REPRESENTATION WORLD MODEL: LEARNING STATES, TRANSITION AND EXECUTABLE PLANS IN REPRESENTATION

Anonymous authors
Paper under double-blind review

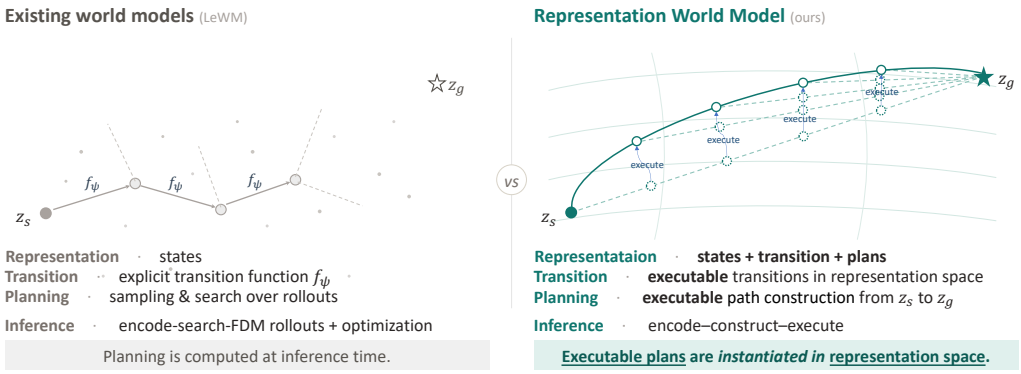


Figure 1: **Representing states, transitions, and executable plans in latent space.** Existing world models such as LeWM represent states in latent space, while modeling transitions with an explicit dynamics function f_ψ, and planning through rollout-based sampling and search at inference time (left). RWM instead learns a latent geometry that jointly represents states, transitions, and executable plans, turning otherwise unconstrained regions of the representation space into actionable paths connecting z_s and z_g (right).

ABSTRACT

We propose the *Representation World Model* (RWM), which learns states, transitions, and executable plans directly in representation space. Unlike existing world models that typically learn latent representations together with explicit dynamics models and perform planning through search, optimization, or policy-based prediction, RWM directly incorporates planning into the learned representation geometry. RWM learns the representation geometry by applying inverse-dynamics supervision locally along latent paths constructed from endpoint representations, requiring these paths to preserve task-relevant state and transition information. At inference, planning is performed by directly constructing a latent path between the current and goal representations, with inverse dynamics used to recover the corresponding actions, without recursive rollouts or action-space search. Experiments on continuous-control benchmarks demonstrate the effectiveness of RWM for direct planning, while results on robotic manipulation further show its potential to extend to more complex embodied control tasks. These results suggest that planning directly in representation space provides a promising alternative to conventional world-model planning.

1 INTRODUCTION

Recent advances in foundation models have demonstrated the power of scalable representation learning across vision, language, and robotics (Radford et al., 2021; Oquab et al., 2024; Brown et al., 2020; Driess et al., 2023; Zitkovich et al., 2023). A good representation should preserve and organize the information needed by downstream tasks. For embodied agents, such relevance is naturally

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.

defined by action and interaction. Visual–action trajectories therefore provide a direct source for learning action-relevant representations, motivating recent latent world models that incorporate actions into representation learning (Maes et al., 2026b; Gao & Xu, 2026).

However, existing approaches still rely on proxy prediction together with predefined representation priors such as SIGReg. Such priors are introduced to prevent representation collapse under forward prediction, but impose fixed assumptions on the latent space that may become restrictive as data and tasks scale (Boylan & Hokamp, 2026). Meanwhile, proxy objectives encourage representations to preserve whatever is useful for prediction, including state information that may be irrelevant to action (Liu et al., 2026). Moreover, they do not directly require the latent geometry itself to be organized around task-relevant transitions or planning (Li et al., 2026). These limitations motivate learning representations more directly from action-level supervision, where the task signal itself rules out the trivial collapsed solution and determines both what information should be emphasized and how it should be structured.

Yet direct action supervision introduces a second challenge: when the encoder and planner are jointly learned, the action loss constrains only their combined output, leaving the division between representation and planning underdetermined. A strong planner may compensate for weak representation structure, while an overly simple planner leaves planning computations that the encoder cannot absorb, making end-to-end training ineffective.

We therefore ask: **How can action-level supervision be directed into the representation itself?**

Our key idea is to constrain planning to a direct geometric construction in representation space and apply action supervision to the resulting latent transitions, thereby directing the learning signal toward the representation geometry itself. Based on this idea, we introduce the *Representation World Model* (RWM), which learns task-relevant states, transitions, and executable plans directly in a shared representation space, as illustrated in Fig. 1.

Given an action-labeled trajectory, RWM constructs a latent path between its endpoint representations and applies a shared inverse dynamics model locally along the path to recover the corresponding actions. This objective directly shapes the representation to preserve action-relevant information and organize it into a geometry that supports executable plan construction.

At inference time, RWM encodes the current and goal observations, directly constructs a latent path between them, and decodes actions locally along the path. Planning is therefore realized through geometric path construction followed by action decoding, without recursive forward rollouts or action-sequence search. The same construction used during training provides the latent plan, while inverse dynamics translates its local transitions into actions. We evaluate this formulation on continuous-control and robotic manipulation benchmarks, demonstrating its utility for search-free control.

The contributions of this paper are as follows:

- We introduce the *Representation World Model* (RWM), a formulation that learns action-relevant states, transitions, and planning structure within a shared representation space.
- We propose an inverse dynamics objective on constructed latent paths, which directly uses action supervision to shape both the information preserved by the representation and the geometry in which executable plans can be constructed.
- We demonstrate better search-free control performance on continuous-control benchmarks and further show the potential of RWM for more complex embodied control tasks.

2 Related Work

2.1 Representation Learning

Representation learning seeks to map high-dimensional observations into latent variables that preserve information useful for downstream tasks. Early approaches based on autoencoders and variational autoencoders learned representations primarily through reconstruction, while later self-supervised methods shifted the focus toward invariance and semantic structure. This line of work, including InstDisc (Wu et al., 2018), MoCo (He et al., 2020), SimCLR (Chen et al., 2020), BYOL (Grill et al., 2020), Barlow Twins (Zbontar et al., 2021), VICReg (Bardes et al., 2022),

DINO (Caron et al., 2021), and iBOT (Zhou et al., 2022), showed that transferable visual representations can emerge from large-scale self-supervision without explicitly reconstructing observations.

More recent work has incorporated temporal structure into representation learning. Predictive approaches such as V-JEPA (Bardes et al., 2024) learn by predicting latent representations across time, while embodied variants (Maes et al., 2026b; Nam et al., 2026) further model state transitions and interaction dynamics in latent space for continuous-control. Related approaches exploit temporal structure and reachability through planning-aligned objectives to shape latent representations for downstream decision making (Li et al., 2026). This moves learned state representations beyond static visual structure toward transition- and decision-relevant structure.

RWM takes a further step by structuring not only represented states, but also the latent regions between them. A fixed path-construction rule provides a simple geometric scaffold, while inverse-dynamics supervision shapes the representation onto this scaffold, turning constructed paths into executable plans.

2.2 World Models

World models learn internal representations of an environment together with mechanisms for modeling its evolution. Early latent world models (Ha & Schmidhuber, 2018) established the use of compact latent variables for sequential prediction, while PlaNet (Hafner et al., 2019) and the Dreamer family Hafner et al. (2020; 2021; 2025) demonstrated that learned latent dynamics can support effective model-based control. MuZero (Schrittwieser et al., 2020) further showed that representations and learned dynamics can be optimized specifically for planning without reconstructing the underlying observations.

As world models scale to richer visual environments, generative systems such as Genie (Bruce et al., 2024) and Cosmos (Agarwal et al., 2025) model future evolution by explicitly generating action-conditioned visual observations. While expressive, such visual imagination can be computationally expensive for downstream control, as each candidate future requires generating high-dimensional observations. This motivates recent latent world models that predict directly in compact representation spaces. LeWM (Maes et al., 2026a), for example, learns action-conditioned latent dynamics from visual–action trajectories without reconstructing future observations, enabling substantially more efficient latent-space planning.

However, such latent representations are still primarily shaped by predictive objectives. Avoiding collapse requires additional latent-space regularization such as SIGReg, introducing fixed assumptions that may become restrictive as data and task increase. Besides, prediction may also preserve behavior-irrelevant factors and fail to learn a geometry aligned with task-relevant search and planning. RWM takes a further step by grounding representation learning directly in action supervision, allowing both state information and planning structure to be learned within the representation space.

2.3 Planning with Learned World Models

A common planning strategy of using latent world model is to imagine and evaluate future trajectories at inference time. Methods such as CEM or MPPI sample candidate action sequences, predict their outcomes, and select actions according to task objectives. Recent latent approaches, including DINO-WM (Zhou et al., 2025), LeWM (Maes et al., 2026b), and Fast-LeWM (Gao & Xu, 2026), improve the underlying representations and predictive models, but planning still relies on evaluating candidate futures at inference time.

Instead, another approach amortizes goal-directed control into a learned policy or inverse dynamics model. Goal-conditioned policies directly predict actions from the current state and target, avoiding iterative online search. GC-IDM (Nguyen et al., 2026) follows this paradigm by recovering actions from current and goal representations.

RWM provides a third planning paradigm. Unlike search-based methods, it does not evaluate candidate futures; unlike goal-conditioned policies, it does not directly map endpoints to actions. Instead, RWM constructs a latent path between the current and goal states and decodes the path into actions, making planning a direct path-construction process in representation space.

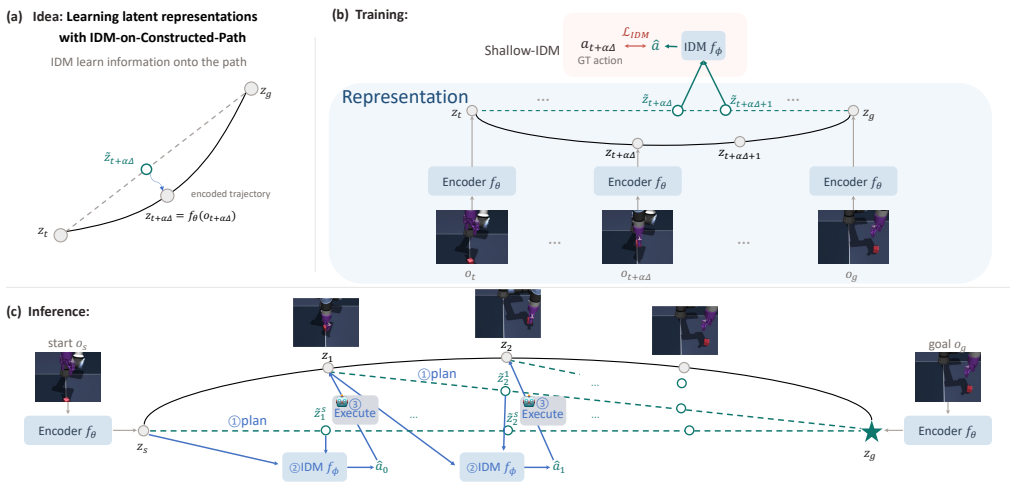


Figure 2: **Overview of RWM**. During training, RWM constructs latent paths between encoded states and learns their executable structure through action supervision. At inference, it constructs paths between current and goal, and decodes locally on it, enabling closed-loop control.

3 METHOD

Our goal is to learn a representation in which states, actionable transitions, and the structure required for executable planning are jointly expressed in the latent geometry. We first formalize the problem, and then introduce how RWM constructs and learns such a representation space.

3.1 PROBLEM SETUP

We consider goal-conditioned control from an offline dataset $\mathcal{D}$ of action-labeled trajectories $\tau = (o_0, a_0, o_1, a_1, \ldots, a_{T-1}, o_T)$, where $o_t \in \mathcal{O}$ denotes an observation (or fixed-length observation chunk) and $a_t \in \mathcal{A}$ denotes the action (or fixed-length action chunk), that connects two consecutive observations. An encoder $f_\theta : \mathcal{O} \to \mathcal{Z}$ maps each observation to a latent representation

$$z_t = f_\theta(o_t) \in \mathcal{Z}. \tag{1}$$

Conventional latent world models additionally learn a forward transition model $f_\psi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$, such that $z_{t+1} = f_\psi(z_t, a_t)$. Given start and goal states (z_s, z_g), planning then requires searching over candidate actions through repeated rollouts:

$$\mathbf{a}^* = \arg \min_{\mathbf{a}=(a_s,\ldots,a_{s+H-1})} d\left(f_\psi^{(H)}(z_s, \mathbf{a}), z_g\right),$$

where $f_\psi^{(H)}$ denotes an H-step rollout and d is a latent discrepancy.

3.2 REPRESENTATION WORLD MODEL

Instead of relying on recursive forward prediction or search, RWM learns actionable transition structure along constructed latent paths. Planning can then be realized by directly constructing a path between the current and goal states and decoding its transitions into actions, as illustrated in Fig. 2.

Consider two observations o_t and o_u from the same trajectory, where $t < u$. After encoding the endpoints as $z_t = f_\theta(o_t)$ and $z_u = f_\theta(o_u)$, we construct the representation at temporal position m, where $t < m < u$, by

$$\tilde{z}_m = (1 - \alpha_{t,m,u}) z_t + \alpha_{t,m,u} z_u, \quad \alpha_{t,m,u} = \frac{m-t}{u-t}.$$

Importantly, $\tilde{z}_m$ is only a geometrically constructed latent intermediate before learning; interpolation alone does not imply a valid transition or executable trajectory. RWM therefore trains the

representation so that these constructed intermediates acquire task-relevant state, transition, and action semantics.

This construction exposes the previously unconstrained regions between observed states to learning. Because every intermediate point is determined by the endpoint representations, supervision applied along the constructed path also shapes the endpoints and, consequently, the geometry of the representation space itself. The objectives introduced in Sec. 3.3 make this geometry informative about the actions required to traverse them.

Planning as a geometric operation in representation space. The same construction can be used directly for planning. Given current and goal observations (o_t, o_g), RWM first encodes $z_t = f_\theta(o_t)$, $z_g = f_\theta(o_g)$, and constructs intermediate transition representations $\tilde{z}_m^t = (1-\alpha_{t,m,g})z_t + \alpha_{t,m,g} z_g$, with $\alpha_m = \frac{m-t}{g-t}$, $m = t+1, \ldots, g-1$.

Together with $\tilde{z}_t^t = z_t$ and $\tilde{z}_g^t = z_g$, these representations define a latent transition path

$$\mathcal{P}_{s:g} = (\tilde{z}_t^t, \tilde{z}_{t+1}^t, \ldots, \tilde{z}_g^t).$$

Unlike rollout-based planning, each intermediate representation is constructed directly from the encoded endpoints, without recursively predicting intermediate states. The key challenge is therefore to make the constructed path actionable, such that it can serve as an executable plan. We address this through action-grounded representation learning.

3.3 Action-Grounded Representation Learning

For a sampled trajectory segment $(o_t, a_t, \ldots, a_{u-1}, o_u)$ we encode $z_i = f_\theta(o_i)$, and construct transition representations along the segment: $\tilde{z}_i = (1-\alpha_i)z_t + \alpha_i z_u$, with $\alpha_i = \frac{i-t}{u-t}$, $i = t, \ldots, u$.

Training is majorly with inverse dynamics that provides action-relevant supervision along the constructed path.

Action supervision on constructed path. Interpolation alone only specifies latent locations between two endpoints. There is no guarantee that the resulting path represents how the system actually transitions between states. We therefore train an inverse dynamics model $f_\phi : \mathcal{Z} \times \mathcal{Z} \to \mathcal{A}$ to recover the action between neighboring interpolated representations:

$$\mathcal{L}_{\text{IDM}}^{\text{int}} = \frac{1}{u-t} \sum_{i=t}^{u-1} |f_\phi(\tilde{z}_i, \tilde{z}_{i+1}) - a_i|_2^2 . \tag{2}$$

This objective grounds the constructed path in action semantics: neighboring interpolated representations must support recovery of the actions executed along the demonstrated trajectory. It therefore encourages the regions between encoded states to represent actionable transitions rather than arbitrary geometric intermediates.

Optionally, we additionally apply inverse dynamics directly to neighboring encoded states, $\mathcal{L}_{\text{IDM}}^{\text{enc}} = \frac{1}{u-t} \sum_{i=t}^{u-1} |f_\phi(z_i, z_{i+1}) - a_i|_2^2$. which provides a direct action-grounding signal on observed latent transitions. This term is not essential to the construction, but can serve as an auxiliary supervision for the encoded representation.

Why IDM on interpolation shapes the representation. IDM first prevents trivial feature collapse: if all observations shared the same representation, identical latent inputs would correspond to different actions, making the IDM objective unable to satisfy.

It also encourages the encoder to preserve transition-relevant state. If distinct object configurations collapse to similar latent endpoints, they induce similar interpolated paths, creating conflicting IDM supervision.

Thus, interpolation-based IDM grounds the constructed path in action semantics and encourages the representation to retain the state information required to recover the correct actions along it.

Soft interpolation consistency. Optionally, we softly align each constructed transition representation with the encoded state at the corresponding temporal position: $\mathcal{L}_{\text{consist}} = \frac{1}{u-t-1}\sum_{i=t+1}^{u-1}|\tilde{z}_i - z_i|_2^2$.

Unlike IDM, this objective does not determine what information the representation should preserve. It only discourages the constructed path from drifting away from the geometry of observed trajectories.

Overall objective. The complete objective is

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{IDM}}^{\text{int}} + \lambda_{\text{IDM}}^{\text{enc}}\mathcal{L}_{\text{IDM}}^{\text{enc}} + \lambda_{\text{consist}}\mathcal{L}_{\text{consist}}. \tag{3}$$

The three terms provide complementary constraints on the learned representation. $\mathcal{L}_{\text{IDM}}^{\text{int}}$ grounds endpoint-constructed transitions in executable actions and propagates this supervision through the interpolated path to the endpoint representations. $\mathcal{L}_{\text{IDM}}^{\text{enc}}$ optionally provides direct action supervision on observed latent transitions. $\mathcal{L}_{\text{consist}}$ supplies soft geometric supervision by aligning constructed intermediates with the corresponding observation-induced states. During training, only the encoder f_θ and IDM f_ϕ are learned; no FDM is required.

3.4 Rollout-Free Inference

At inference, RWM only encodes the current and goal observations. Following Fig. 2 (c), it directly constructs a sequence of intermediate transition representations between the two encoded endpoints. The shared IDM then recovers the action associated with each consecutive pair: $\hat{a}_m = f_\phi\left(\hat{z}_m^t, \hat{z}_{m+1}^t\right)_{m=t,\ldots,g-1}$.

Because all intermediate representations are constructed directly from the encoded endpoints, the action sequence can be decoded in parallel for open-loop execution. For closed-loop control, RWM re-encodes the current observation after each action chunk and re-constructs the latent plan toward the goal.

Inference thus follows a direct *encode–construct–decode* procedure: the learned representation geometry turns endpoint interpolation into an executable latent plan, requiring neither recursive forward prediction nor action-space search.

4 Experiments

4.1 Experimental Setup

Continuous-control benchmarks. Full training and implementation details of RWM are provided in Appendix A.1. For evaluation, RWM use the same task definitions and evaluation protocol as INTACT variants. We evaluate PushT, Cube, Reacher, and TwoRoom following INTACT Sun et al. (2026). Our comparisons include DINO-WM (Zhou et al., 2025), LeWM (Maes et al., 2026a), Fast-LeWM (Gao & Xu, 2026), Qantara (Rakhimov et al., 2026), PRISM (Wang et al., 2026), C-JEPA (Nam et al., 2026), GC-IDM (Nguyen et al., 2026) and INTACT (Sun et al., 2026). Which are classified as search-based and direct methods. With their correspondingly control type listed in inference column of Table 1.

LIBERO-Goal. We further evaluate the ten LIBERO-Goal tasks. The comparison follows the setting in RC-aux (Li et al., 2026) for LeWM Maes et al. (2026a), RC-aux (Li et al., 2026), and RWM; OpenVLA-OFT 7B (Kim et al., 2025) is included as a large pretrained external reference. Implementation details of RWM specific to the goal-unobserved LIBERO-Goal setting are provided in Appendix A.2.

We evaluate whether RWM can scale to visually complex multi-task manipulation. We report model size (B) and success rate (SR) for latent world models and one external large pretrained reference.

Table 1: **Success rate (%) on continuous-control benchmarks.** Published rows are external references and are not paired controls; † denotes partial task coverage or a distinct reproduction protocol from INTACT Sun et al. (2026).

Class	Method	Inference	PushT	Cube	Reacher	TwoRoom	Macro
Search-based	DINO-WM (Zhou et al., 2025)	CEM	74.0±4.5	86.0±4.7	79.0±5.1	100.0±.0	84.75
	LeWM (Maes et al., 2026a)	CEM 300×(30/10)	**96.0±4.0**	74.0±3.0	86.0±5.0	87.0±2.5	85.75
	Fast-LeWM (Gao & Xu, 2026)	CEM	96.0	80.0	88.0	98.0	90.50
	Fast-LeWM + SC (Gao & Xu, 2026)	CEM + score	98.0	82.0	90.0	98.0	92.00
	Qantara† (Rakhimov et al., 2026)	CEM 300×30, $K = 4$	90.1±1.1	93.7±.7	80.9±1.8	100.0±.0	91.18
	PRISM† (Wang et al., 2026)	MPPI 128×30	89±4	79±6	–	–	–
	C-JEPA† (Nam et al., 2026)	CEM 300×30	88.67	–	–	–	–
	INTACT (Sun et al., 2026)	Pure CEM 300×30	88.44±1.17	68.44±.77	83.67±.67	82.89±.84	80.86±.51
	INTACT (Sun et al., 2026)	Actor-guided CEM 300×30	93.56±.96	96.89±.19	86.67±.88	98.00±1.15	93.78±.77
	INTACT (Sun et al., 2026)	Guarded actor 128×3	92.22±.69	99.78±.19	97.44±.77	98.00±1.15	96.86±.38
Direct	GC-IDM (Nguyen et al., 2026)	Goal-conditioned IDM	84.7±5.0	99.3±1.2	**100.0±.0**	**100.0±.0**	96.00
	INTACT (Sun et al., 2026)	Goal-conditioned IDM	85.78±1.54	100.00±.00	97.67±.00	97.89±1.26	95.33±.58
	RWM (Ours)	Representation plan + IDM	**95.44±0.69**	**100.00±.00**	**97.89±.19**	99.67±.00	**98.25±.14**

4.2 Continuous-Control Results

We follow the INTACT evaluation protocol: each model trained with one of three training seeds is evaluated over three evaluation seeds, and we report the mean and standard deviation across the three training seeds. As shown in Table 1, RWM achieves 95.44%, 100.00%, 97.89%, and 99.67% success rate on PushT, Cube, Reacher, and TwoRoom, respectively, yielding a macro average of $\mathbf{98.25}\%$. RWM outperforms prior search-based methods while requiring no online trajectory search, including the strongest matched INTACT variant (96.86% macro SR) that evaluates 384 candidate trajectories at inference time. RWM also exceed over the goal-conditioned IDM INTACT baseline (95.33%), with the largest gain on PushT ($85.78\% \rightarrow 95.44\%$). These results show that learning executable planing structure into the representation enables high quality search-free control.

4.3 LIBERO-Goal Results

As shown in Table 2, RWM achieves an average success rate of $\mathbf{93.0}\%$ on LIBERO-Goal, substantially outperforming the LeWM (71.2%) and RC-aux (81.2%) baselines. The improvement is particularly clear on challenging tasks; for example, on T5, RWM reaches 90% success compared with 44% for LeWM and 48% for RC-aux. Overall, RWM achieves strong performance across the task suite with a smaller model, demonstrating that the proposed representation-space path construction scales effectively to visually complex robotic manipulation. OpenVLA-OFT achieves a higher average SR of 97.0% with a much larger 7B pretrained model, serving only as an external reference while highlighting the strong potential of RWM.

Table 2: **Per-task success rate on LIBERO-Goal.**

Method	Model size	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	Mean
LeWM + OFT-head	0.07 B	0.64	0.78	0.70	0.76	0.90	0.44	0.60	0.94	0.70	0.66	0.712
RC-aux + OFT-head	0.07B	0.92	0.86	0.80	0.78	0.96	0.48	**0.70**	0.96	0.86	0.80	0.812
RWM (Ours)	**0.03B**	**1.00**	**0.98**	**0.94**	**0.92**	**0.96**	**0.90**	0.64	**1.00**	**1.00**	**0.96**	**0.930**
OpenVLA-OFT	7B	0.98	0.92	0.96	0.86	1.00	1.00	1.00	1.00	1.00	0.98	0.970

4.4 Ablation studies

Our ablations examine if IDM supervision learn informative representation, how replanning frequency affects closed-loop control, and how each training objective contributes to performance. Unless otherwise specified, all ablation models are trained with seed 3072.

Can RWM learn informative representations without explicit representation regularization? LeWM learns its representation through forward-dynamics prediction together with explicit representation regularization, whereas RWM learns its representation through action-grounded and geometric objectives without such explicit representation regularization. We therefore ask whether

RWM nevertheless preserves state- and action-relevant information comparable to that captured by LeWM.

We freeze the trained encoders and train identical tiny MLP probes for one epoch on the same training data. The state probe predicts the simulator configuration from a single encoded representation z, excluding velocities, while the action probe predicts the action from a pair of consecutive representations (z_t, z_{t+1}). We report R^2 on the episode-split validation set.

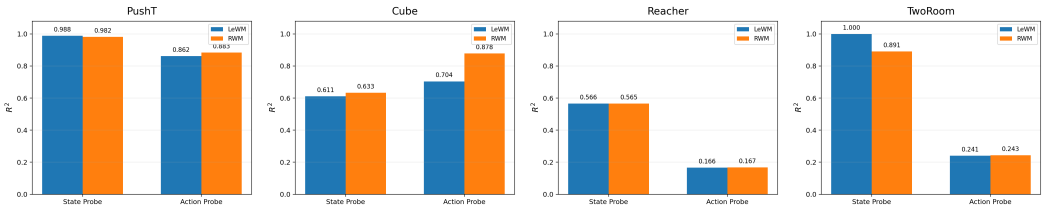


Figure 3: **State and action probing of LeWM and RWM encoder representations.**

As shown in Fig. 3, RWM achieves broadly comparable state and action recoverability to LeWM across the four environments. State information is similarly recoverable on PushT, Cube, and Reacher, although RWM is lower on TwoRoom. For action prediction, RWM matches LeWM on PushT, Reacher, and TwoRoom and substantially improves over it on Cube. These results suggest that RWM can learn representations that preserve much of the physical-state and action-relevant information captured by LeWM, without relying on forward-dynamics prediction or explicit representation regularization.

Do constructed representations preserve state and action information? Having established that the encoded representation z captures meaningful physical information, we next examine whether the constructed representation $\tilde{z}$ preserves the same information when moving off the observed-state manifold.

We freeze the trained RWM encoder and train identical tiny MLP probes for one epoch on the same training data. For state probing, we train the probe only on encoded representations z, and evaluate it on both z and constructed representations $\tilde{z}$, excluding velocities. For action probing, we train only on observed-state pairs (z_t, z_{t+1}) and evaluate action prediction on three pair types: (z_t, z_{t+1}), $(z_t, \tilde{z}_{t+1})$, and $(\tilde{z}_t, \tilde{z}_{t+1})$. We report R^2 on the episode-split validation set.

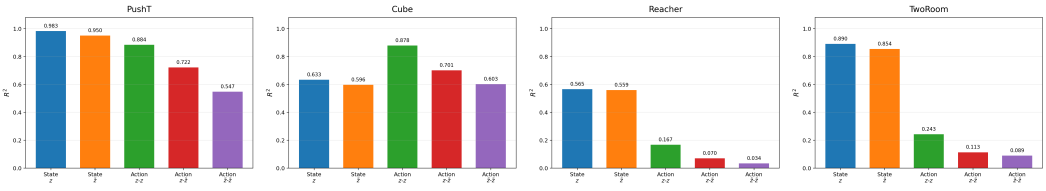


Figure 4: **Information retained by interpolated RWM representations.**

As shown in Fig. 4, physical state remains highly recoverable from $\tilde{z}$, with performance close to that of the encoded representation z across all four environments. Moreover, actions remain recoverable when one or both encoded states are replaced by constructed representations, particularly on PushT and Cube. Although action recoverability decreases as more constructed representations are involved, these results show that $\tilde{z}$ is not merely an arbitrary point between two latent codes: it retains substantial physical-state information and action-relevant structure required for transition execution.

Effect of the closeloop interval. We vary the number of executed actions between two consecutive replanning steps, $h \in \{5, 10, 15, 20, 25\}$, while keeping the trained model fixed. As shown in Fig. 5, more frequent replanning generally improves control performance. RWM achieves the

highest macro-average SR of 98.33% at $h = 5$, which we use as the default setting. Performance remains strong for moderate intervals, but degrades as the execution becomes increasingly open-loop, with the macro-average SR dropping to 69.17% at $h = 25$. The degradation is mainly driven by PushT and Reacher, whereas Cube and TwoRoom remain comparatively robust to longer execution horizons. This result highlights the importance of periodically incorporating new observations to correct accumulated execution errors.

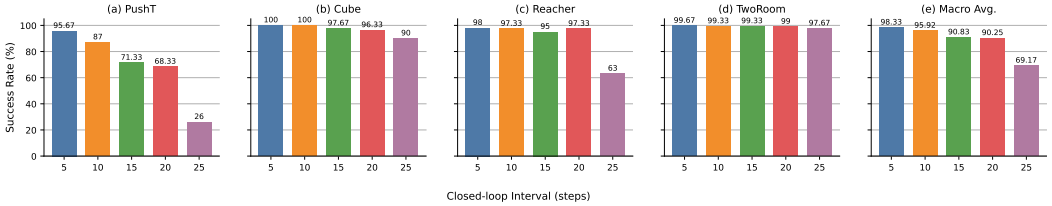


Figure 5: **Effect of the closed-loop interval.**

Role of the training objectives. We ablate the three training objectives to examine their distinct roles in RWM. As shown in Table 3, $\mathcal{L}^{\text{int}}_{\text{IDM}}$ is critical for control: removing it reduces PushT SR from 96.00% to 39.37%. In contrast, all variants containing this term achieve similar SR (95.23–96.17%), indicating that action grounding along the constructed latent path is the primary learning signal that makes representation-space planning executable.

The other objectives provide complementary constraints on the learned representation. $\mathcal{L}^{\text{enc}}_{\text{IDM}}$ improves physical-state recoverability, increasing state-probe R^2 from 0.95 to 0.98, while $\mathcal{L}_{\text{consist}}$ geometrically aligns constructed representations with their observation-induced counterparts. We therefore use the full objective by default, which preserves these complementary properties while achieving SR comparable to the best ablated variant.

Table 3: **Component ablation of RWM.**

Loss=	$\mathcal{L}^{\text{int}}_{\text{IDM}}$	$\mathcal{L}^{\text{enc}}_{\text{IDM}}$	$\mathcal{L}_{\text{consist}}$	PushT SR ↑
$\mathcal{L}^{\text{int}}_{\text{IDM}}$	✓			95.56
$\mathcal{L}^{\text{enc}}_{\text{IDM}} + \mathcal{L}_{\text{consist}}$		✓	✓	39.37
$\mathcal{L}^{\text{int}}_{\text{IDM}} + \mathcal{L}^{\text{enc}}_{\text{IDM}}$	✓	✓		96.17
$\mathcal{L}^{\text{int}}_{\text{IDM}} + \mathcal{L}_{\text{consist}}$	✓		✓	95.23
$\mathcal{L}^{\text{int}}_{\text{IDM}} + \mathcal{L}^{\text{enc}}_{\text{IDM}} + \mathcal{L}_{\text{consist}}$	✓	✓	✓	96.00

5 Conclusion

We introduced the *Representation World Model* (RWM), which learns a representation space in which executable plans can be directly constructed between states. During training, action-grounded supervision shapes endpoint-constructed latent paths into executable plans. At inference, RWM directly constructs and decodes such a path between the current and goal states, without recursive dynamics rollout or action-space search. Across continuous-control benchmarks, RWM achieves strong search-free control while preserving informative state and action structure in both observed and constructed representations. Besides, RWM also demonstrate high potential on more complex embodied control tasks. These results suggest that representation can itself learns both high quality action-relevant states and executable plans.

However, the current implementation constructs a straight latent path between endpoint representations, without supporting of multiple options. Although close-loop operation can lead to different new trajectories for multiple options, this variation arises from feedback rather than explicit modeling of multiple options. Future work will explores to address this limitation.

AI USE STATEMENT

In this work, we used generative AI tools for **none of the required disclosure**. We have not used generative AI tools for **all of the required disclosure**, and **all of the required disclosure** are not applicable to this work. Additionally, we used generative AI tools for **paper language polishing**. We have reviewed all AI-assisted work. We input paper sentences to AI tools to correct the grammar and refine the writing to be standard and easy to read. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024.

Jack Boylan and Chris Hokamp. No gaussian required: Contrastive inverse dynamics for jepa world models. *arXiv preprint arXiv:2608.17542*, 2026.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607, 2020.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

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

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

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: Rc-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026.

Chang Liu, Fei Suo, Yanzhou Jin, Yusuke Iwasawa, Yutaka Matsuo, and Yaonan Zhu. Temporally centered sigreg improves multi-task leworldmodel learning: From analysis to method. *arXiv preprint arXiv:2607.26924*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026a.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026b.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent masking. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *arXiv preprint arXiv:2605.08732*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Ruslan Rakhimov, George Bredis, Yuriy Maksyuta, and Daniil Gavrilov. Qantara: Bridge-flow training for multi-paradigm JEPA control. *arXiv preprint arXiv:2607.04978*, 2026.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020.

Junhan Sun, Hao Zhao, and Guofeng Zhang. Intact: Isomorphic intent-to-action learning for search-free world models. *arXiv preprint arXiv:2607.26056*, 2026.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided imagination sampling in world models. *arXiv preprint arXiv:2606.07974*, 2026.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 12310–12320, 2021.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

A Appendix

A.1 Training details

We follow the data preprocessing and training protocol of LeWM (Maes et al., 2026a) unless otherwise specified. All observations are resized to 224×224 pixels. We apply a frame skip of 5, grouping the consecutive low-level actions between two observations into a single action block. We use a $90\%/10\%$ train–validation sample-level split.

Architecture. The visual encoder is a ViT-Tiny with patch size 14, trained from scratch. Following the encoder, RWM constructs latent paths as described in Sec. 3. The inverse dynamics model (IDM) is implemented as a three-layer MLP. The encoder and IDM are optimized jointly, with action supervision applied to both encoded and constructed representations. Unlike rollout-based world models, RWM does not train an explicit forward dynamics predictor.

Optimization. We train all models using AdamW with a learning rate of 5×10^{-4} and weight decay of 10^{-3} (Reacher is with 5×10^{-5}). We use a linear-warmup cosine learning-rate schedule, a batch size of 128, bfloat16 mixed precision, and gradient clipping with a maximum norm of 1.0. Unless otherwise stated, models are trained for 10 epochs. For the ablation studies, we use training seed 3072. The complete training configuration is summarized in Table 4.

A.2 Implementation Details on Libero-Goal Test

LIBERO-Goal differs from the continuous-control benchmarks in that the goal observation is not available at test time. Therefore, the target latent z_g cannot be directly encoded and used to construct a goal-conditioned latent path as in the standard RWM setting. To handle this setting, we introduce an additional *delta predictor* that predicts the latent displacement used to construct the next representation.

During training, the full demonstration trajectory remains available. We therefore use future states from the demonstrated trajectory to define the target latent displacements and supervise the delta predictor. At test time, the predicted latent delta replaces the unavailable goal-conditioned latent

Table 4: **Training configuration for RWM.** We follow the LeWM training protocol wherever applicable and modify only the model components and objectives required by RWM.

Setting	Value
Input resolution	224×224
Frame skip	5
Train / validation split	$90\%/10\%$
Encoder network	ViT-Tiny/14
IDM network	3-layer MLP
IDM observation history	3
Optimizer	AdamW
Learning rate	5×10^{-4}
Weight decay	10^{-3}
LR schedule	Linear warmup + cosine decay
Batch size	128
Training epochs	10
Precision	bfloat16
Gradient clipping	1.0
Training seed	0,42,3072
Training GPU	RTX5090 $\times$ 8

displacement, allows RWM to iteratively construct the latent path without access to an explicit goal representation z_g.

For LIBERO-Goal, we use a ViT-Small visual encoder. The visual representation is fused with the robot proprioceptive state before being passed to the downstream modules. Since LIBERO contains multiple manipulation tasks with different goal semantics, we additionally provide the task ID as conditioning information to both the delta predictor and the inverse dynamics model (IDM). The delta predictor constructs the latent path, while the IDM decodes each consecutive latents into the corresponding robot action.

A.3 Additional Physical-State Probing Visualizations

We further provide qualitative visualizations of the physical states decoded from the learned representations in Fig. 6,7,8,9. For each task, we apply the corresponding state probe to encoded representations z and constructed interpolated representations $\tilde{z}$, and render the predicted physical states in the simulator. Black borders denote states decoded from encoded representations z, while green borders denote states decoded from interpolated representations $\tilde{z}$. The decoded configurations from $\tilde{z}$ evolve consistently between encoded states, showing that the constructed representations preserve substantial physical-state information.

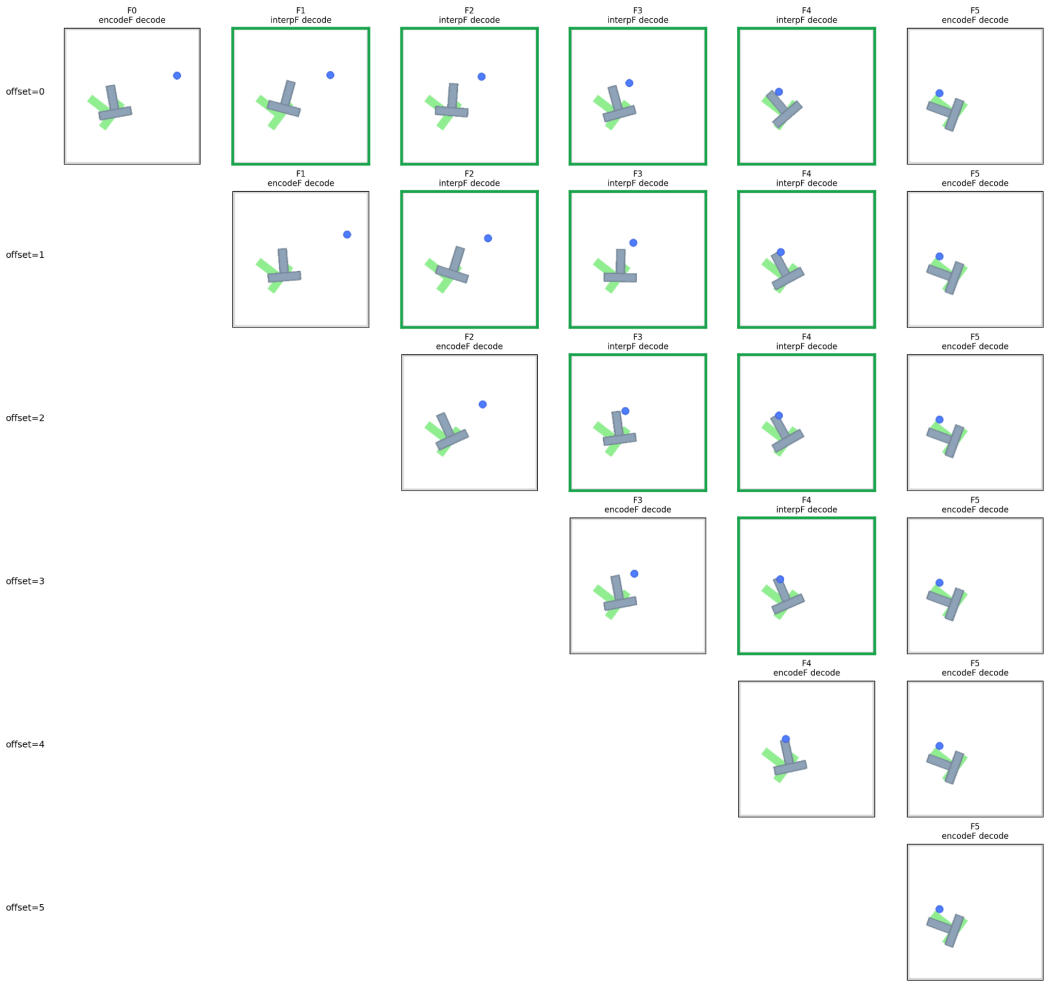


Figure 6: **Physical-state decoding on PushT.** Black-bordered images correspond to physical states decoded from encoded representations z, while green-bordered images correspond to states decoded from interpolated representations $\tilde{z}$.

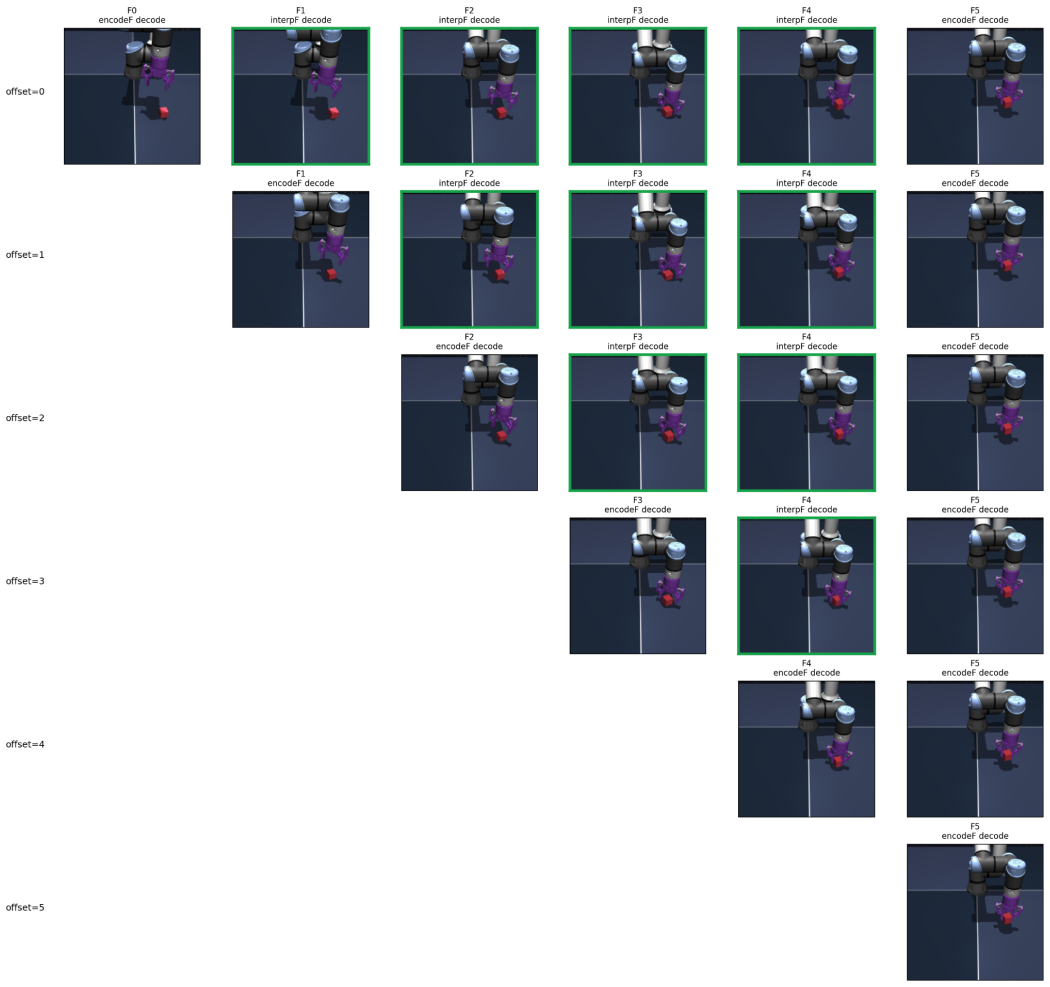


Figure 7: **Physical-state decoding on Cube.** Black-bordered images correspond to physical states decoded from encoded representations z, while green-bordered images correspond to states decoded from interpolated representations $\tilde{z}$.



Figure 8: **Physical-state decoding on Reacher.** Black-bordered images correspond to physical states decoded from encoded representations z, while green-bordered images correspond to states decoded from interpolated representations $\tilde{z}$.

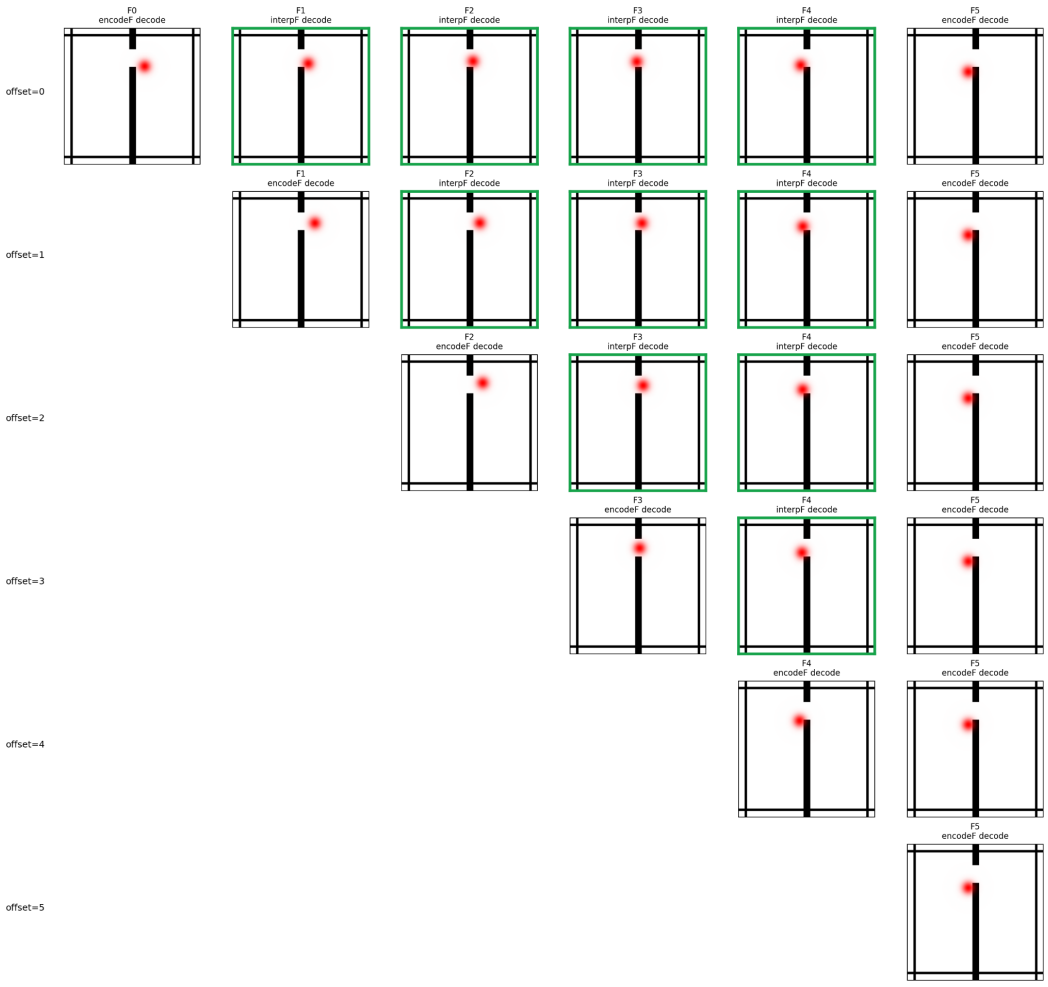


Figure 9: **Physical-state decoding on TwoRoom.** Black-bordered images correspond to physical states decoded from encoded representations z, while green-bordered images correspond to states decoded from interpolated representations $\tilde{z}$.