k)} \leftarrow F_k(h_i, a_i, z_i)^\top B(h_j^+)$
    $\bar{M}_{ij}^{(k)} \leftarrow \bar{F}_k(\mathrm{sg}(h'_i), a'_i, z_i)^\top \bar{B}(\mathrm{sg}(h_j^+))$
  end for
  $\bar{M}_{ij} \leftarrow \min_k \bar{M}_{ij}^{(k)}$
  Update g_θ, F_1, F_2, B with Eq. 28
  Update π with Eq. 31
  Polyak-update $\bar{F}_1, \bar{F}_2, \bar{B}$
end for

Zero-shot inference
Require: Reward-labeled histories $\{(o_{k-2:k}, r_k)\}_{k=1}^{K}$
$c_k \leftarrow \mathrm{CLS}(\phi(o_k))$
$\tilde{c}_k \leftarrow [c_{k-2}; c_{k-1}; c_k]$
$h_k \leftarrow g_\theta(\tilde{c}_k)$
$\hat{z}_R \leftarrow K^{-1} \sum_k r_k B(h_k)$
$z_R \leftarrow \sqrt{d_z}\hat{z}_R / \max(\|\hat{z}_R\|_2, \epsilon)$
Execute the mean action of $\pi(\cdot \mid h_k, z_R)$

E.4 Predictive World-Model Baselines

Dreamer-v3 and TD-MPC2 are adapted to the reward-free downstream setting. Their pretrained representation and latent dynamics remain frozen when downstream reward labels become available.

For every task, the same $K = 5120$ labeled examples available to SWM are used to fit a scalar reward predictor in the frozen latent space. The reward head has architecture

$$D \to 256 \to 256 \to 1 \tag{45}$$

with SiLU activations. It is optimized with reward MSE using Adam with learning rate 3×10^{-4}, batch size 256, zero weight decay, and 5000 updates.

After fitting the reward predictor, downstream actions are selected by CEM in the frozen dynamics.

Table 11: **Main optimization and architecture settings.**

Setting	Value	Setting	Value
Optimizer	Adam	Learning rate	10^{-4}
d_z	50	Projector output	512
F hidden width	1024	Actor hidden width	1024
B hidden width	256	Forward feature width	512
Discount γ	0.99	Target τ	0.01
Actor std	0.2	Noise clip	0.3
Replay capacity	5000 episodes	Update interval	2 agent steps
z resampling	300 steps	B-derived z prob.	0.5
λ_B	1.0	Normalize B, z	yes
Hindsight future	0	Auxiliary Q / IDM	off / 0
Seed frames	4000	Reward samples K	5120
Evaluation interval	10k frames	Training frames	2,000,010

Table 12: **Domain-specific training settings.**

Domain	Batch size	Decisions/episode	Frames	K
Walker	128	500	2,000,010	5120
Quadruped	128	500	2,000,010	5120
Cheetah	1024	500	2,000,010	5120
Jaco	1024	125	2,000,010	5120

The first action of the final elite mean is executed before replanning at the next environment step. Neither the representation nor the learned dynamics are updated during downstream evaluation.

F Target-Domain Compute Accounting

We count computation incurred during reward-free pretraining on the target control domain. One multiply-add is counted as two FLOPs. The accounting includes current and target F/B computation, actor computation, collection policy inference, the SWM projector, and expected backward-derived-z computation.

Rendering, simulator physics, logging, evaluation episodes, optimizer arithmetic, EMA-update arithmetic, and external DINO pretraining are excluded.

The accounting uses the actual domain-specific training batch sizes: **128 for Walker and Quadruped, and 1024 for Cheetah and Jaco**.

F.1 Training Schedule

For $T = 2{,}000{,}010$ environment frames and action repeat $R = 2$,

$$N = \left\lfloor \frac{T}{R} \right\rfloor = 1{,}000{,}005 \tag{46}$$

agent steps are executed. The first 4000 environment frames correspond to 2000 seed agent steps. With one update every two agent steps, the configured schedule contains 499,003 optimization updates.

F.2 Frozen-DINO Compute

Under the same convention, one DINOv2-Base forward pass at 224×224 requires approximately 46.32 GFLOPs. The analytical count includes patch embedding and the major matrix multiplications in the 12 ViT-B/14 transformer blocks; element-wise operations such as normalization and activation are excluded.

As described in Appendix E.2, DINO is evaluated only for newly collected images. Cached CLS histories are reused during replay, so DINO is not rerun for replayed samples and the three-frame

Table 13: **Predictive-world-model downstream planning settings.**

Setting	Value
Planning method	CEM
Planning horizon	15
Population	512
Elites	64
Iterations	5
Discount	0.99
Initial mean / std	$0/1$
Minimum std	0.05
Warm start	no
Terminal value	0
Policy-prior trajectories	no

Table 14: **Target-domain reward-free pretraining compute.** Values are PFLOPs (10^{15} FLOPs).

Domain	SWM	CNN-FB	CNN/SWM
Walker	52.177	285.330	5.469×
Quadruped	52.184	285.338	5.468×
Cheetah	91.495	2183.554	23.865×
Jaco	92.748	2276.641	24.547×

history does not triple DINO inference. External DINO pretraining is excluded because it is not incurred on the target domain.

F.3 Backward-Derived Policy-Index Mixing

Each replay row is selected independently for backward-derived policy-index mixing with probability

$$p_B = 0.5. \tag{47}$$

For U updates of batch size b, the expected number of selected rows is

$$\mathbb{E}[N_B] = p_B U b. \tag{48}$$

The corresponding expected backward-network computation is included in the reported totals.

Across the four evaluated domains, SWM requires at least $5\times$ fewer target-domain FLOPs than CNN-FB.