CLAW: CONTINUOUS LATENT ACTION WORLD MODELS VIA ADVERSARIAL REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce CLAW, a fully end-to-end self-supervised framework for learning a world model jointly with continuous latent action representations directly from action-free videos. Our approach leverages adversarial latent regularization to prevent the latent from encoding future observations, so that it captures the structure of the action. The diffusion-based world model then models environment dynamics as a distribution over future observations rather than as a single deterministic rollout. Both components are learned from observation sequences alone without any action annotations. By simultaneously training the Latent Action Model and world model, CLAW learns to reason about how inferred actions induce environment transitions from visual observations alone. We show that the resulting latent action world model supports both imitation learning from observation and goal-directed planning. In imitation learning, latent actions extracted from raw videos enable behavior cloning. For planning, CLAW generates sequences of latent actions and maps them to executable actions to reach desired goals. Extensive experiments across diverse tasks and embodiments demonstrate that CLAW produces semantically meaningful latent action representations, supports effective action transfer, and enables planning and imitation from observation, outperforming existing methods.

1 INTRODUCTION

Models that predict how actions affect the state of an environment have long been fundamental to planning (Bertsekas, 2012), control (Tassa et al., 2012), and learning (Sutton, 1990; 1991; Atkeson & Santamaría, 1997; Deisenroth & Rasmussen, 2011). By modeling the consequences of actions, agents can evaluate candidate behaviors, optimize policies through imagined experience, and adapt to new tasks with improved data efficiency. Inspired by human cognition, renewed attention has focused on "world models" (Conant & Ross Ashby, 1970; Richalet et al., 1978; Lozano-Perez, 1983; Bryson, 2018; Ha & Schmidhuber, 2018) as forward-predictive models that capture how an environment evolves in response to the behavior of an interacting agent (Clark, 2013; Bubic et al., 2010; Hafner et al., 2019; 2020; Schrittwieser et al., 2020; Hafner et al., 2024). Contemporary world models often condition their predictions on explicit action labels (Hafner et al., 2019; Hu et al., 2023; Bar et al., 2025). However, action representations are often specific to a particular agent or action space, limiting transfer across agents with different embodiments or action spaces (Gao et al., 2025; Garrido et al., 2026; Huang et al., 2026). Moreover, requiring action labels prevents learning at scale from the vast amounts of unlabeled video data available online (Ye et al., 2025).

A world model that instead reasons over learned latent action representations is not tied to a particular agent's action space, allowing it to be trained and applied across tasks and embodiments without explicit action labels (Bu et al., 2025). Recent methods train such models from visual observations alone (Bruce et al., 2024; Schmidt & Jiang, 2024; Chen et al., 2024; Gao et al., 2025; Ye et al., 2025). However, it remains difficult to learn latent actions that are semantically meaningful, i.e., structured so that transitions with similar effects map to nearby latents across contexts and embodiments. This structure is what ultimately allows the latent actions to be grounded to a robot's executable actions from only a small amount of supervision (Zhang et al., 2026). Another challenge is the tendency for the latent to either *leak* (Lee et al., 2026) content from the future observation, letting the world model take a shortcut that bypasses learning action semantics, or *collapse* (Zhang et al., 2026; Wang et al., 2025) to an uninformative representation (Bruce et al., 2024; Nikulin et al., 2025).

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.

To address these challenges, we introduce CLAW, a fully self-supervised framework that jointly learns continuous latent action representations and a predictive world model directly from raw, action-free video. Our approach couples a Latent Action Model (LAM) with the world model through reciprocal supervision: the LAM infers compact representations of observed transitions that condition future prediction, while the world model provides the learning signal that shapes these representations. We further introduce adversarial latent regularization to prevent the latent actions from encoding future observations directly, encouraging them instead to capture the underlying structure of actions. Unlike prior continuous latent action methods, which use future prediction primarily to shape the latent representation and discard or freeze the predictive model afterward, CLAW retains the world model as a component of the learned system.

Once trained, the learned latent action world model supports two complementary forms of agent learning and decision making. First, the learned latent action representations provide supervisory signals for policy learning from action-free videos, enabling both imitation learning from observation (ILfO) (Torabi et al., 2019) and latent action pretraining for downstream policy adaptation. Second, the learned world model enables goal-directed planning by predicting the outcomes of candidate latent action sequences, allowing an agent to optimize its behavior toward specified visual goals (Mendonca et al., 2023; Ebert et al., 2018).

We evaluate our method across diverse environments, tasks, and agent embodiments, spanning simulated and real-world manipulation, open-world gameplay, and procedurally generated environments. We assess the learned latent action representations through ILfO, latent action policy pretraining, nearest-neighbor action retrieval, and action transfer (Gao et al., 2025; Bruce et al., 2024). We further evaluate the learned dynamics through goal-directed visual planning and controllability under latent action conditioning. These evaluations show that CLAW learns structured and transferable latent actions that support effective policy learning and planning, outperforming existing approaches across a broad range of downstream tasks.

2 Related Work

Model-based Learning and World Models World models encompass a broad class of models that predict the consequences of an agent's actions (Ha & Schmidhuber, 2018). Such models are used extensively in model-based reinforcement learning (Williams et al., 2017; Nagabandi et al., 2017; Hafner et al., 2020; 2022; 2024; Janner et al., 2021) and planning (Hafner et al., 2019; Sekar et al., 2020; Micheli et al., 2023; Schrittwieser et al., 2020). World models learn dynamics from image-based inputs, which capture fine details and don't require hand-designed state representations (Alonso et al., 2024; Bai et al., 2025; Assran et al., 2025; Bar et al., 2025; Wu et al., 2024; Zhou et al., 2025). Typically, these models are trained with ground-truth actions, conditioning future predictions on known control inputs to model environment dynamics and enable downstream planning or policy learning (Bar et al., 2025; Zhou et al., 2025; Alonso et al., 2024; Wang et al., 2026).

Continuous Latent Action Learning Recent work has explored continuous latent action representations as an alternative to discrete latent action spaces. CLAM (Liang et al., 2025) trains a latent inverse dynamics model by deterministic next-observation reconstruction, but relies on an action decoder trained on action-labeled play data to keep the latent decodable. UniSkill (Kim et al., 2025a) learns skill representations from future-frame prediction without action labels, but imposes an information bottleneck and builds on pretrained components, including InstructPix2Pix initialization and a frozen DepthAnythingV2 depth estimator. CoMo (Yang et al., 2025) combines three mechanisms: early temporal feature differencing, a temporal contrastive objective focused on foreground motion, and a constrained latent dimensionality. Each of these methods restricts what the latent can represent or supervises it with action labels, and trains the representation separately from the policy that consumes it. CLAW instead removes future-observation information adversarially while retaining a continuous latent space without an explicit information bottleneck, and trains the inverse and forward models simultaneously.

Latent Action World Models Latent action world models infer actions from observation sequences alone, enabling training with unlabeled video and promising transferability across embodiments and contexts (Gao et al., 2025; Bruce et al., 2024; Nikulin et al., 2025; Ye et al., 2025). Latent action world models typically use an inverse dynamics model that encodes a latent action representation based on observation sequences, and a forward dynamics model (FDM) that predicts the future state

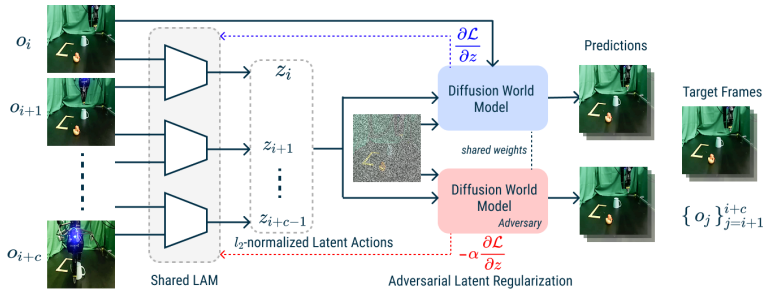


Figure 1: Overview of the CLAW framework. The LAM encodes compact, normalized action representations (left). The adversarial latent regularization employs two diffusion world models with shared weights, with only the first receiving the initial frame alongside the action (right).

conditioned on the latent action and past observations (Schmidt & Jiang, 2024; Bu et al., 2025; Cui et al., 2024). Often, latent action world models involve pretraining an action-agnostic FDM and only then introduce inferred actions in a second fine-tuning stage (Wu et al., 2024; Mendonca et al., 2023; NVIDIA et al., 2025; Che et al., 2024; He et al., 2025). Instead, we build upon research on 'action-aware pretraining' that learns the world model and action representation simultaneously, which has been shown to improve downstream controllability and transfer (Bruce et al., 2024).

A core challenge in training latent action world models is that the action representation will often embed information about future observations instead of encoding information about the action alone, especially with image-based observations (Nikulin et al., 2025). Prior approaches typically regulate information leakage through explicit bottlenecks, most commonly via quantization (Bruce et al., 2024; Ye et al., 2025; Bu et al., 2025), variational constraints (Gao et al., 2025), or capacity-limited continuous latents (Yang et al., 2025). However, recent work has shown that continuous actions facilitate finer-grained control and often stronger performance than discretization (Physical Intelligence et al., 2025; Yang et al., 2025; Kim et al., 2025b). Alternatively, other works introduce supervision through a small amount of action-labeled data (Nikulin et al., 2025; Yang et al., 2025), which reduces leakage but requires some action annotations. In contrast, our approach applies adversarial latent regularization to discourage encoding of future visual information. By jointly training the LAM and world model, we learn representations that capture both action structure and environment dynamics without relying on quantization or action supervision.

3 Latent Action World Model

An action-conditioned world model (from now on, "world model") approximates observation transition dynamics $P(o_{i+1} \mid o_i, a_i)$, predicting future observations o from past observations and actions a. Unlike standard world model formulations, latent action world models do not assume access to action annotations or the ability to sample from $\mathcal{A}$ during training. Instead, we introduce a learned latent variable z_i that captures the effect of the underlying action on the observation dynamics. We then model these dynamics as $P(o_{i+1} \mid o_i, z_i)$ and learn the world model directly from observation sequences. The training data is an action-free video dataset $\mathcal{V} = \{o_{1:T}\}^N$, where each video $o_{1:T} = \{o_1, \ldots, o_T\}$ is a sequence of RGB frames $o_i \in \mathbb{R}^{W \times H \times 3}$. Videos vary in length (T) and comprise diverse, unlabeled trajectories. These videos may come, for example, from gameplay recordings or videos of robotic play (Lynch et al., 2020).

Figure 1 provides an overview of our proposed architecture and training setup. Similar to conventional inverse dynamics models (IDMs) (Edwards et al., 2019; Schmidt & Jiang, 2024), the Latent Action Model (LAM) is designed to infer the action responsible for the transition between consecutive observations. We use a ViT backbone (Dosovitskiy, 2020) to map consecutive observations, o_i, o_{i+1}, into a continuous latent action representation $z_i \in \mathbb{R}^n$ (Fig. 5(a)): $z_i = \text{LAM}_\theta(o_i, o_{i+1})$. The predicted latent actions are constrained to lie on the unit sphere through ℓ_2 normalization.

CLAW subsequently projects the LAM output corresponding to the action token through an MLP to yield the continuous latent action representation, which conditions the world model and captures

the action responsible for the observed transition. The LAM is trained end-to-end with the forward dynamics world model, described next.

3.1 Diffusion-based World Model

We represent the dynamics as a diffusion-based world model, implemented with a UViT backbone (Bao et al., 2023). Given an initial observation frame o_i and the sequence of latent actions $\{z_j\}_{j=i}^{i+c}$ inferred by the LAM, it predicts the corresponding future frames $\{o_j\}_{j=i+1}^{i+c+1}$, where c defines the horizon. By conditioning on both the initial frame and inferred latent actions, the world model learns to use the inverse dynamics learned by the LAM to produce more accurate predictions.

The forward diffusion process follows a variance-preserving stochastic differential equation that progressively adds Gaussian noise to the target frames. The diffusion backbone operates on image patches within a transformer architecture. We structure the input as $\left[t_\tau\ \{z_j\}_{j=i}^{c}\ o_i\ \{x_j\}_{j=i}^{c}\right]$, where t_τ is the diffusion timestep embedding, $\{z_j\}_{j=i}^{c} \in \mathbb{R}^{c\times d}$ are latent action tokens for c transitions from the LAM, o_i is the set of conditioning frame patches, and x_j are patches from the noised observations o_j with $i < j \leq i + c$ generated by the forward diffusion process $q(o_t^j \mid o_0^j)$. Latent action tokens are projected to match the transformer embedding dimension. To maintain causal consistency and prevent information leakage, we use a modified causal attention mask that blocks noisy target tokens from attending to future action tokens or observations beyond their timestep (depicted in Figure 5(b)).

Training optimizes a score network $s_\phi(x^t, t, z^t, o_i)$ to predict the noise ϵ added during the forward diffusion process.

$$\mathcal{L} = \mathbb{E}_{t\sim\mathcal{U}(0,1),\,\epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\epsilon - s_\phi(x^t, t, z^t, o_i)\right\|^2\right], \tag{1}$$

where the noisy samples are $x^t = \sqrt{\alpha(t)}\,x^0 + \sqrt{1-\alpha(t)}\,\epsilon$. At each step, the model conditions on the latent action representation z^t and the reference frame o_i to iteratively denoise the sample.

3.2 Adversarial Information Regularization

The LAM and world model are trained together end-to-end to predict future frames. Unfortunately, this can lead to two kinds of degenerate solutions. The latent can collapse onto $z_i = o_{i+1}$, in which case the world model predicts accurately but learns no action semantics; or it can become uninformative, in which case the world model ignores it and predicts from o_i alone. Adversarial latent regularization targets the first, and its gradual schedule avoids inducing the second.

In our approach, the LAM is coupled to the world model through two prediction pathways that share the same latent action representation z_i, as shown in Figure 1. The first conditions the world model on both o_i and z_i, encouraging z_i to encode prediction-relevant information. The second provides only z_i. Since accurate predictions of future observations in this second branch can only occur if information about o_{i+1} leaks into z_i, we *penalize* accurate predictions by inserting a gradient reversal layer (GRL) (Ganin et al., 2016) between z_i and the world model. The GRL scales and reverses the gradient during backpropagation as $\frac{\partial\mathcal{L}}{\partial z} \leftarrow -\delta\frac{\partial\mathcal{L}}{\partial z}$; this *adversarial latent regularization* encourages the LAM to avoid encoding information about future observations into the latent action z_i.

During training, the strength of the adversarial signal is controlled by a gradually increasing weight δ, following a sigmoidal schedule

$$\delta = \min\left(\frac{2}{1+\exp(-\gamma p)} - 1, 0.999\right) \quad \text{where} \quad p = \frac{\text{step}}{n_{\text{steps}}} \tag{2}$$

where $\gamma = 10$ for all experiments as in Ganin et al. (2016), and n_{steps} is the total number of training steps. The schedule lets the model first learn to predict accurately, then progressively increases the pressure on the LAM to remove future-observation information, with δ capped at 0.999. Starting from $\delta \approx 0$ also guards against the second failure mode, since the world model learns to depend on the latent before any adversarial pressure is applied.

4 Experiments

We evaluate CLAW in terms of (1) the quality and structure of the learned latent action space, and (2) goal-directed planning using the latent action world model. Following prior work (Gao et al., 2025;

Bruce et al., 2024), we train all latent action world models solely on task-agnostic interaction data, collected by an agent exploring the environment without explicit objectives (Table 8).

4.1 Baselines

We compare CLAW against AdaWorld (Gao et al., 2025), LAPO (Schmidt & Jiang, 2024), and CoMo (Yang et al., 2025), all of which learn latent actions from visual observations without ground-truth action labels. These baselines span continuous variational latent actions (AdaWorld), discrete vector-quantized latent actions (LAPO), and continuous contrastively learned latent motion (CoMo). We exclude CLAM (Liang et al., 2025), whose latent representation is shaped by action-labeled play data, and UniSkill (Kim et al., 2025a), which relies on pretrained perceptual components. Genie (Bruce et al., 2024) is not publicly available, so LAPO serves as the discrete vector-quantized method. LAPA (Ye et al., 2025) is a larger-scale variant of a similar latent action learning framework that also uses vector quantization to learn discrete latent actions and supports downstream policy finetuning with action-labeled data. In addition to these baselines, we report CLAW w/o Adv, an ablation of our method trained with the adversarial weight set to $\delta = 0$, so that the second prediction pathway continues to train the world model but provides no gradient signal to the LAM.

4.2 Benchmarks and Environments

We assess CLAW across 14 tasks in 5 simulated environments and 2 real-world robotic tasks. **VP**^{2} (Tian et al., 2023) is a benchmark for visual planning in tabletop manipulation based on the Robosuite (Zhu et al., 2020) and RoboDesk (Kannan et al., 2021) environments. **OGBench** (Park et al., 2025) is a benchmark in which a UR5e arm manipulates articulated objects; we evaluate `Open Drawer`, `Close Drawer`, and `Push Button`. **Crafter** (Hafner, 2021) is a 2D open-world survival environment with procedurally generated worlds, where we evaluate a long-horizon task (`Greedy Wood Collection`) and a shorter goal-conditioned one (`Single Wood Collection`). **Procgen** (Cobbe et al., 2019) is a suite of procedurally generated environments; we evaluate visual planning on Climber, Jumper, and Ninja using MPC-CEM (De Boer et al., 2005). Environment configurations, data generation, and task definitions are in Appendix C.

Real-world tabletop manipulation. We evaluate latent action pretraining on a 6-DOF UR5 robot, training the latent action world model on Something-Something V2 (Goyal et al., 2017), a dataset of humans interacting with household items, without access to robot actions or demonstrations. In `Recycle Bottle`, a randomly positioned bottle must be placed in the correctly colored bin. In `Push Ramen`, a ramen package and distractor objects are placed at randomized positions. Performance uses a staged progress metric (Kress-Gazit et al., 2024) normalized to $[0, 1]$ and defined in Appendix C.5.

4.3 Evaluating the Learned Latent Action Representation

4.3.1 Latent Action Policy Learning

First, we evaluate our learned latent action representations in the Imitation Learning from Observations (ILfO) setting Bruce et al. (2024), where the agent learns to perform downstream tasks from expert demonstrations without access to the expert's actions. To enable imitation from action-free videos, we extract latent actions z_i from expert videos using the trained LAM. These latent actions serve as targets for training a behavior cloning policy $\pi(z_i \mid o_i)$. During inference, predicted latent actions are mapped to executable actions by nearest-neighbor retrieval against a small set of action-annotated reference videos.

Results As shown in Table 1, the CLAW-based latent action policy outperforms all baselines across both Crafter tasks, achieving higher success rates and collecting substantially more wood on average in the `Greedy Wood Collection` task. On OGBench, CLAW achieves the highest success rate on `Open Drawer` and `Close Drawer`, while AdaWorld performs slightly better on `Push Button` (16.5% vs. 15.0% for CLAW, n=200). Notably, despite being pretrained only on general survival videos without explicit supervision for wood harvesting, the learned latent actions still enable effective wood collection in Crafter, indicating disentanglement of action representations from task-specific rewards while preserving meaningful signals for imitation.

Table 1: Results from latent action policy learning evaluations on Crafter and OGBench. Each policy is evaluated over 200 random environment seeds per task.

	Crafter			OGBench		
Method	Wood Harvest (%)	Avg. Wood	Wood Acquire (%)	Open Drawer (%)	Close Drawer (%)	Push Button (%)
Random	25.0	0.28	27.5	2.5	14.5	8.0
LAPO	35.5	1.12	38.0	4.5	39.5	9.5
AdaWorld	43.5	1.60	46.5	7.5	29.0	**16.5**
CoMo	80.5	3.01	79.5	9.0	34.0	11.5
CLAW	**86.0**	**4.46**	**84.0**	**13.5**	**42.0**	15.0

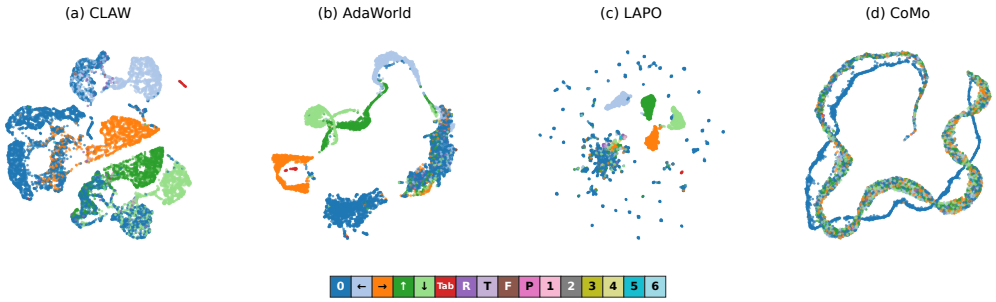


Figure 2: Latent action structure on Crafter. Latent actions from matched observation transitions are projected independently with UMAP for each method and colored by ground-truth Crafter actions, which are not used during training. Legend chips show the corresponding Crafter keyboard controls (arrows for movement; `0` denotes no-op).

4.3.2 Real-World Latent Action Pretraining

Following the experimental protocol of Ye et al. (2025), we evaluate whether latent actions inferred by the LAM can substitute for ground-truth actions as a pretraining signal for real-world robot policies. We pretrain an ACT policy (Zhao et al., 2023) to predict our inferred latent actions from visual observations, then freeze its vision encoder and finetune the remaining components on ground-truth actions for each task. We compare against the same architecture trained on ground-truth actions alone following the standard setup of Zhao et al. (2023) (`Scratch`). Task definitions, the staged progress metric, and the demonstration protocol are in Appendix C.5.

Results Table 3 reports mean normalized progress over 20 trials per task. CLAW achieves the highest score on both tasks, with the clearest margin on the harder `Push Ramen` task, where it improves over `Scratch` by 0.20. AdaWorld and CoMo do not exceed `Scratch` on either task, suggesting that effective latent-action pretraining depends on learning useful action structure

4.3.3 Latent Action Structure

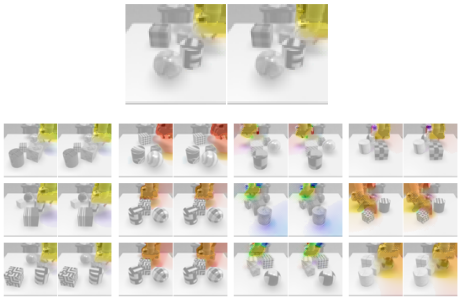
We evaluate whether latent representations learned without action supervision recover structure aligned with the underlying action space. Using the same set of Crafter transitions for all methods, we extract latent actions with each pretrained model and independently project the resulting representations into two dimensions using UMAP. Ground-truth Crafter actions are used only to color points for visualization and are never provided during latent action learning.

Table 2: Visual planning on Crafter and Procgen.

Method	Crafter	Jumper	Climber	Ninja
LAPO	46.67%	58.33%	53.33%	48.33%
AdaWorld	53.33%	56.67%	**66.67%**	46.67%
CoMo	61.67%	38.33%	45.00%	31.66%
CLAW w/o Adv	26.67%	35.00%	38.33%	25.00%
CLAW	**68.33%**	**63.33%**	60.00%	**56.67%**

Table 3: Real-world manipulation results.

Method	Push Ramen	Recycle Bottle
Scratch	0.65	0.85
LAPA	0.73	0.83
AdaWorld	0.50	0.80
CoMo	0.48	0.53
CLAW	**0.85**	**0.88**



(a) CLAW (b) AdaWorld (c) LAPO (d) CoMo

Figure 3: **Closest Latent Action retrieval via L_2 nearest-neighbor:** (Top) Query transition. (Bottom) Top-3 neighbors retrieved via L_2 search in the latent space of CLAW, AdaWorld, LAPO, and CoMo. All frames are shown as grayscale with the optical flow ($t \rightarrow t{+}1$) overlaid using the Middlebury color wheel.

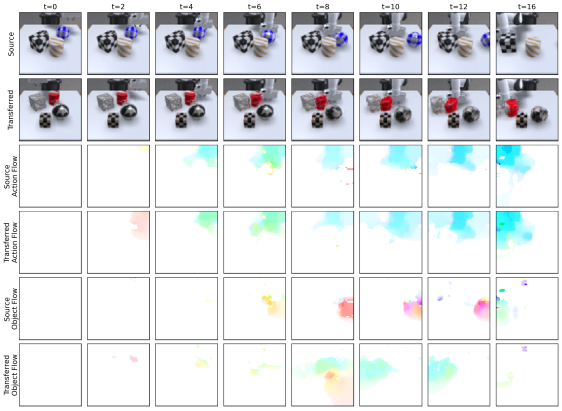


Figure 4: **Latent action transfer via action swap with optical-flow validation.** Row 1: source trajectory. Row 2: transferred rollout from a random target initial frame using source latent actions. Rows 3–4: source and transferred action (robot's gripper) optical flow. Rows 5–6: source and transferred object flow.

As shown in Figure 2, the methods exhibit qualitatively different latent geometries. CLAW organizes transitions into locally action-consistent regions, with several behaviors, including movement directions, forming distinguishable regions in the embedding. Some overlap between action labels is expected, because the recorded Crafter action is the command the player issued, not the transition it produced. A blocked movement or a failed craft carries a non-noop label but produces almost no visual change, while a `Noop` can coincide with visible change and land far from the centroid of the no-op cluster. Discrete labels and observable transition effects are therefore related but not equivalent, and we quantify both in Appendix D.1.2.

4.3.4 Closest Action Retrieval

Finally, we qualitatively evaluate the structure of the learned latent action space via L_2 nearest-neighbor retrieval. For each query, we sample a pair of consecutive frames representing an action transition and encode it using the LAM. We then retrieve the n nearest latent actions from the dataset under the L_2 metric and visualize the corresponding frame transitions. Qualitative inspection of the retrieved neighbors allows us to assess whether the latent space organizes semantically similar actions into compact, consistent neighborhoods, reflecting the degree to which the learned representations capture meaningful behavioral structure rather than low-level visual similarity.

Results As shown in Figure 3, across queries, the nearest neighbors in CLAW's learned latent action space exhibit similar end-effector motion, whereas non-action information like object identity varies substantially. This is in contrast to baseline methods, where nearest-neighbor retrieval reveals that the inferred latent actions confuse end-effector motion with other non-action scene properties. See Appendix D.2 for further explanation and more examples in Robosuite and other environments.

4.4 Evaluating the Dynamics Model

4.4.1 Visual Planning

Here, we use standard model predictive control (MPC) to plan sequences of latent actions through the world model toward a goal observation. Each predicted latent action is mapped to a true action by retrieving its nearest neighbor from a small set of annotated reference videos. We evaluate on goal-reaching success when executing the plan via the nearest-neighbor actions.

Results Table 4 reports visual planning success rates on the VP^2 benchmark across Robosuite and RoboDesk tasks. CLAW achieves the highest success rate on four of the six tasks. However, CLAW has lower success rates on the Green and Blue Button tasks. We hypothesize that these tasks are

Table 4: Visual planning evaluation results on Robosuite and RoboDesk tasks (mean ± standard error of the mean (SEM) over three control runs).

Method	Push	Open Slide	Blue Button	Green Button	Red Button	Upright Block
Random	$0.00 \pm 0.00\%$	$0.00 \pm 0.00\%$	$10.0 \pm 2.72\%$	$1.11 \pm 1.57\%$	$1.11 \pm 1.57\%$	$0.00 \pm 0.00\%$
LAPO	$18.33 \pm 1.89\%$	$11.67 \pm 3.40\%$	$8.89 \pm 2.40\%$	$6.67 \pm 1.83\%$	$3.33 \pm 2.27\%$	$15.56 \pm 4.54\%$
AdaWorld*	$\mathbf{64.60 \pm 1.52\%}$	$5.83 \pm 2.85\%$	$\mathbf{29.17 \pm 2.50\%}$	$\mathbf{10.83 \pm 2.50\%}$	$10.00 \pm 2.36\%$	$5.00 \pm 0.96\%$
CoMo	$33.43 \pm 2.25\%$	$11.10 \pm 0.91\%$	$16.67 \pm 3.14\%$	$3.33 \pm 1.92\%$	$11.10 \pm 0.91\%$	$\mathbf{38.89 \pm 0.91\%}$
CLAW w/o Adv	$16.98 \pm 1.35\%$	$6.37 \pm 2.27\%$	$8.33 \pm 1.44\%$	$6.67 \pm 1.57\%$	$3.33 \pm 1.57\%$	$2.22 \pm 1.81\%$
CLAW	$\mathbf{63.70 \pm 1.71\%}$	$\mathbf{23.33 \pm 5.67\%}$	$12.22 \pm 2.40\%$	$\mathbf{11.11 \pm 0.91\%}$	$\mathbf{25.56 \pm 0.91\%}$	$\mathbf{43.33 \pm 4.71\%}$

* Results reported from the original paper Gao et al. (2025).

Table 5: Latent action controllability and future-observation leakage. Higher Δ_tPSNR indicates better controllability, while lower leakage indicates that inferred actions produce consistent effects across different contexts. Bold marks the best Δ_tPSNR; leakage is not bolded, as it is interpretable only jointly with Δ_tPSNR. *Future-copy* and *Oracle* are anchors whose Δ_tPSNR diverges.

Environment	Metric	LAPO	AdaWorld	CoMo	CLAW w/o Adv	CLAW	Future-copy	Oracle
Crafter	Δ_tPSNR↑	**9.14**	0.15	7.43	2.47	8.31	∞	∞
	Leakage↓	0.42 ± 0.15	0.01 ± 0.02	0.44 ± 0.14	0.05 ± 0.06	0.28 ± 0.12	1.00 ± 0.00	0.00 ± 0.00
Climber	Δ_tPSNR↑	4.90	1.86	5.59	4.30	**5.83**	∞	∞
	Leakage↓	0.68 ± 0.09	0.17 ± 0.06	0.80 ± 0.08	0.42 ± 0.12	0.51 ± 0.11	1.00 ± 0.00	0.00 ± 0.00
Jumper	Δ_tPSNR↑	5.91	1.84	6.91	4.18	**9.07**	∞	∞
	Leakage↓	0.50 ± 0.09	0.17 ± 0.06	0.86 ± 0.05	0.31 ± 0.13	0.48 ± 0.12	1.00 ± 0.00	0.00 ± 0.00
Ninja	Δ_tPSNR↑	5.27	2.56	7.32	4.34	**9.02**	∞	∞
	Leakage↓	0.35 ± 0.10	0.23 ± 0.09	0.86 ± 0.06	0.24 ± 0.15	0.17 ± 0.11	1.00 ± 0.00	0.00 ± 0.00

inherently challenging for latent action planning because the goal region is partially occluded and the fixed camera viewpoint makes it difficult to estimate vertical end-effector motion.

Table 2 presents visual planning success rates on Crafter and three Procgen tasks using the MPC-CEM planning procedure. CLAW achieves the best performance on three of the four tasks and attains the highest average success rate overall. The consistent improvements over LAPO highlight the advantage of conditioning the world model on expressive continuous latent actions. Additional visual planning trajectories are in Appendix D.3.3.

4.4.2 Action Transfer

We qualitatively evaluate the ability of CLAW to transfer actions across videos using an action swap task, following the protocol from Gao et al. (2025). In this experiment, a sequence of latent actions is extracted from a source video and applied to an initial observation from a different target video. To assess the quality of action transfer, we compare three sequences: the original source and target videos, and the new video with the source video actions applied to the target video initial observation. This comparison provides insight into whether the transferred actions induce the intended dynamics while preserving consistency with the target scene, highlighting the generalizability and robustness of the learned latent action representation.

Results As shown in Figure 4, by $t = 4$ the transferred end-effector motion closely matches the source behavior (rows three and four). The end-effector tracks the source trajectory from a different initial frame, matching what an absolute action space would produce. The objects, by contrast, move differently, consistent with their different initialization in the new scene (rows five and six). CLAW therefore separates the action from the scene configuration it was inferred from, and learns plausible dynamics conditioned on it. Additional action transfer experiments in Procgen are in Appendix D.3.2.

4.5 Controllability and Leakage

Controllability We evaluate controllability using Δ_tPSNR (Bruce et al., 2024), which compares predictions conditioned on latents inferred from ground-truth transitions against predictions conditioned on latents drawn at random from the empirical latent distribution. Higher Δ_tPSNR indicates stronger dependence of the predicted future on the latent action.

Leakage We introduce a *pairwise leakage score* measuring whether latent actions encode context-specific information beyond the underlying semantic action. Within groups of transitions sharing the same ground-truth action, we transfer each latent to the other contexts and compare its cross-context prediction error against its matched-context error. The score lies in $[-1, 1]$: values near zero indicate cross-context consistency, positive values indicate source-context dependence. Table 5 includes two reference setups. *Oracle* predicts each target context's true next frame and has zero leakage; *Future-copy* simulates a latent action that encodes only the future frame and attains the maximum leakage score of 1. Full definitions, the evaluation protocol, and spatial leakage maps are in Appendix D.3.1.

Results A latent action representation can fail in two ways. In the first, the latent retains information about the context it was inferred from, so the world model makes good predictions in that context but inferred actions transfer poorly, resulting in high Δ_tPSNR but high leakage. As seen in Table 5, CoMo shows this pattern most clearly. In the second, the world model makes little use of the latent, resulting in low Δ_tPSNR and driving leakage toward zero for reasons unrelated to representation quality. CLAW w/o Adv reports lower leakage than CLAW on three environments, but this is only because its world model predictions depend only weakly on the inferred latent actions, as indicated by its low Δ_tPSNR and consistent with its low planning success on Crafter and Robosuite Push (Tables 2, 4). CLAW avoids both failure modes, attaining the highest Δ_tPSNR on Climber, Jumper, and Ninja while keeping leakage below LAPO and CoMo on most environments.

5 Discussion

CLAW's latent space is organized by action rather than by observation. Nearby latents correspond to similar transitions across visually unrelated scenes (Sec. 4.3.3, Sec. 4.3.4), and perturbing a latent degrades downstream performance gradually rather than abruptly, suggesting action information is spread across the space rather than concentrated in isolated points (Appendix D.1). This matters because it allows a small set of annotated transitions to ground the inferred latents to executable actions, which our policy-learning and planning evaluations rely on. Schmidt & Jiang (2024) established that discrete latent actions recover this structure; our results show it also emerges in a continuous space shaped by an adversarial objective, without quantization or a codebook.

Successful planning depends on latents that transfer: the world model must predict the same behavior from a given latent action regardless of the initial observation. CLAW's low scores on our proposed leakage metric and high Δ_tPSNR indicate that it learns an action representation that consistently controls the world model across contexts. Sec. 4.4.2 shows this qualitatively, with latents extracted from one trajectory producing the same behavior in a different context. Ablations confirm that adversarial latent regularization is the cause: CLAW w/o Adv reports even lower leakage in Table 5, but only because its predictions barely depend on the latent, and its planning success falls accordingly.

6 Conclusion

We introduce an end-to-end self-supervised method for jointly learning a world model and continuous latent action representations from unlabeled video. Adversarial regularization addresses future-observation leakage without vector quantization or action supervision, and across imitation learning from observation, visual planning, and real-world policy pretraining, CLAW achieves the best overall performance among evaluated latent action methods. Its main limitation, shared with video world models generally, is that the model learns what transitions look like rather than which ones are permitted, so it can generate futures that the environment would never produce. Addressing this likely requires interaction rather than observation alone; we discuss it and other limitations in Appendix E.

AI use statement

In this work, we used generative AI tools to assist with writing, specifically for drafting and editing portions of the manuscript text and for improving clarity and concision. We also used them to assist with plotting and figure generation code; all such code was reviewed and tested by the authors, and the underlying results it visualizes were produced by our own experimental code. We did not use generative AI tools for research ideation, experimental design, algorithm development, or the analysis and interpretation of results. All experiments were designed, implemented, and run by the authors.

We have reviewed all AI-assisted work, and every technical claim, numerical result, figure, and table in this paper was produced and checked by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts.

Ethics Statement

We do not believe this work raises ethical concerns. All datasets and environments used are publicly available and used under their original licenses, and the real-robot experiments involved no human subjects.

Reproducibility statement

We have made the following efforts to support reproduction of our results. Architectures, training hyperparameters, and the diffusion configuration for both the latent action model and the world model are specified in Appendix B and Table 7. The latent action policy training pipeline, including latent extraction, policy architecture, and training objective, is described in Appendix B.4, and the MPC-CEM planning procedure with all planner settings is given in Appendix B.5 and Algorithm 1.

For each benchmark, Appendix C describes the environment configuration and the data generation procedure, including the Crafter expert agent and its reward modifications, the OGBench oracle settings, and the Procgen collection protocol; dataset sizes and evaluation protocols are summarized in Table 8. Full definitions of the controllability and leakage metrics, including the idealized reference anchors, are given in Appendix D.3.1. The real-robot setup, camera configuration, task and staged-metric definitions, and the per-method latent extraction procedures are described in Appendix D.4 and Appendix C.5

We will open-source all code, including training, evaluation, and data generation scripts, together with the pretrained model weights for every environment reported in this paper.

References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in Atari. *arXiv preprint arXiv:2405.12399*, 2024.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Christopher G. Atkeson and Juan Carlos Santamaría. A comparison of direct and model-based reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3557–3564, 1997.

Yutong Bai, Danny Tran, Amir Bar, Yann LeCun, Trevor Darrell, and Jitendra Malik. Whole-body conditioned egocentric video prediction. *arXiv preprint arXiv:2506.21552*, 2025.

Simon Baker, Daniel Scharstein, James P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22669–22679, 2023.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2025.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control: Volume I*. Athena Scientific, 2012.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Arthur Earl Bryson. *Applied optimal control: Optimization, estimation and control*. Routledge, 2018.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Andreja Bubic, D. Yves Von Cramon, and Ricarda I. Schubotz. Prediction, cognition and the brain. *Frontiers in Human Neuroscience*, 4, 2010.

Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. GameGen-X: Interactive open-world game video generation. *arXiv preprint arXiv:2411.00769*, 2024.

Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. IGOR: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.

Andy Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Roger C Conant and W Ross Ashby. Every good regulator of a system must be a model of that system. *International Journal of Systems Science*, 1(2):89–97, 1970.

Zichen Jeff Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. DynaMo: In-domain dynamics pretraining for visuo-motor control. *arXiv preprint arXiv:2409.12192*, 2024.

Pieter-Tjerk De Boer, Dirk P. Kroese, Shie Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1755–1763, 2019.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025.

Quentin Garrido, Tushar Nagarajan, Basile Terver, Nicolas Ballas, Yann LeCun, and Michael Rabbat. Learning latent action world models in the wild. *arXiv preprint arXiv:2601.05230*, 2026.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something"' video database for learning and evaluating visual common sense. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 5842–5850, 2017.

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

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2019.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2024.

Haoran He, Yang Zhang, Liang Lin, Zhongwen Xu, and Ling Pan. Pre-trained video generative models as world simulators. *arXiv preprint arXiv:2502.07825*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Huang Huang, Sriram Yenamandra, Arjun Majumdar, Elie Aljalbout, Tushar Nagarajan, Tsung-Yen Yang, Akshara Rai, Michael Rabbat, Li Fei-Fei, Jiajun Wu, Tingfan Wu, and Franziska Meier. Cross-embodiment robot foundation world models with latent actions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2021.

Harini Kannan, Danijar Hafner, Chelsea Finn, and Dumitru Erhan. RoboDesk: A multi-task reinforcement learning benchmark. https://github.com/google-research/robodesk, 2021.

Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. UniSkill: Imitating human videos via cross-embodiment skill representations. *arXiv preprint arXiv:2505.08787*, 2025a.

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

Hadas Kress-Gazit, Kunimatsu Hashimoto, Naveen Kuppuswamy, Paarth Shah, Phoebe Horgan, Gordon Richardson, Siyuan Feng, and Benjamin Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv preprint arXiv:2409.09491*, 2024.

Jung Min Lee, Taehyun Cho, Li Zhao, and Jungwoo Lee. Why latent actions fail, and how to prevent it. *arXiv preprint arXiv:2605.20223*, 2026.

Anthony Liang, Pavel Czempin, Matthew M Hong, Yutai Zhou, Jingzhen Wang, Erdem Biyik, and Stephen Tu. CLAM: Continuous latent action models for robot learning from unlabeled demonstrations. *arXiv preprint arXiv:2505.04999*, 2025.

Tomas Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1113–1132, 2020.

Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2023.

Seungyong Moon, Junyoung Yeom, Bumsoo Park, and Hyun Oh Song. Discovering hierarchical achievements in reinforcement learning via contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 63674–63686, 2023.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. *arXiv preprint arXiv:2502.00379*, 2025.

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Jacques Richalet, André Rault, Jean-Louis Testud, and Jean Papon. Model predictive heuristic control. *Automatica*, 14(5):413–428, 1978.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. *arXiv preprint arXiv:2312.10812*, 2024.

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

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. *arXiv preprint arXiv:2005.05960*, 2020.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings*, pp. 216–224. 1990.

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

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4906–4913, 2012.

Stephen Tian, Chelsea Finn, and Jiajun Wu. A control-centric benchmark for video prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546*, 2026.

Yucen Wang, Fengming Zhang, De-Chuan Zhan, Li Zhao, Kaixin Wang, and Jiang Bian. Co-evolving latent action world models. *arXiv preprint arXiv:2510.26433*, 2025.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, 2017.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive videogpts are scalable world models. *arXiv preprint arXiv:2405.15223*, 2024.

Jiange Yang, Yansong Shi, Haoyi Zhu, Mingyu Liu, Kaijing Ma, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. CoMo: Learning continuous latent motion from internet videos for scalable robot learning. *arXiv preprint arXiv:2505.17006*, 2025.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2025.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 146676–146697, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

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*, 2025.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX

A EXTENDED RELATED WORK

A.1 COMPARISON WITH CONTINUOUS LATENT ACTION METHODS

Table 6 summarizes the main supervision, pretraining, and downstream-use differences between CLAW and recent continuous latent action methods. We focus the table on high-level properties and discuss the specific learning mechanisms below.

Table 6: **Comparison of continuous latent action learning methods.** A checkmark indicates that the property holds for the method.

Property	CLAW	CLAM	UniSkill	CoMo
Latent space learned from action-free video alone	✓	✗	✓	✓
No pretrained perceptual components	✓	✓*	✗	✗
Single-stage training of representation and dynamics	✓	✗	✗	✗
Dynamics model used for planning	✓	✗	✗†	✗

*CLAM learns its latent representation without pretrained perception in its primary setting; see text. †UniSkill uses its forward skill dynamics model to generate sub-goal images for policy conditioning, but does not optimize over it as a dynamics model for planning.

CLAM (Liang et al., 2025) jointly trains a latent inverse dynamics model and a forward dynamics model through next-observation reconstruction. Its latent representation is additionally shaped by an action decoder trained on action-labeled play data, with supervised gradients propagating into the latent inverse dynamics model; this joint training is what keeps the latent space decodable. Although CLAM's inverse dynamics model is trained on unlabeled observations, its latent action space is therefore not learned from action-free video alone. After pretraining, the forward dynamics model is discarded and the latent representation is used to relabel observation-only demonstrations for a separately trained policy.

UniSkill (Kim et al., 2025a) jointly trains an inverse skill dynamics model and a diffusion-based forward skill dynamics model, with future-frame prediction supervising the learned skill representation and an explicit information bottleneck preventing trivial solutions. It requires no ground-truth action labels, but builds on pretrained components including InstructPix2Pix initialization for the forward model and a frozen DepthAnythingV2 depth estimator, whose depth maps the inverse model consumes in place of raw RGB. The skill representation is subsequently used to train a skill-conditioned policy.

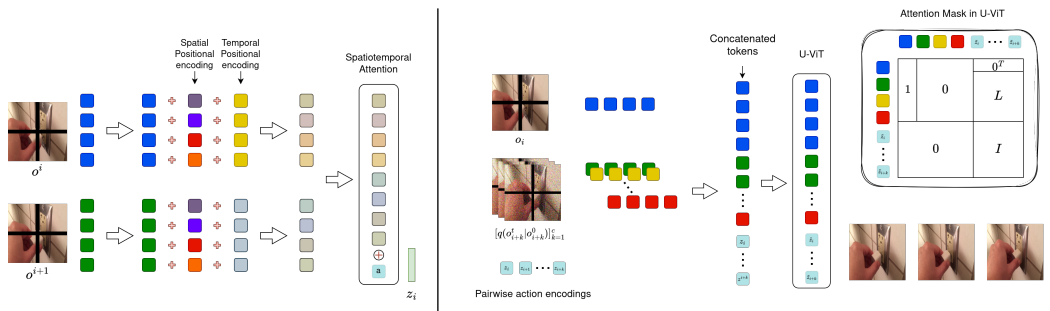
CoMo (Yang et al., 2025) learns continuous latent motion from action-free Internet videos using an MAE-pretrained ViT as its visual feature extractor. It combines three mechanisms to prevent shortcut learning: early temporal feature differencing, a temporal contrastive objective that emphasizes foreground motion and temporal direction, and a constrained latent dimensionality following the information bottleneck principle. The forward dynamics decoder is used only to supervise the inverse model during this first stage; the resulting latent motion then serves as pseudo-action supervision for a separately trained policy.

In contrast, CLAW discourages the inferred latent from encoding future visual information adversarially rather than through discrete quantization, an explicit information bottleneck, or action supervision, while retaining a continuous latent representation. The predictive model is retained after training and used directly for controllable prediction and goal-directed planning, rather than serving only as a representation-learning objective.

B CLAW IMPLEMENTATION DETAILS

B.1 LATENT ACTION MODEL

CLAW's LAM uses a ViT backbone (Dosovitskiy, 2020; He et al., 2022) as a transition model that maps consecutive observations to a continuous action code. Given a pair of frames $\{o_i, o_{i+1}\}$, the encoder produces a latent action $z_i \in \mathbb{R}^n$ as illustrated in Figure 5(a): $z_i = \text{LAM}_\theta(o_i, o_{i+1})$.



(a) Latent Action Model (b) Diffusion-based World Model

Figure 5: CLAW consists of two main components: a ViT based latent action model (LAM) and a U-ViT based diffusion world model. The LAM is designed to infer compact latent actions directly from visual state transitions, while the diffusion model predicts future observations conditioned on these latent actions. Under the causal attention mask, where L denotes a lower-triangular block and I an identity block, each noised target token attends only to the other tokens in the same frame, to the initial frame tokens, and to the action tokens corresponding to transitions from the initial frame up to the timestep of the noised target frame. This ensures that the model cannot access future information while learning to denoise based on valid action-conditioned dynamics.

These patches are projected into the model's hidden dimension and augmented with both spatial and temporal positional encodings. The spatial positional encoding is shared across corresponding spatial locations over time, enabling the model to preserve spatial correspondence between frames, while the temporal positional encoding identifies the timestep associated with each frame. Together, these encodings produce a structured spatiotemporal token representation that captures both appearance and temporal evolution.

A learnable action token is concatenated with the spatiotemporal patch tokens and processed using a spatiotemporal Vision Transformer encoder. The full sequence is processed by a ViT-style transformer encoder, which outputs the latent action that summarizes the transition between the two input observations. The final representation of this token is projected into a compact latent action vector, which is subsequently ℓ_2 normalized to produce the latent action embedding. This latent representation serves as a compressed description of the transition between observations and forms the core of the latent action model.

B.2 Diffusion-based World Model

Our world model is a diffusion-based video predictor built on a U-ViT (Bao et al., 2023) backbone. It learns to denoise target frames conditioned on latent actions extracted from observed transitions, so rollouts proceed by applying a sequence of inferred actions while iteratively refining predictions from an initial scene. Because generation is autoregressive in time, the model must respect temporal causality: predictions at step t may use the starting observation and actions up to t, but must not depend on future frames or future actions. We therefore train with a modified causal attention mask that enforces this structure in the token sequence rather than relying on the network to learn it implicitly.

The mask is designed to block information leakage at the token level. Noisy target tokens cannot attend to future action tokens, nor to visual tokens from timesteps beyond their own. As illustrated in Figure 5(b), each noised target token is allowed to attend only to (i) other tokens within the same frame, (ii) tokens from the initial conditioning frame, and (iii) action tokens associated with transitions from the initial frame through the timestep of the noised target. Intuitively, a token being denoised at time t may aggregate spatial context within its frame, retain access to the rollout's starting state, and read the action history that explains how the scene should evolve up to t, but it cannot "peek" at later actions or later observations. This restricted attention pattern keeps denoising aligned with valid action-conditioned dynamics and makes sequential transfer reliable: each generated frame

Table 7: Training and model configuration for the latent action model and diffusion world model.

Component	Configuration
Training Steps	100,000
Batch Size	512
Optimizer	AdamW
Learning Rate	3×10^{-4}
Weight Decay	0.03
Betas	$(0.99, 0.99)$
Warmup Steps	5,000
Latent Action Model (LAM)	
Spatiotemporal Encoder	ViT
Input Resolution	64×64
Patch Size	4×4
Embedding Dimension	512
Depth	8
Attention Heads	8
Latent Action Dimension	32
Latent Normalization	ℓ_2 normalization
Diffusion World Model	
Backbone	U-ViT
Embedding Dimension	512
Depth	12
Attention Heads	8
Patch Size	4×4
Diffusion Process	Variance-preserving SDE
β_0	0.1
β_1	20
Conditioning	Query frame + noised target + latent actions
Attention Mask	Structured causal masking
Sampler	Probability-flow ODE
Sampling Steps	50

is produced from causal action context, not from future information in the training or inference sequence.

We model frame generation with a variance-preserving diffusion process. Given a clean target frame o_0 (and conditioning on a query frame and latent action), the forward noising distribution is

$$q(o_t \mid o_0) = \mathcal{N}\big(\sqrt{\alpha_t}\, o_0,\ \beta_t\, I\big), \qquad x_t = \sqrt{\alpha_t}\, o_0 + \sqrt{\beta_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{3}$$

where $t \in [0, 1]$ indexes noise level, $\beta_t = 1 - \alpha_t$.

B.3 CLAW Configuration

Training and hyperparameter details for CLAW are provided in Table 7.

B.4 Latent Action Policy Training Pipeline

B.4.1 Latent Action Extraction

For each partition, latent actions are computed from consecutive pairs of visual observations using the Latent Action Model (LAM). Given two successive frames (o_t, o_{t+1}), a short temporal sequence is formed (i.e., the action chunk) and passed through LAM. The LAM produces a latent action embedding $z_t = LAM(o_t, o_{t+1})$. We use a latent vector size of 32 for all our experiments. These latent vectors represent the transformation between consecutive observations. For each video, we extract latent actions and save them together with the corresponding video frames: $\mathcal{D} = \{(o^i, z_i)\}_{i=1}^{T}$.

B.4.2 Latent Action Policy Network

The policy network is implemented as a convolutional neural network based on a ResNet-18 (He et al., 2016) backbone. The final fully connected layer is replaced with a linear projection to the latent action dimension 32 to match our latent action dimension. Given an observation o_t, the policy predicts a latent action $\hat{z}_t = \pi_\theta(o_t)$ where π_θ denotes the policy network with parameters θ. The output vector is normalized to unit norm to match the geometry of the latent action space learned by the encoder.

B.4.3 Training Objective

The policy is trained in a supervised regression setting using the extracted latent actions as targets. For a batch of observations and latent labels (o_t, z_t), the training objective minimizes the L1 distance between predicted and target latent actions:

$$\mathcal{L}(\theta) = \|\pi_\theta(o_t) - z_t\|_1.$$

The Adam optimizer is used for training with a learning rate of 10^{-4} and a batch size of 32. Training is performed for 100 epochs.

B.5 Visual Planning

We perform planning using Model Predictive Control combined with the Cross-Entropy Method (MPC-CEM) (De Boer et al., 2005). At each planning step, the planner searches over sequences of latent actions using the learned world model. CEM samples candidate latent action sequences $\tilde{z}_{1:H} = \{\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_H\}$, where $\tilde{z}_i \in \mathbb{R}^d$. Because the world model is trained on latent actions produced by the LAM, arbitrary samples from the CEM proposal distribution may lie outside the latent distribution encountered during training. We therefore project each sampled latent action onto the empirical support of the learned latent space by retrieving its nearest neighbor from a reference latent set $\mathcal{Z}_{\text{ref}}$:

$$z_i = \text{NN}\left(\tilde{z}_i, \mathcal{Z}_{\text{ref}}\right). \tag{4}$$

The resulting sequence $z_{1:H} = \{z_1, z_2, \ldots, z_H\}$ is used to condition the world model. This constrains planning queries to latent actions produced by the LAM and prevents the planner from exploiting out-of-distribution latent vectors on which the world model was not trained.

For each candidate sequence, the world model f_θ predicts a future rollout

$$\hat{o}_{t+1:t+H} = f_\theta(o_t, z_{1:H}).$$

The planner minimizes a visual goal-reaching objective defined on the terminal predicted observation,

$$J(z_{1:H}) = d(\hat{o}_{t+H}, o_g)\,,$$

where $d(\cdot,\cdot)$ is a task-dependent visual distance to the goal. For Procgen, d is an ℓ_2 distance between frames. For Crafter, d is a patch-based distance that measures local disagreement with the goal image. The same distance is used to determine task termination.

We use a CEM-style search procedure with elite retention and fixed isotropic exploration scale. At the first iteration, N candidate latent-action sequences are sampled from $\mathcal{N}(\mu, \sigma_0^2 I)$, where μ is initialized from the unused suffix of the previous MPC plan when available and otherwise to zero. After projection and evaluation, the top K projected sequences are retained and μ is set to the single lowest-cost projected sequence. In subsequent iterations, the retained elites are carried forward unchanged, while the remaining candidates are formed by adding those elites to fresh samples from $\mathcal{N}(\mu, \sigma_0^2 I)$. Thus, the exploration scale σ_0 remains fixed throughout optimization, and the proposal mean is determined by the current best projected sequence rather than by averaging the elite population.

After optimization, the first k latent actions of the best projected sequence are grounded to executable environment actions using the paired reference set $\mathcal{R} = \{(\mathbf{z}^{(m)}, \mathbf{a}^{(m)})\}_{m=1}^{M}$ and executed before replanning. The unused suffix of the selected plan is used to initialize the next planning step. Thus, nearest-neighbor retrieval serves two distinct roles: projection onto the empirical latent support before world model evaluation, and grounding of the selected latent actions to executable environment actions.

For the Procgen and Crafter planning experiments, we run five CEM updates per MPC iteration, sample $N = 64$ candidate sequences with planning horizon $H = 5$, and retain the top $K = 32$. After optimization, the first $k = 5$ actions of the best sequence are executed, so no suffix remains for warm starting. Planning continues for at most 20 MPC iterations or until $d(o_t, o_g) < \epsilon$. World-model rollouts use 50 diffusion denoising steps.

For VP^2, we use the benchmark's provided planner, which scores sampled action sequences using a cost that combines pixel-wise MSE to the goal image with a learned task-success classifier. We use the benchmark's recommended weights $\lambda_1 = 0.5$ and $\lambda_2 = 10$.

Algorithm 1 Model-Based Visual Planning with MPC-CEM

Require: World model f_θ, environment $\mathcal{E}$, reference sets $\mathcal{R}$ and $\mathcal{Z}_{\text{ref}}$
Require: Initial observation o_0, goal o_g, horizon H, execution horizon k
Require: CEM iterations I, samples N, elites $K = N/2$, noise scale σ_0, threshold ϵ
Ensure: Executed actions $\mathcal{A}_{\text{exec}}$
$o_t \leftarrow o_0$, $\mathcal{A}_{\text{exec}} \leftarrow \emptyset$, $t \leftarrow 0$
$\mathbf{z}^{\text{warm}} \leftarrow \emptyset$
while $d(o_t, o_g) \geq \epsilon$ and $t < t_{\max}$ **do**
  if $\mathbf{z}^{\text{warm}} = \emptyset$ **then**
    $\mu \leftarrow \mathbf{0}$
  else
    $\mu \leftarrow \mathbf{z}^{\text{warm}}$, zero-padded to length H
  end if
  $\mathcal{Z}_{\text{elite}} \leftarrow \emptyset$
  for $j = 1$ to I **do**
    if $j = 1$ **then**
      Sample N sequences $\tilde{\mathbf{z}}^{(i)}_{1:H} \sim \mathcal{N}(\mu, \sigma_0^2 I)$
    else
      Keep the K sequences in $\mathcal{Z}_{\text{elite}}$
      Sample $\boldsymbol{\delta}^{(\ell)} \sim \mathcal{N}(\mu, \sigma_0^2 I)$, $\ell = 1, \ldots, K$
      Form K additional candidates

$$\tilde{\mathbf{z}}^{(K+\ell)}_{1:H} \leftarrow \mathbf{z}^{\text{elite},(\ell)}_{1:H} + \boldsymbol{\delta}^{(\ell)}, \qquad \ell = 1, \ldots, K$$

    end if
    for $i = 1$ to N **do**
      for $h = 1$ to H **do**
        $z_h^{(i)} \leftarrow \arg\min_{z \in \mathcal{Z}_{\text{ref}}} \|\tilde{z}_h^{(i)} - z\|_2$
      end for
      $\hat{o}^{(i)}_{t+1:t+H} \leftarrow f_\theta(o_t, \mathbf{z}^{(i)}_{1:H})$
      $J_i \leftarrow d(\hat{o}^{(i)}_{t+H}, o_g)$
    end for
    $\mathcal{Z}_{\text{elite}} \leftarrow K$ lowest-cost projected sequences
    $i^\star \leftarrow \arg\min_i J_i$
    $\mathbf{z}^*_{1:H} \leftarrow \mathbf{z}^{(i^\star)}_{1:H}$
    $\mu \leftarrow \mathbf{z}^*_{1:H}$
  end for
  for $h = 1$ to k **do**
    $(\cdot, a^*_h) \leftarrow \text{NN}_{\mathcal{R}}(z^*_h)$
  end for
  $o_{t+k} \leftarrow \mathcal{E}(o_t, \mathbf{a}^*_{1:k})$
  $\mathcal{A}_{\text{exec}} \leftarrow \mathcal{A}_{\text{exec}} \cup \mathbf{a}^*_{1:k}$
  $\mathbf{z}^{\text{warm}} \leftarrow \mathbf{z}^*_{k+1:H}$
  $o_t \leftarrow o_{t+k}$, $t \leftarrow t + k$
end while
return $\mathcal{A}_{\text{exec}}$

C Benchmark and Dataset

In this Appendix section, we give further details about each benchmark experiment and the corresponding tasks. See Table 8 for an overview and Figure 6 for visuals.



Figure 6: Benchmarks and environments that we evaluate CLAW on. Top row (left-to-right): VP2 Robosuite, VP2 RoboDesk, OGBench Scene, Real-world human, Real-world robot. Bottom row (left-to-right): Procgen Climber, Procgen Ninja, Procgen Jumper, Crafter, Something-Something v2.

Table 8: Breakdown of CLAW benchmarks: environments, latent action world model (LAWM) training details, task adaptation details, and evaluation details. For evaluation methods, we abbreviate Closest Action Retrieval (AR), Action Transfer (AT), Visual Planning (VP), Latent Action Policy Learning (PL), Latent Action Pretraining (LAP), and Controllability (C).

Env	**LAWM Training**		**Task Adaptation**		**Evaluation**	
	Dataset	**Training Data**	**Method**	**# Trajs**	**Methods**	**# Tasks**
VP^2	Robosuite	175k frames	Retrieval	40	AR, AT, VP	1
	RoboDesk	13.9k videos	Retrieval	40/task	AR, AT, VP	5
OGBench	Oracle	826k frames	Retrieval	25/task	AR, AT, VP, PL	3
Crafter	Human Expert	62k frames	Retrieval	20	AR, AT, VP, PL, C	2
Procgen	Climber	70k frames	Retrieval	40	AR, AT, VP	1
	Jumper	70k frames	Retrieval	40	AR, AT, VP	1
	Ninja	70k frames	Retrieval	40	AR, AT, VP	1
Real-World	Something-Something V2	168k videos	Fine-tuning	50/task	LAP	2

C.1 VP2

For the VP^2 (Tian et al., 2023) benchmark, we train CLAW and the baselines on the benchmark datasets and evaluate their performance on the provided planning tasks without additional task-specific supervision. For Robosuite, we use the 5k video dataset and evaluate on 100 visual planning problems for the Push task derived from Zhu et al. (2020). For RoboDesk (Kannan et al., 2021), we use 13.9k training videos and evaluate on five tabletop tasks.

C.2 OGBench

For the OGBench (Park et al., 2025) benchmark, we collected 750 videos using the environment's built-in oracles to train our world models, then evaluated latent action policy learning on three downstream tasks: Open Drawer, Close Drawer, and Push Button.

C.2.1 OGBench Data Generation

We generated data within OGBench's (Park et al., 2025) `Scene` environment, a manipulation setup with a single robot arm, sliding handle drawer, sliding window, a cube, and two buttons. We employed OGBench's default `PlanOracle`, a scripted, non-Markovian expert policy, to collect 750 total trajectories with `noise` set to zero, filtering the tasks to just 'drawer' and 'button'. For each trajectory, we set a maximum episode length of 1001 (`max_episode_length`=1001), which terminates early upon task completion. For the pixel-based observations, we use the untransparent robotic arm setting (`pixel_transparent_arm`=False).

C.3 CRAFTER

In the Crafter (Hafner, 2021) environment, we evaluate the latent action models on two tasks. The `Greedy Wood Collection` task performance is measured by two distinct metrics, the first being success rate, the percentage of trajectories in which the agent collects at least one piece of wood. The second measures the average number of wood units collected, reflecting the agent's ability to maximize cumulative reward over a long horizon. The `Single Wood Collection` is also measured in success rate. All policy evaluations are performed by deploying the learned model in the environment and measuring its performance over 200 random seeds over the environment state.

C.3.1 CRAFTER BENCHMARK DATA GENERATION

For our imitation learning, we generated expert video samples in the Crafter game environment (Hafner, 2021) using a PPO agent based on Achievement Distillation (Moon et al., 2023). We trained separate agents on two distinct tasks, `Single Wood Collection` and `Greedy Wood Collection`.

Agent Setup and Training Details We use PPO_AD (Achievement Distillation) (Moon et al., 2023) implemented from the open-sourced codebase, using the default `PPOADModel` / `PPOADAlgorithm` architecture and losses as defined in the original paper and repository. Unless otherwise noted, we follow the repository defaults, with the following key settings: 8 parallel workers (`nproc`=8), 512-step rollouts (`nstep`=512), discount $\gamma = 0.95$, and GAE $\lambda = 0.65$. PPO optimization uses 3 epochs per update with 8 minibatches (`nepoch`=3, `nbatch`=8), clipping $\epsilon = 0.2$, value loss coefficient 0.5, entropy coefficient 0.01, learning rate 3×10^{-4}, and gradient clipping 0.5. We terminated training after 80 epochs.

Crafter Environment and Reward We train in Crafter, which provides an environment reward at each timestep, as the sum of two components. First, an *achievement* reward gives $+1$ only when an achievement is unlocked for the first time within the current episode (and 0 otherwise). The only achievement relevant to our benchmarks is `collect_wood`, which is given upon the first wood collected by the agent. Second, a *health-shaping* reward provides a small dense signal equal to -0.1 for each health point lost and $+0.1$ for each health point regenerated. Beyond these standard rewards, we make the following edits for the two specific tasks:

`Single Wood Collection`: The only achievement that yields a $+1$ achievement reward is `collect_wood`, with all other achievement rewards removed.

`Greedy Wood Collection`: In addition to removing all non wood-collecting achievement rewards, we add a dense reward of $+1$ for each additional wood collected beyond the first.

C.4 PROCGEN BENCHMARK

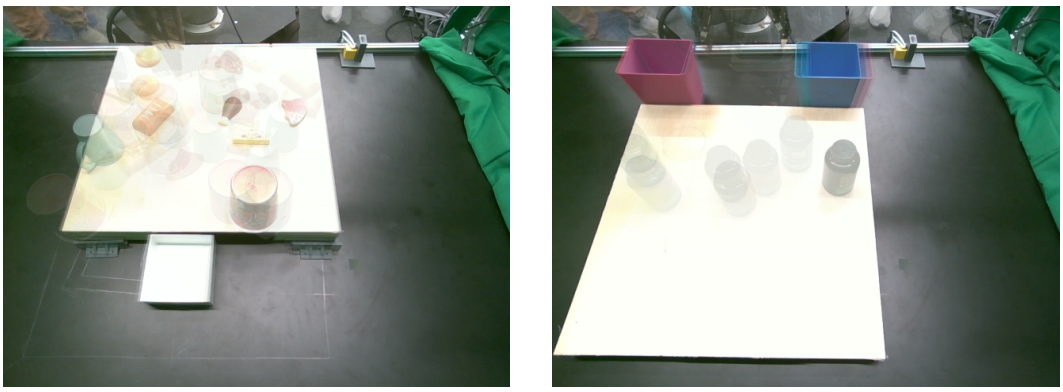
For the Procgen (Cobbe et al., 2019) benchmark, we followed the data collection procedure of AdaWorld (outlined in Appendix C.4.1). We generated 500 videos for each task (Climber, Jumper, and Ninja) as training data. We evaluate the trained model on downstream visual planning tasks. For each environment, planning problems are constructed by rolling out random action sequences of length 15 and using the initial frame as the starting observation for the world model and the frame at ($t = 15$) as the target. Planning is performed using a standard MPC-CEM procedure, as described in Appendix B.5.

C.4.1 PROCGEN BENCHMARK DATA GENERATION

We followed the same method as AdaWorld (Gao et al., 2025) to collect frame transitions from the following Procgen benchmark tasks: Climber, Jumper, and Ninja. Within each task, we collected 500 videos of an agent acting in the environment, which takes a random action at each step. Each video started from a random scene, and terminated after 1000 environment steps or when the environment returned done. Following AdaWorld, we replaced uniform action sampling with a biased schedule. For a short window we increased the probability of selecting one action, then switched to another. Cycling the favored action this way encouraged each video to cover a broader range of scenes.

C.5 Real-World Task Details

Consistent with prior work on long-horizon manipulation evaluation and recent recommendations for more nuanced robot policy evaluation beyond binary success metrics (Kress-Gazit et al., 2024), we measure performance using a staged progress-based completion metric rather than binary success. A policy receives credit for each sequential stage of a task that it completes, and we report the average progress score across trials.



(a) Push Ramen (b) Recycle Bottle

Figure 7: Initial states of the real-robot training demonstrations, viewed from `cam5`. Each panel overlays the first frame of eight randomly sampled successful trajectories, so variation in object and end-effector placement is visible in a single image.

Tasks `Recycle Bottle` is decomposed into two sequential stages: (1) reaching and grasping the bottle, and (2) lifting it and releasing it into the recycling bin. `Push Ramen` is likewise decomposed into (1) reaching and making contact with the ramen package, and (2) pushing it to the target box. Scores therefore range from 0 to 2 and are normalized to $[0, 1]$ for reporting, so a policy that consistently reaches the object but fails to complete the task scores 0.5. Each task is evaluated over 20 independent trials. Object positions are randomized across trials as described in Sec. 4.2. Figure 7 shows the spread of initial object and end-effector configurations across training demonstrations for both tasks.

D Additional Experiments and Results

D.1 Analysis of Learned Latent Action

D.1.1 Clustering

To further examine the organization of CLAW's continuous latent action space beyond the comparative UMAP analysis in Sec. 4.3.3, we visualize its structure using an alternative nonlinear projection. We encode consecutive Crafter observations as $z_i = \mathrm{LAM}_\theta(o_i, o_{i+1})$ and project the resulting latent actions into two dimensions using t-SNE. Ground-truth Crafter action labels are used only to color the visualization and are never provided to the latent action model during training.

Figure 8 provides a more detailed view of the structure that emerges within CLAW's latent space. Actions corresponding to related behaviors occupy coherent regions despite the absence of action supervision or a discrete codebook. In particular, locomotion actions such as Move Left, Move Right, Move Up, and Move Down form distinguishable regions, while several object interaction and placement actions also exhibit coherent organization. The emergence of action-aligned neighborhoods under a second visualization method is consistent with the comparative results in Sec. 4.3.3 and further suggests that the learned representation captures behaviorally relevant transition structure.

Legend for the Main-Paper UMAP Figure. The shared legend in Figure 2 of the main paper applies to all UMAP panels in that figure, with each colored square indicating the ground-truth Crafter action associated with points of the same color. Labels follow the Crafter GUI key bindings, ordered

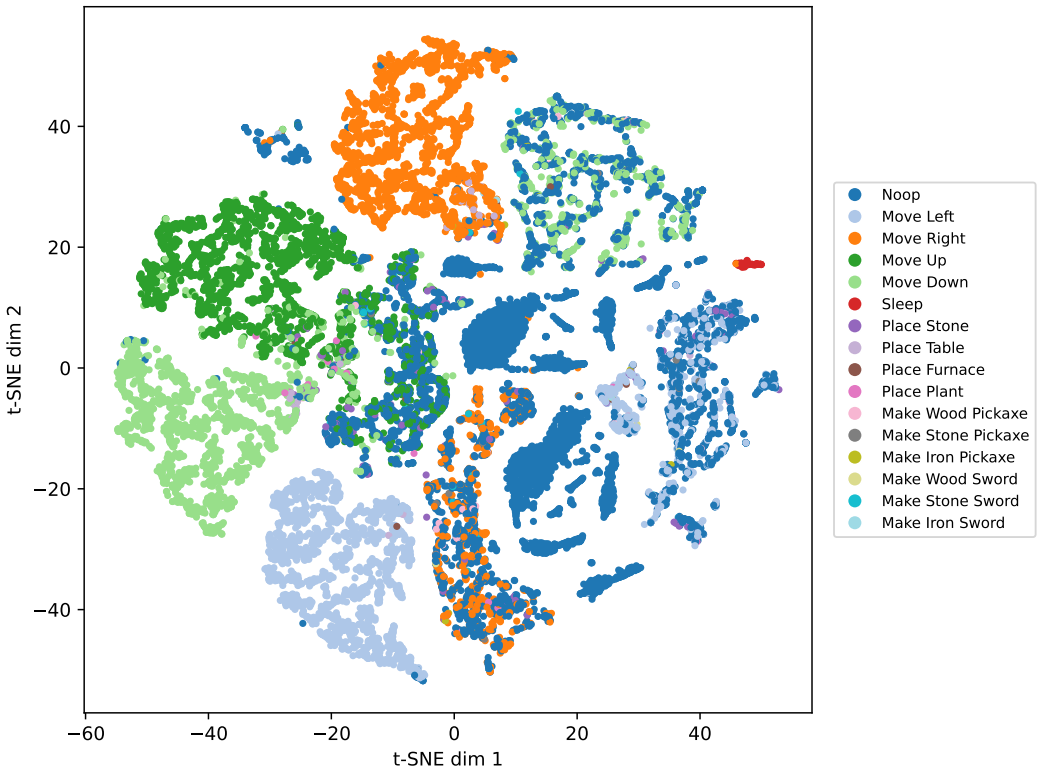


Figure 8: **Detailed structure of CLAW's continuous latent action space.** Latent actions extracted from Crafter transitions are projected into two dimensions using t-SNE. Each point represents an observation transition and is colored by its ground-truth action label for visualization only. Although CLAW is trained without action supervision or a discrete codebook, related behaviors form coherent regions in the learned continuous latent space.

from left to right: `0` denotes no-op (no dedicated key); ←/→/↑/↓ denote movement left/right/up/down (WASD in the GUI); `Tab` denotes sleep; `R`/`T`/`F`/`P` denote placing stone, table, furnace, and plant; and `1`–`6` denote crafting wood, stone, and iron pickaxes followed by wood, stone, and iron swords. The `Do` action is omitted from this visualization. Ground-truth action labels are used only for coloring the UMAP embeddings and are never provided to the models during latent-action learning.

D.1.2 Pseudo-Noop Transitions and Off-Centroid Ground-Truth Noops

The action-colored latent-space visualizations in Sec. 4.3.3 and Sec. D.1.1 use ground-truth Crafter actions as an external reference for interpreting the learned representation. However, these labels record the command issued by the policy and need not correspond one-to-one with observable transitions. We therefore examine two complementary sources of action–observation ambiguity within the dominant confusion class: non-noop commands that produce negligible visual change, and ground-truth no-op commands whose resulting transitions exhibit substantial visual change.

Notation. Let a trajectory contain observations $\{x_t\}_{t=0}^{T-1}$ and discrete actions $\{a_t\}_{t=0}^{T-2}$, where a_t is applied between x_t and x_{t+1}. We denote the environment's dedicated no-op action by $a^\star$. For the Crafter analyses below, we use the same action vocabulary as the clustering visualizations.

CLAW encodes each observation transition as

$$z_t = f_\theta(x_t, x_{t+1}) \in \mathbb{S}^{d-1},$$

where $d = 32$ and the latent representation is normalized to the unit sphere.

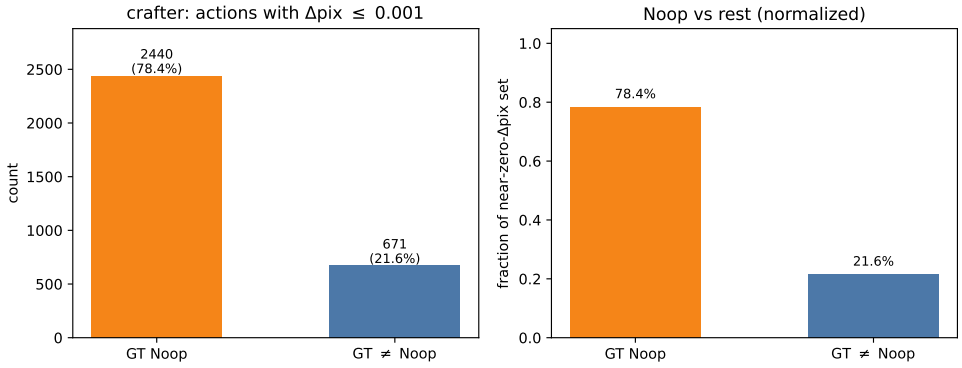


Figure 9: **True no-op and pseudo-noop transitions in Crafter.** Distribution of ground-truth `Noop` and non-noop actions among transitions with near-zero visual change ($\Delta_{\text{pix}} \leq 10^{-3}$). The analysis uses the same action vocabulary as the latent-space clustering visualizations, with `Do` excluded.

Pseudo-noop transitions. We call a transition (x_t, x_{t+1}, a_t) a *pseudo-noop* when the ground-truth action is non-noop but the resulting visual change is negligible:

$$a_t \neq a^\star \qquad \text{and} \qquad \Delta_{\text{pix}}(x_t, x_{t+1}) \leq \varepsilon, \tag{5}$$

where

$$\Delta_{\text{pix}}(x_t, x_{t+1}) = \frac{1}{CHW} \|x_{t+1} - x_t\|_1. \tag{6}$$

We normalize frames to $[0, 1]$ and use $\varepsilon = 10^{-3}$.

Pseudo-noops are not necessarily annotation errors. The recorded action correctly specifies the command issued by the human demonstrator, while the resulting observation transition may nevertheless be effectively idle. In Crafter, these cases most commonly arise from blocked locomotion, with unsuccessful place and craft commands occurring less frequently.

Prevalence in Crafter. We analyze approximately 34k transitions from a Crafter training subset using the same action vocabulary as the clustering analysis. Among transitions satisfying $\Delta_{\text{pix}} \leq 10^{-3}$, approximately 78% are ground-truth `Noop`, while approximately 22% carry a non-noop command and therefore constitute pseudo-noops. Thus, although visually static transitions are strongly enriched for true no-ops, a non-negligible fraction correspond to other issued actions.

Figure 9 summarizes this distribution.

Off-centroid ground-truth noops. The complementary case also occurs: a transition can carry a ground-truth `Noop` label while exhibiting non-negligible visual change. To examine the structure of these transitions in CLAW's latent space, we define an empirical no-op centroid. Let $\mathcal{D}^\star$ denote the set of CLAW latent representations associated with ground-truth `Noop` transitions. We compute

$$\bar{z}^\star = \frac{1}{|\mathcal{D}^\star|} \sum_{z \in \mathcal{D}^\star} z, \qquad c^\star = \frac{\bar{z}^\star}{\|\bar{z}^\star\|_2}, \tag{7}$$

and measure the distance of a latent transition to this reference using

$$d(z, c^\star) = \|z - c^\star\|_2. \tag{8}$$

We refer to ground-truth `Noop` transitions with large distance from $c^\star$ as *off-centroid ground-truth noops*. These examples probe heterogeneity within the labeled idle class rather than disagreement with the ground-truth action label itself. The no-op centroid should therefore be interpreted as a prototype of common no-op transitions, rather than as a representation of every transition carrying the discrete no-op command.

Qualitatively, ground-truth `Noop` transitions far from the centroid often exhibit appreciable visual changes, including changes in the on-screen status display or other scene dynamics despite the idle command. Conversely, pseudo-noop transitions can exhibit almost no visible change even though the issued command is non-noop. Figure 10 illustrates these complementary cases.

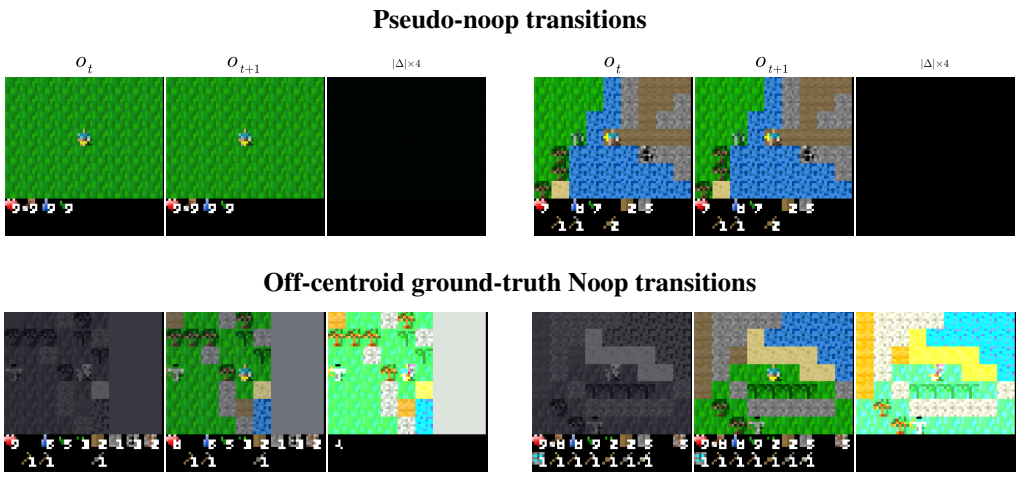


Figure 10: **Two forms of action–observation mismatch in Crafter.** Top: pseudo-noop examples corresponding to `Move Down` and `Move Left`, where non-noop movement commands produce little or no observable change. Bottom: two ground-truth `Noop` transitions whose CLAW representations lie far from the empirical no-op centroid and exhibit appreciable visual change despite the idle command. Each example shows consecutive observations together with their visual difference.

Implications for latent-action evaluation. These observations highlight an important limitation of interpreting observation-based latent actions solely through discrete ground-truth action labels. Pseudo-noops show that a non-noop command can induce an observation transition that is effectively indistinguishable from idleness, while off-centroid ground-truth noops show that an idle command can coexist with substantial visual change. Thus, the discrete action label and the observed transition effect do not define identical partitions of the data.

This distinction is particularly relevant when interpreting action-colored clustering visualizations. Ground-truth actions remain a useful external reference for assessing whether action-related structure emerges, but they should not be treated as a perfect semantic partition of visual transitions. Some apparent mixing can therefore reflect genuine ambiguity in the action–observation relationship rather than representational failure.

We accordingly interpret the clustering analyses together with nearest-neighbor retrieval, action transfer, and latent perturbation experiments. These complementary evaluations probe whether the learned representations capture consistent behavioral effects across visual contexts and whether their local geometry remains useful for downstream policy learning and planning.

D.1.3 Latent Action Policy Robustness to Latent Perturbations

A central premise of our method is that its learned latent action space captures meaningful and controllable environment dynamics. To evaluate whether downstream policies rely on this learned structure, we study the effect of perturbing latent actions during execution. Specifically, we consider the `Greedy Wood Collection` policy trained using latent action supervision and progressively corrupt its predicted latent actions with Gaussian noise.

At each environment step, the policy network maps the current observation to a latent action embedding $\mathbf{z}_\pi \in \mathbb{R}^{32}$. We generate a perturbed latent action by interpolating between the policy prediction and a randomly sampled latent vector:

$$\mathbf{z} = (1-k)\mathbf{z}_\pi + k\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{9}$$

where $k \in [0, 1]$ controls the perturbation magnitude. When $k = 0$, execution uses the policy's predicted latent action directly, while $k = 1$ corresponds to entirely random latent actions.

Following the policy evaluation protocol described in Section 4.3.1, perturbed latent actions are converted to executable Crafter actions via nearest-neighbor retrieval in the latent action dataset. We evaluate policies in `CrafterReward-v1` across 200 episodes for each noise level, sweeping k

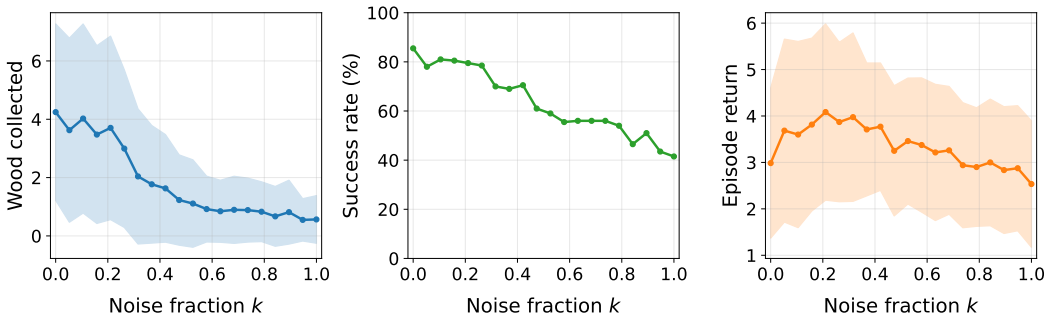


Figure 11: **Robustness of latent-action policies to latent perturbations.** Performance of the `Greedy Wood Collection` policy as Gaussian noise is interpolated into the predicted latent action (Eq. 9). Left: average wood collected. Center: success rate. Right: cumulative environment return. Shaded regions indicate one standard deviation over 200 evaluation episodes. Performance decreases smoothly as latent actions move away from the learned action, suggesting that neighboring latent actions correspond to similar behaviors and that policy performance depends on the semantic structure of the learned latent space.

uniformly between 0 and 1. We report three metrics: (i) the average amount of wood collected per episode, (ii) the success rate, defined as the fraction of episodes that collect at least one unit of wood, and (iii) the cumulative environment return. Shaded regions denote one standard deviation across evaluation episodes.

Figure 11 shows that performance degrades smoothly as latent actions are increasingly corrupted. The policy maintains strong performance under small perturbations, but progressively loses effectiveness as latent actions move away from the policy predictions. At the extreme case of pure noise ($k = 1$), both task success and reward approach random-behavior levels. These results suggest that successful control depends on the semantic structure encoded within CLAW's latent action space rather than on the retrieval mechanism alone. The smooth degradation in performance suggests that useful task information is distributed continuously throughout the latent space rather than being concentrated in a small set of isolated latent codes.

D.1.4 Structure of the Learned Latent Actions

The clustering analyses in Sec. 4.3.3 and Appendix D.1 use ground-truth actions as an external reference, but that reference is imperfect. In Crafter the recorded action is the command the demonstrator issued, not the transition it produced. Different commands can induce nearly indistinguishable transitions, and `Noop` transitions can themselves contain appreciable visual change. Appendix D.1.2 quantifies both cases. Action labels are therefore a useful reference for latent structure without defining a perfect partition of observable transition effects, and some overlap between action categories reflects ambiguity in the action–observation relationship rather than representational failure.

The remaining analyses avoid this dependence by probing behavioral consistency directly. Nearest-neighbor retrieval (Sec. 4.3.4, with further examples in Appendix D.2) shows that latent proximity tracks similar transitions across visually unrelated scenes. Perturbation (Appendix D.1.3) shows performance degrading smoothly as the latent is corrupted, so task-relevant information is distributed across the space rather than concentrated in isolated codes. Action transfer (Sec. 4.4.2, Appendix D.3.2) shows latents extracted from one trajectory inducing corresponding behavior in a different visual context. None of these compares against a discrete label.

D.2 Retrieval Task

As shown in Figure 12, our model retrieves a more diverse set of transitions that correspond to the correct actions, while exhibiting reduced reliance on superficial visual attributes such as color or texture. In contrast, AdaWorld's Latent Action Model frequently retrieves transitions from visually similar scenes, revealing a bias toward scene-specific appearance cues rather than action semantics.

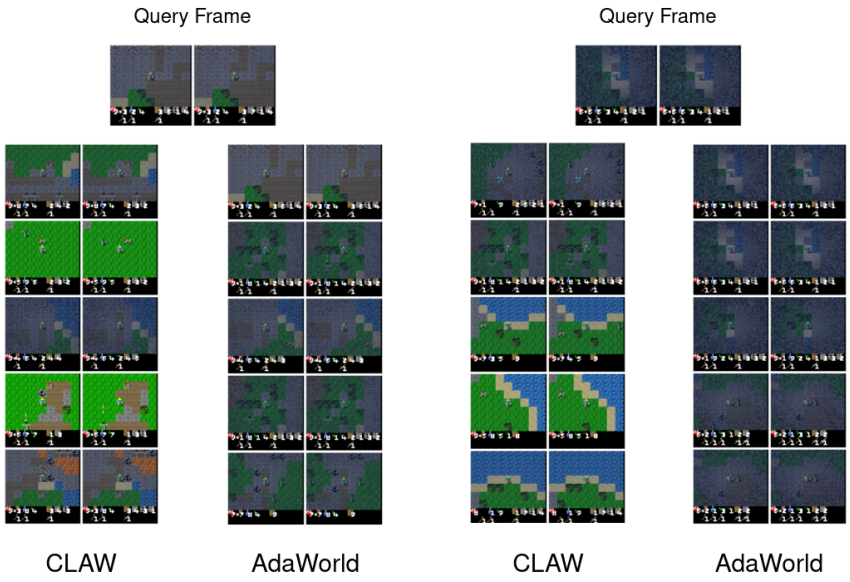


Figure 12: Nearest action retrieval comparison between CLAW and AdaWorld. For each query transition, we retrieve the closest latent actions from the latent action embeddings of the Crafter dataset and visualize their corresponding frame pairs. CLAW retrieves a more diverse set of transitions that accurately reflect the underlying action semantics, with reduced sensitivity to color and texture. In contrast, AdaWorld predominantly retrieves transitions from visually similar scenes, suggesting a stronger dependence on appearance cues rather than action-specific dynamics.

In Figure 12, the query corresponds to a `DO` action where the agent interacts with an object in the environment. CLAW retrieves transitions that capture similar object interactions, whereas AdaWorld mainly retrieves transitions from visually similar scenes with limited or no interaction. On the right side of the figure, both CLAW and AdaWorld recover the correct action category; however, CLAW shows substantially less reliance on scene color and retrieves transitions from a more diverse set of environments.

Additionally, Figure 17 presents a query transition corresponding to a `chopping tree` action. The nearest neighbors retrieved by CLAW consistently correspond to the same chopping behavior. Although AdaWorld also retrieves several chopping tree examples, it additionally includes multiple `no-op` actions. In contrast, LAPO retrieves a heterogeneous mixture of actions involving different motions and behaviors. These results indicate that CLAW learns action representations with stronger semantic clustering compared to the baselines.

Additional 10 closest action retrieval examples for different query transitions, along with a broader set of transitions, are shown in Figures 13, 15, and 16. Overall, these results demonstrate that CLAW 's latent action representations are robust and transferable, enabling more reliable planning and control in downstream visual tasks.

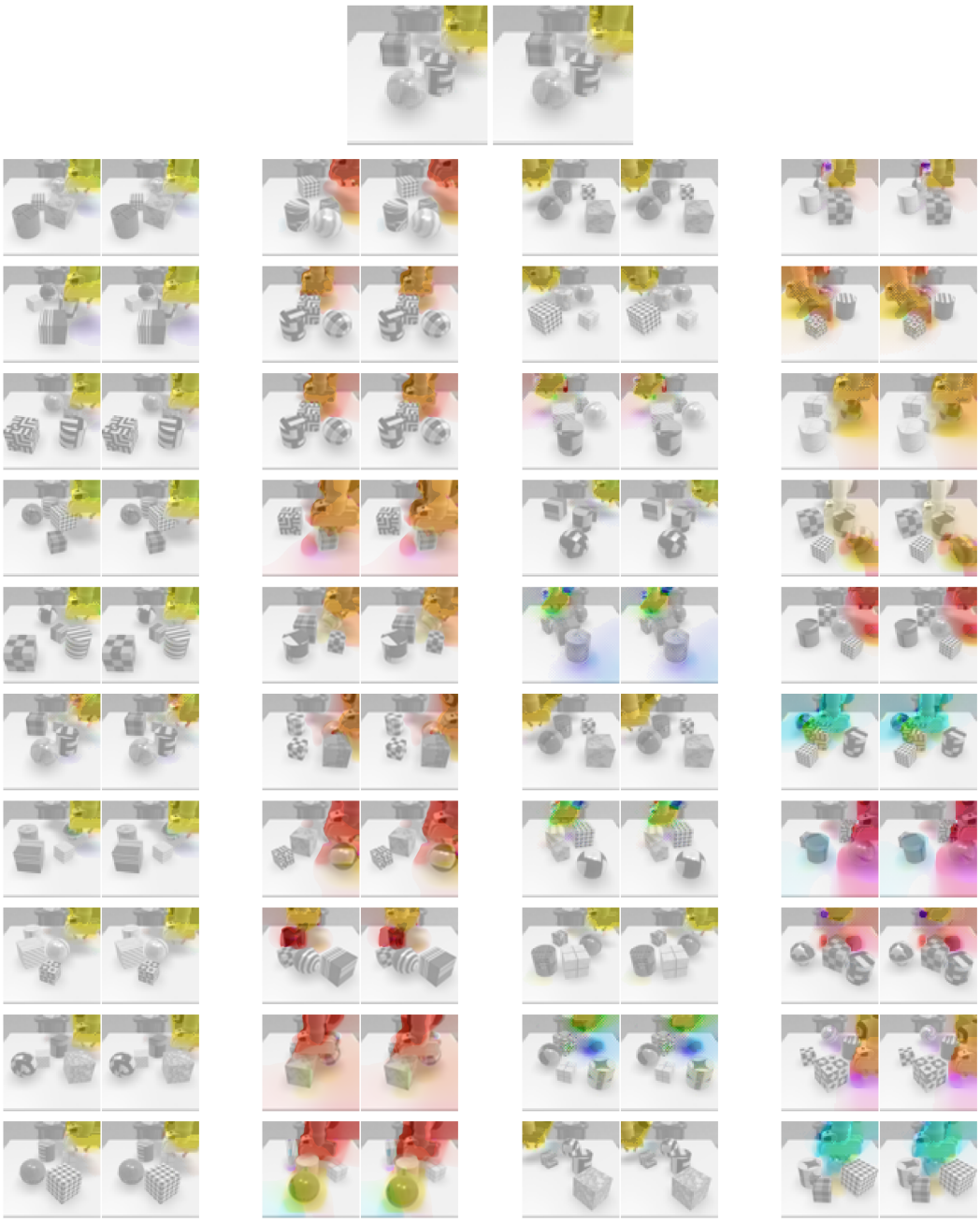
D.3 Analysis of Learned World Model

D.3.1 Bounded Latent Action Leakage Metric

We evaluate whether a learned latent action captures the underlying action independently of the visual context from which it was inferred. To control for action identity, we construct groups of transitions that share the same ground-truth semantic action:

$$\mathcal{G} = \left\{ (I_i^j, a, I_{i+1}^j) \right\}_{j=1}^{n}. \tag{10}$$

In our experiments, we randomly sample 200 same-action groups per setting, each containing $n = 8$ transitions.



(a) CLAW (b) AdaWorld (c) LAPO (d) CoMo

Figure 13: **Closest Latent Action retrieval via L_2 nearest-neighbor:** (Top) query transition. (Bottom) top-10 neighbors retrieved via L_2 search in each method's latent space (CLAW, AdaWorld, LAPO, CoMo). Retrieval is independent per encoder. All frames shown as grayscale with optical flow ($t \rightarrow t+1$) overlaid with Middlebury color wheel (Fig. 14).

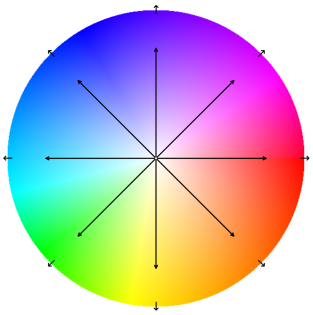


Figure 14: Middlebury Color Wheel.

For each transition j, the latent action encoder extracts

$$z^j = E(I_i^j, I_{i+1}^j). \tag{11}$$

Because all transitions in $\mathcal{G}$ correspond to the same semantic action, an ideal action representation should remain effective when transferred between contexts in the group.

We therefore apply each source latent z^j to every target context k using the learned world model:

$$\hat{I}_{i+1}^{j\to k} = W(I_i^k, z^j). \tag{12}$$

The case $j = k$ corresponds to matched-context prediction, while $j \neq k$ tests cross-context transfer.

We define the image-level prediction error

$$M_{j,k} = \text{MSE}\left(W(I_i^k, z^j), I_{i+1}^k\right). \tag{13}$$

For each cross-context transfer, we then define the bounded pairwise leakage score

$$D_{j,k} = \frac{M_{j,k} - M_{j,j}}{M_{j,k} + M_{j,j} + \epsilon}, \qquad j \neq k. \tag{14}$$

This comparison keeps the latent z^j fixed and changes only the context. Thus, $D_{j,k}$ measures whether a latent performs preferentially in the context from which it was extracted. Since the prediction errors are nonnegative, $D_{j,k} \in [-1, 1]$. A score near zero indicates that the latent transfers with similar accuracy across same-action contexts, while a positive score indicates increased error after transfer and therefore greater context dependence. Negative values indicate cases in which the transferred latent performs better than in its source context.

We aggregate across all off-diagonal transfers in a group:

$$D_{\mathcal{G}} = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k \neq j} D_{j,k}. \tag{15}$$

We report the mean and standard deviation of $D_{\mathcal{G}}$ across evaluation groups. Lower values near zero indicate less context-specific information in the latent action representation, while systematically positive values indicate stronger leakage from the source transition.

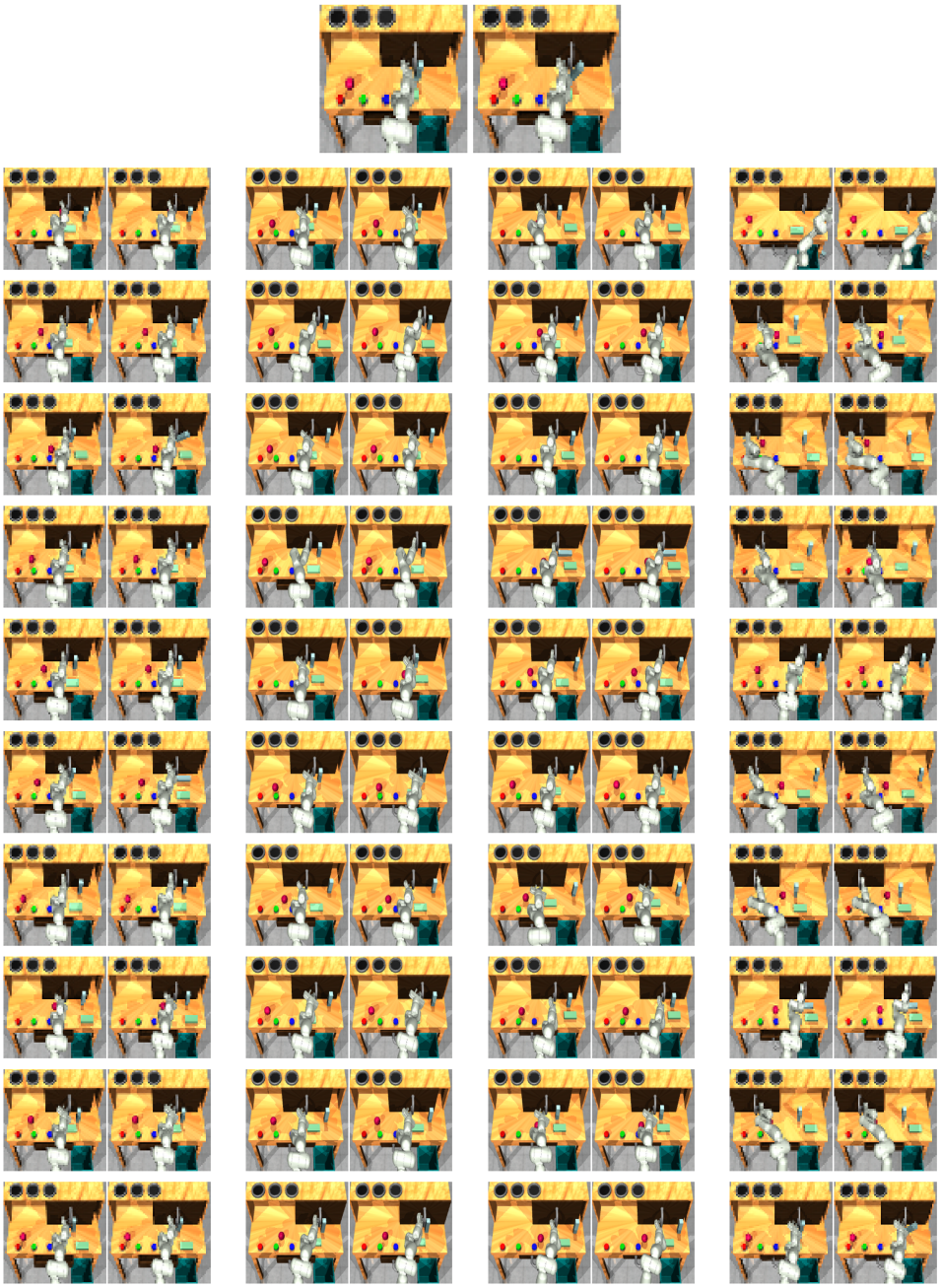
Reference anchors. To calibrate the scale we include two idealized references. *Oracle* directly predicts each target context's true next frame, so $M_{j,k} = M_{j,j}$ for all pairs and $D_{j,k} = 0$. *Future-copy* represents complete future-observation leakage: instead of encoding the underlying action, the "latent" from source transition j is taken to be the full future frame I_{t+1}^j. Copying this representation across target contexts gives near-zero error in its source context but large cross-context error, so $D_{j,k}$ approaches 1.

Why Δ_tPSNR diverges for the reference anchors. Both anchors calibrate the leakage scale but not controllability. Since $\text{PSNR} = 10 \log_{10}(\text{MAX}^2/\text{MSE})$, a prediction that matches its target exactly gives zero MSE and infinite PSNR, and both anchors are exact in the matched condition by



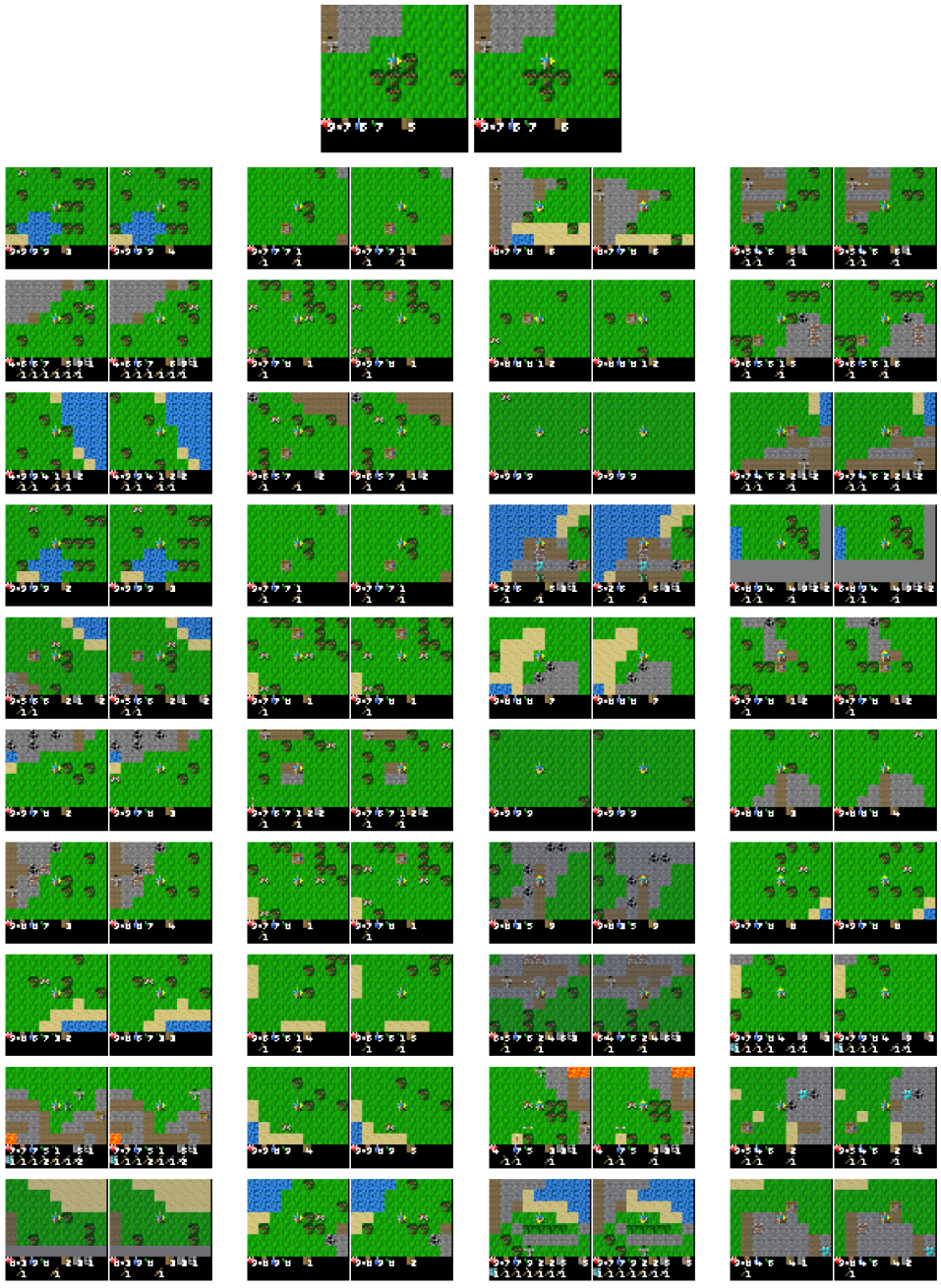
(a) CLAW (b) AdaWorld (c) LAPO (d) CoMo

Figure 15: **RGB Visualization of Closest Latent Action retrieval via L_2 nearest-neighbor:** (Top) query transition. (Bottom) top-10 neighbors retrieved via L_2 search in each method's latent space (CLAW, AdaWorld, LAPO, CoMo). Retrieval is independent per encoder. All frames shown as RGB ($t \rightarrow t{+}1$).



(a) CLAW (b) AdaWorld (c) LAPO (d) CoMo

Figure 16: **Closest Latent Action retrieval via L_2 nearest-neighbor:** (Top) query transition. (Bottom) top-10 neighbors retrieved via L_2 search in each method's latent space (CLAW, AdaWorld, LAPO, CoMo). Retrieval is independent per encoder.



(a) CLAW (b) AdaWorld (c) LAPO (d) CoMo

Figure 17: **Closest Latent Action retrieval via L_2 nearest-neighbor:** (Top) query transition. (Bottom) top-10 neighbors retrieved via L_2 search in each method's latent space (CLAW, AdaWorld, LAPO, CoMo). Retrieval is independent per encoder.

construction: *Oracle* predicts the true next frame for whichever action it is given, and *Future-copy* reproduces the latent action source future frame it was handed. Under a random latent, drawn from the empirical latent distribution, each produces a different frame and therefore finite PSNR, so the difference between the two terms diverges. What that wrong prediction is differs between them. For Oracle, a random latent is a different action, which it executes correctly in the current scene, so the prediction is a valid transition that does not match the target. For Future-copy, the latent is itself a frame, so a random latent recovers an unrelated observation and a rollout becomes a sequence of disconnected frames rather than a trajectory. Oracle's random prediction therefore remains scene-consistent while Future-copy's does not, and this is the distinction the leakage score captures. We report the anchors only for leakage in Table 5, where they bound it on $[0, 1]$.

Interpreting low leakage. A score near zero has two distinct causes, which the metric alone cannot separate. It arises when the latent genuinely transfers across contexts, but also when the world model makes little use of z: in that case $W(I_i^k, z^j)$ depends only weakly on z^j, so $M_{j,k}$ and $M_{j,j}$ are similar by construction and $D_{j,k} \to 0$ regardless of what the latent encodes. Leakage must therefore be interpreted alongside Δ_tPSNR, which measures how strongly predictions depend on the latent in the first place. Figure 18 reports both for each method.

Spatial leakage decomposition. The scalar leakage score measures the magnitude of context dependence but does not indicate where the resulting prediction degradation occurs. To visualize its spatial origin, we define the per-pixel squared prediction error

$$e_{j,k}(p) = \left\| \hat{I}_{i+1}^{j \to k}(p) - I_{i+1}^{k}(p) \right\|^2 \tag{16}$$

and construct the normalized spatial difference map

$$\mathbf{D}_{j,k}(p) = \frac{e_{j,k}(p) - e_{j,j}(p)}{M_{j,k} + M_{j,j} + \epsilon}. \tag{17}$$

Positive regions identify image locations where transferring the latent increases prediction error relative to its matched context, while negative regions indicate improved prediction after transfer. The denominator is shared across all pixels in a source-target pair, avoiding instability from pixel-specific normalization. Importantly, spatially averaging the map recovers the scalar leakage score:

$$\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbf{D}_{j,k}(p) = D_{j,k}. \tag{18}$$

Thus, $D_{j,k}$ measures how much context-specific leakage is present, while $\mathbf{D}_{j,k}(p)$ visualizes where its effect appears in the predicted frame.

Figure 18 visualizes the group-averaged leakage matrices $D_{j,k}$ on Ninja. Because each group shares a fixed ground-truth action, the off-diagonal entries measure context dependence of the latent: values near zero indicate that z^j transfers about as well as in its source context, whereas systematically positive values indicate that the world model preferentially uses the latent in the context from which it was inferred. The two outer panels bound this scale. *Oracle* has access to each target context's true next frame, so transfer costs nothing and the matrix is uniformly zero. *Future-copy* replaces the latent with the source future frame I_{t+1}^j, which is exact in its source context and uninformative everywhere else, saturating the off-diagonal entries near 1. The learned methods fall between these extremes, and their position is interpretable only alongside Δ_tPSNR: high Δ_tPSNR with high μ_D indicates a latent that is informative but context-entangled, whereas lower μ_D at comparable controllability indicates greater cross-context consistency of the action representation.

D.3.2 Action Transfer

In Figure 4 and Figure 13, CLAW's end-effector motion corresponds to rightward and slightly downward displacement much like the source (yellow in the Middlebury color wheel (Baker et al., 2011), key in Figure 14), even when object identity and scene layout differ substantially from the query. This consistency is reflected in the optical flow visualizations, where retrieved neighbors share a coherent dominant flow orientation, in contrast to retrievals from baseline methods that exhibit heterogeneous or weakly aligned flow patterns.

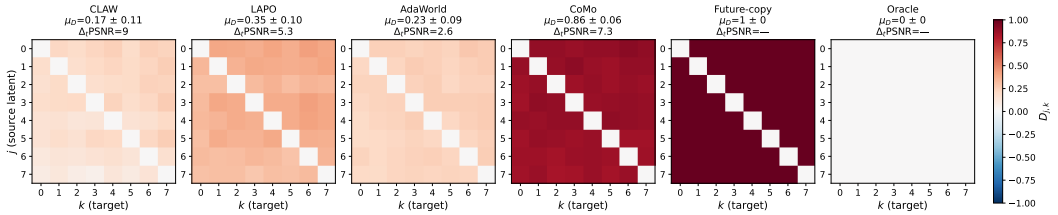


Figure 18: **Mean pairwise leakage matrices $D_{j,k}$ on a dataset taken from Procgen Ninja**, averaged over 200 same-action groups of size n=8. Rows index the source latent z^j and columns the target context k. The diagonal is zero by construction, so only the off-diagonal entries carry information. A large positive value there means the latent predicts better in the context it came from than in other contexts with the same ground-truth action, i.e. the context-specific leakage defined above. The two rightmost panels bound the scale. Future-copy uses the source future frame I^j_{t+1} in place of the latent, which is exact at home and useless anywhere else, so its off-diagonal entries approach 1. Oracle sees each target context's true next frame, so transfer costs it nothing and the matrix is zero throughout. Neither anchor conditions on a learned latent, so Δ_tPSNR does not apply. Read the two metrics together: CLAW and AdaWorld have similarly pale matrices, but AdaWorld's Δ_tPSNR of 2.6 against CLAW's 9 shows that its low leakage reflects weak latent conditioning rather than a cleaner action representation.

Additional Action Transfer experiments in the Procgen Ninja, Climber, and Jumper environments are shown in Figure 19, Figure 20, and Figure 21, respectively. We compare CLAW, AdaWorld, CoMo, and LAPO in each environment. Across all environments, as predictions progress further away from the initial observation, AdaWorld and LAPO show greater degradation than CLAW.

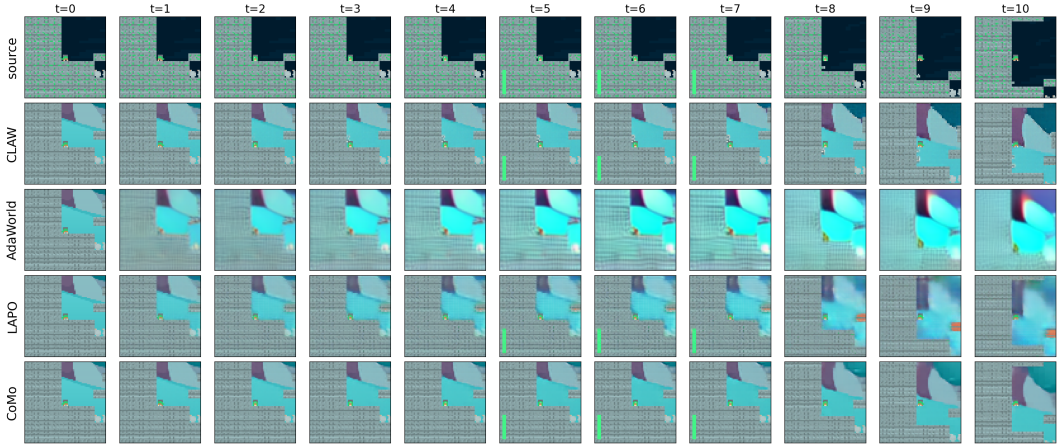


Figure 19: Action transfer on Procgen Ninja. Top: source trajectory. Below: rollouts on the target initial frame using CLAW, AdaWorld, LAPO and CoMo.

D.3.3 Visual Planning Task

We display successful and failed visual planning trajectories for the VP2 Robodesk and Robosuite benchmarks in Figure 22 and Figure 23. Visual planning trajectory successes and failures for the Procgen Ninja and Jumper environments are shown in Figure 24 and Figure 25.

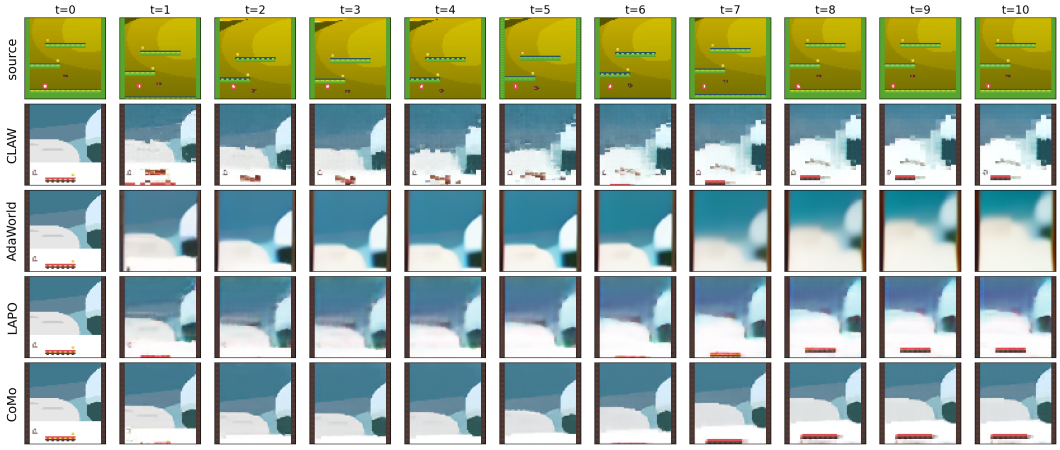


Figure 20: Action transfer on Procgen Climber. Top: source trajectory. Below: rollouts on the target initial frame using CLAW, AdaWorld, LAPO and CoMo.

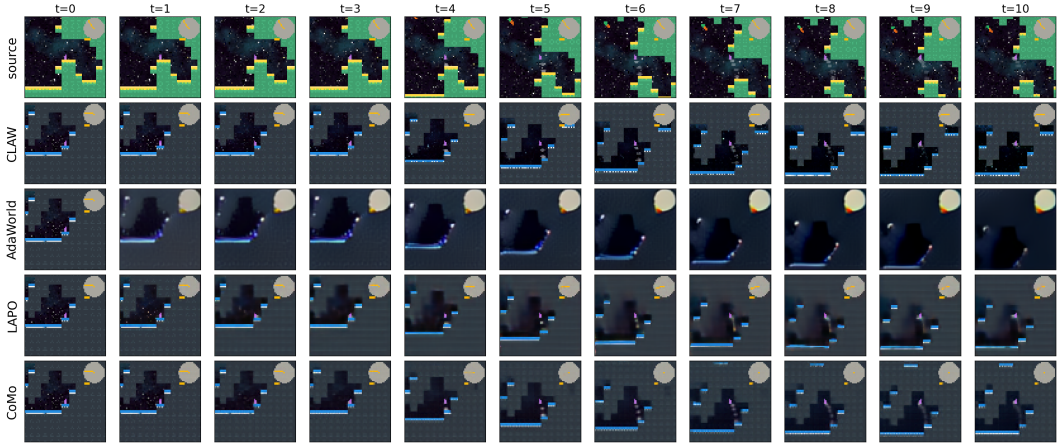


Figure 21: Action transfer on Procgen Jumper. Top: source trajectory. Below: rollouts on the target initial frame using CLAW, AdaWorld, LAPO and CoMo.



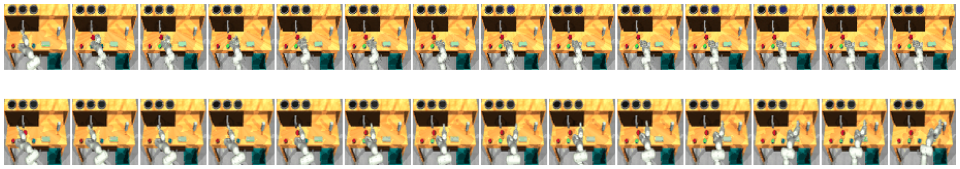
(a) Initial (b) Goal



(c) Successful trajectory for `open slide`

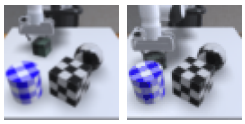


(d) Initial (e) Goal

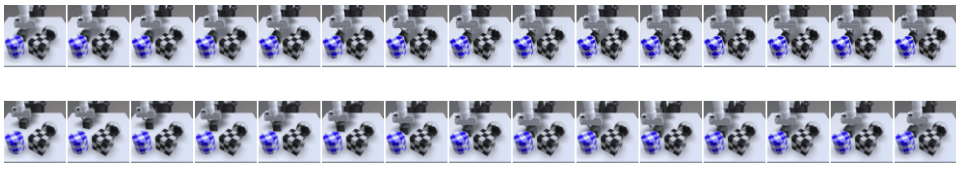


(f) Failure trajectory for `open slide`

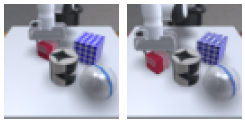
Figure 22: Visual planning results in the **VP2** benchmark using the **robodesk** environment for the task `open slide`. We show both successful and failed trajectories generated by the planner.



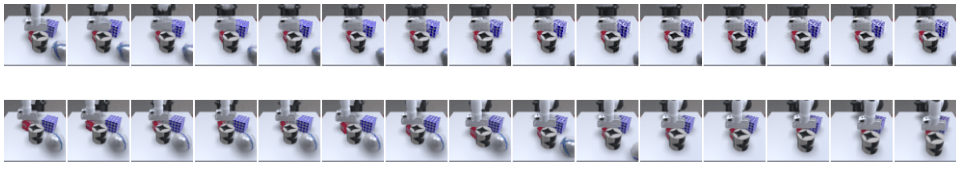
(a) Initial (b) Goal



(c) Successful trajectory for `push`

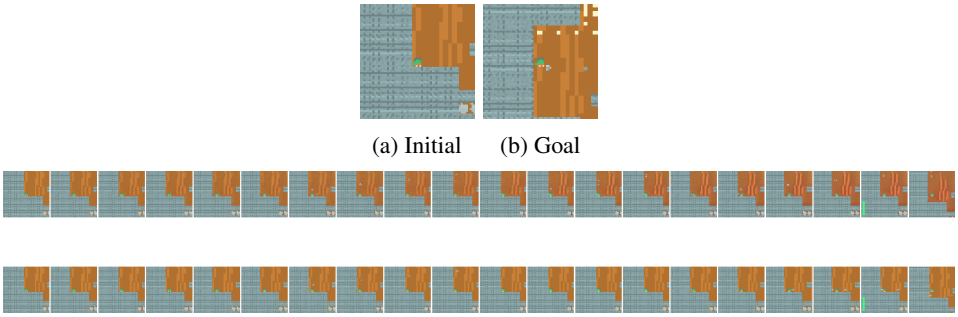


(d) Initial (e) Goal



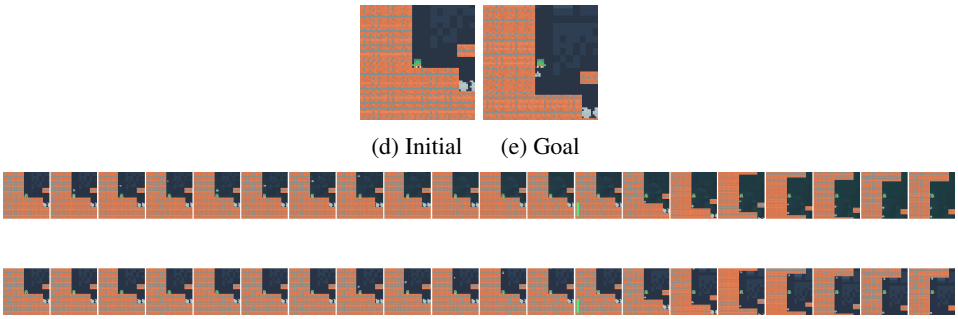
(f) Failure trajectory for `push`

Figure 23: Visual planning results in the **Robosuite** benchmark using the **robosuite** environment for the task `push`. We show both successful and failed trajectories generated by the planner.



(a) Initial (b) Goal

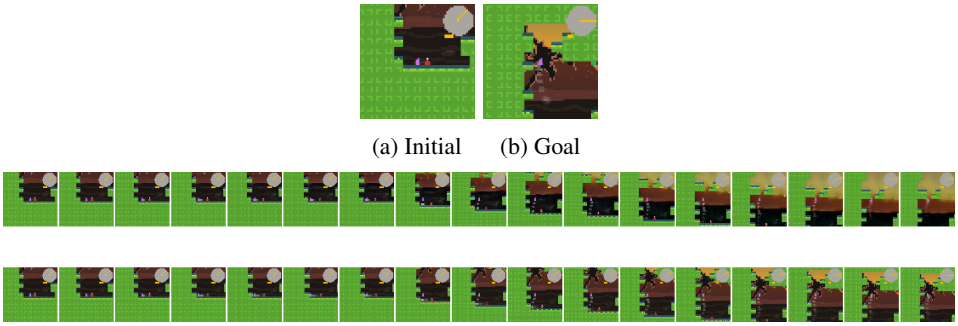
(c) Successful trajectory for `Ninja`



(d) Initial (e) Goal

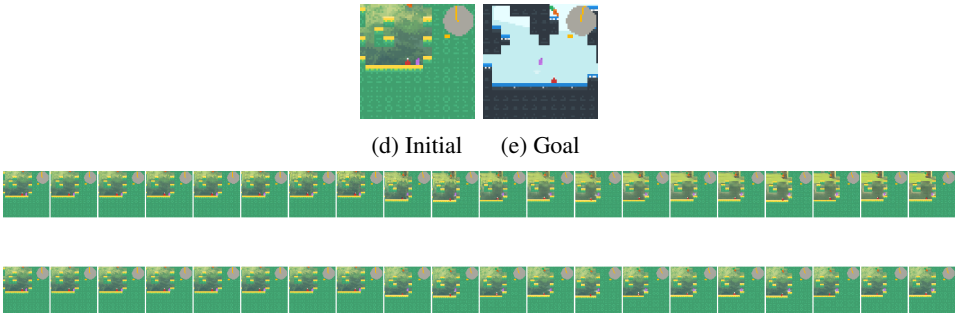
(f) Failure trajectory for `Ninja`

Figure 24: Visual planning results in the **ProcGen** benchmark using the **ninja** environment for the task `Ninja`. We show both successful and failed trajectories generated by the planner.



(a) Initial (b) Goal

(c) Successful trajectory for `Jumper`



(d) Initial (e) Goal

(f) Failure trajectory for `Jumper`

Figure 25: Visual planning results in the **ProcGen** benchmark using the **jumper** environment for the task `Jumper`. We show both successful and failed trajectories generated by the planner.

D.3.4 World Model Controllability and Transfer.

Beyond the latent action representation, our results also characterize how the learned world model uses these latents to model environment dynamics. Sec. 4.5 shows that CLAW's predictions are strongly conditioned on the latent action while remaining comparatively robust to cross-context transfer of latents representing the same semantic action. The high Δ_tPSNR indicates that different latent inputs induce meaningfully different predicted futures, while the bounded leakage analysis shows that this conditioning is not explained solely by source-context-specific information. Together, these results suggest that the world model learns dynamics that are meaningfully conditioned on the latent action, with different latent inputs inducing distinct and transferable future transitions.

This interpretation is further supported by the planning experiments in Sec. 4.4.1, where optimizing latent action sequences through the world model produces goal-directed behavior. Thus, the learned forward dynamics are not only predictive, but sufficiently action-sensitive and transferable to support downstream planning.

D.3.5 What the Adversarial Penalty Removes

The adversarial pathway penalizes prediction of the future observation from z_i alone, which raises a question about action spaces in which the action is highly predictive of the next frame. If a robot is controlled in absolute joint coordinates, the action determines the end-effector configuration outright, and a latent that faithfully encodes it will carry real information about the future. Penalizing that would remove action content rather than leakage.

The distinction is context specific. An absolute-coordinate action predicts where the end-effector will be in any scene; it does not predict which objects are present, where they sit, or what the background looks like. Because the adversarial pathway receives only z_i, its loss decreases only through information that predicts this particular future without the current observation, so the gradient pushes against scene-specific content rather than against action content that generalizes across contexts. Whether the optimization separates the two cleanly is an empirical question, and one the leakage metric (Appendix D.3.1) is designed to probe.

The action transfer results in Sec. 4.4.2 are consistent with this account. We observe that the learned latents specify end-effector motion directly: transferred to a different initial frame, they reproduce the source end-effector trajectory while the objects respond according to the new scene. This suggests the absolute component of the action survives the regularizer, though we have not isolated it directly.

The penalty is applied uniformly, with a schedule rather than a learned per-dimension weighting, so in domains where the action determines almost the entire observation the pressure may suppress action content along with context. We did not observe this in our settings, but we have not tested action spaces in which the agent controls the full scene.

D.3.6 Sampling Diversity Under a Fixed Latent Action

The Crafter view is centered on the agent, so moving up exposes a previously off-screen row whose contents are not recoverable from I_t. Figure 26 shows ten samples from each world model conditioned on the same initial frame and the same latent action. CLAW reproduces the same scrolling motion in every sample while varying the newly revealed terrain, so the samples agree on what the action does and differ only where the outcome is genuinely unknown. LAPO and CoMo decode deterministically and return an identical frame for every seed, committing to one completion of a region they cannot observe.

D.4 Further Details For the Real-World Experiments

As shown in Figure 27, a UR5 robotic arm is mounted on a fixed base in a tabletop manipulation setting. The workspace is captured from three cameras: a wrist-mounted camera (`cam2`, red) that travels with the end-effector for a close-up egocentric view, and two static third-person cameras (`cam5`, yellow, and `cam6`, purple) that provide complementary perspectives of the full workspace. Demonstrations are collected via teleoperation, with each rollout recorded as a synchronized multi-view RGB trajectory.

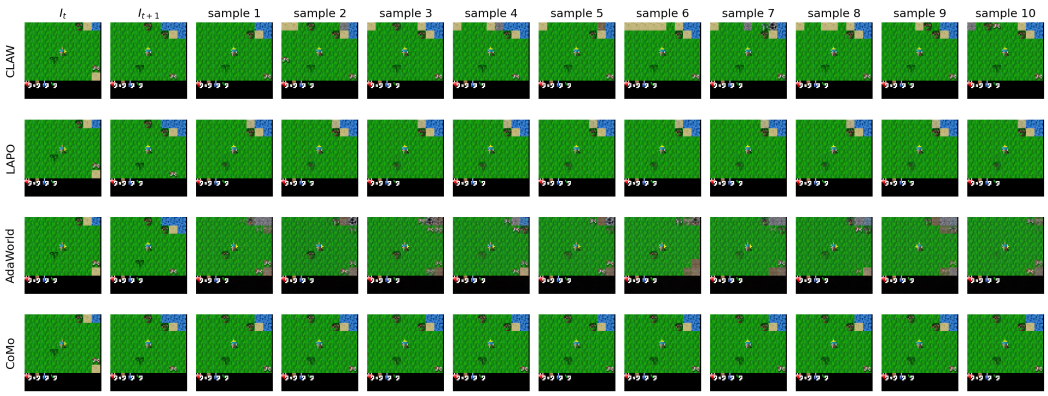


Figure 26: **Sampling diversity on a shared up transition in Crafter.** The first two columns show the frames the latent z is extracted from. The remaining columns are ten independent samples of $\hat{I} = W(I_t, z)$ under the same latent, varying only the sampling seed. The motion is the same in every sample; only the newly visible terrain along the top edge changes. LAPO and CoMo are deterministic, so each seed returns the same frame.

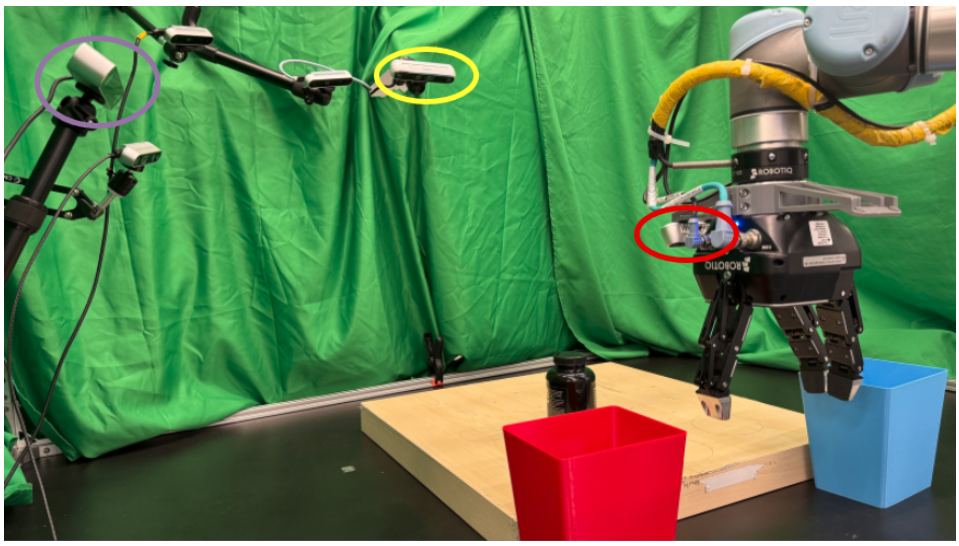


Figure 27: Workspace setup for the real robot experiments in Sec. 4.3.2. Three camera views are used for observation: a wrist-mounted camera (`cam2`, red) and two third-person cameras (`cam5`, yellow, and `cam6`, purple).

D.4.1 UR5 Latent Action Extraction.

We extract latent actions offline from frozen pretrained models on UR5 teleoperation demonstrations. Each successful rollout provides synchronized RGB observations from all three camera views (`cam2`, `cam5`, and `cam6`). Demonstrations are processed episode-by-episode, with latents indexed by the timestep of the first frame in each encoded transition (our method and AdaWorld) or by the observation timestep itself (LAPA). All models are run in evaluation mode with gradients disabled, and RGB inputs are resized to 224×224 with bilinear interpolation where required.

CLAW. We use a frozen LAM trained on Something-Something V2 Dataset. For each camera view, the encoder receives a two-frame clip $(o_j, o_{j+\Delta})$ with $\Delta = 3$, so each latent summarizes the visual change over three control steps.

AdaWorld. We apply the publicly released AdaWorld LAM with the same 32-d latent dimension. On UR5, the LAM is fed consecutive frame pairs (o_{j-1}, o_j), with latents subsampled every $\Delta{-}1 = 2$ timesteps to reduce redundancy at the demonstration control frequency. Extraction is performed separately for all three camera views.

LAPA. LAPA uses a qualitatively different interface: a frozen 7B `DeltaSampler` that predicts discrete delta tokens rather than a continuous transition embedding. For every frame, the image is paired with a fixed task-specific natural-language instruction ("Pick up the bottle and place it in the recycling bin" and "Push the ramen to the box"), and the `DeltaSampler` autoregressively generates four delta token IDs from a vocabulary of eight. These token IDs serve as the downstream latent representation and are distinct from the continuous pre-quantization vectors in the LAQ submodule.

CoMo. We use a frozen latent-motion model pretrained on Something-Something v2. As with CLAW, the encoder takes a two-frame clip $(o_j, o_{j+\Delta})$ with $\Delta = 3$, so each latent covers three control steps. A frozen ViT-MAE embeds both frames, and eight learned motion queries read out the difference between their patch tokens, each projected to 16 dimensions for a continuous 128-d code. No vector quantization is applied.

E LIMITATIONS

The LAM encodes whatever changed between two frames, and it has no way to tell an agent action from an environment-induced transition. Crafter's day-night cycle is one case: ground-truth `Noop` transitions that coincide with a lighting change sit farther from the learned no-op centroid than idle transitions do (Figure 10), because the LAM registers the change even though the agent caused none of it. Procgen Ninja shows the same effect during rollout, where lights appearing in the scene are treated as part of the action (Figure 19). This is not specific to CLAW: every method that generates visually plausible future observation reproduces the lights, since each infers its latent from the full visual transition. Factoring the latent into agent-driven and environment-driven components is a natural next step, which we leave to future work.

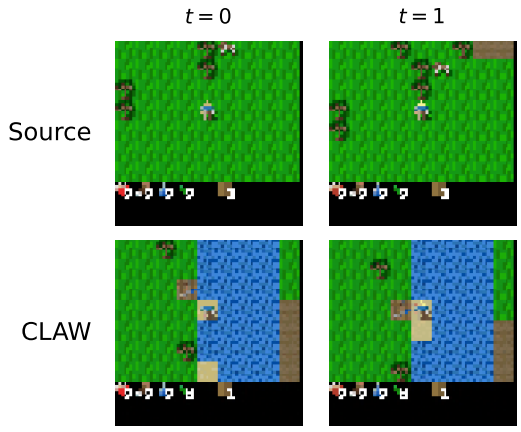


Figure 28: **Physically implausible rollout in Crafter.** Given a source transition encoding upward movement, the world model moves the agent onto a water tile by replacing it with sand. The environment would return a `no-op`.

The world model is also free to roll out transitions the environment would never permit. Because it is trained to generate plausible continuations rather than to respect the environment's rules, it can satisfy a latent action by changing the scene instead of refusing the action. Figure 28 shows this in Crafter: given a latent corresponding to upward movement while the tile above the agent is water, the environment would block the move and return a no-op transition, but the world model instead renders the agent moving up with the water tile replaced by sand. The rollout is visually coherent and locally consistent with the action, yet it does not correspond to any transition the environment can produce. This matters for planning, where a plan optimized against such rollouts may be unexecutable. Constraints of this kind are hard to recover from passive video: a blocked command produces no

visual change, so the LAM encodes it as a near-idle latent rather than as evidence that the action was refused. Learning which actions an environment rejects would likely require interaction with it rather than observation alone.

The world model also predicts from the most recent frame and latent action with no explicit memory. In partially observable settings this costs accuracy and makes long rollouts drift, since anything that left the frame is gone.

Finally, latent actions are not executable on their own. Grounding them to a particular embodiment still takes a small amount of action-labeled data, so CLAW reduces the annotation burden rather than removing it.