LeWAM: End-to-End World and Action Modeling with JEPAs

Anonymous authors
Paper under double-blind review

Website Code

Abstract

Joint-embedding predictive architectures (JEPAs) learn compact latent world models without pixel reconstruction. However, existing JEPAs typically learn representations and dynamics independently of downstream control objectives, complicating the pipeline and leaving unexplored how world- and action- learning benefit each other. We introduce **LeWAM**, an end-to-end JEPA for control with: (1) a joint training objective over next latent state prediction and action flow matching, and (2) an isotropic Gaussian constraint on the latent state, enabling a single model to support direct policy execution and efficient planning by evaluating sampled actions with its learned latent dynamics. Compared with prior JEPAs in visual planning, LeWAM improves success rate from 26.8% to 85.5% on contact-rich tasks and from 10.9% to 46.2% on long-horizon tasks, and plans 32.3× faster. On behavior cloning tasks in simulation and physical robots, LeWAM is also competitive with established methods. Interpretability analyses show that LeWAM learns control-relevant representations and action distributions effectively guiding test-time planning and control.

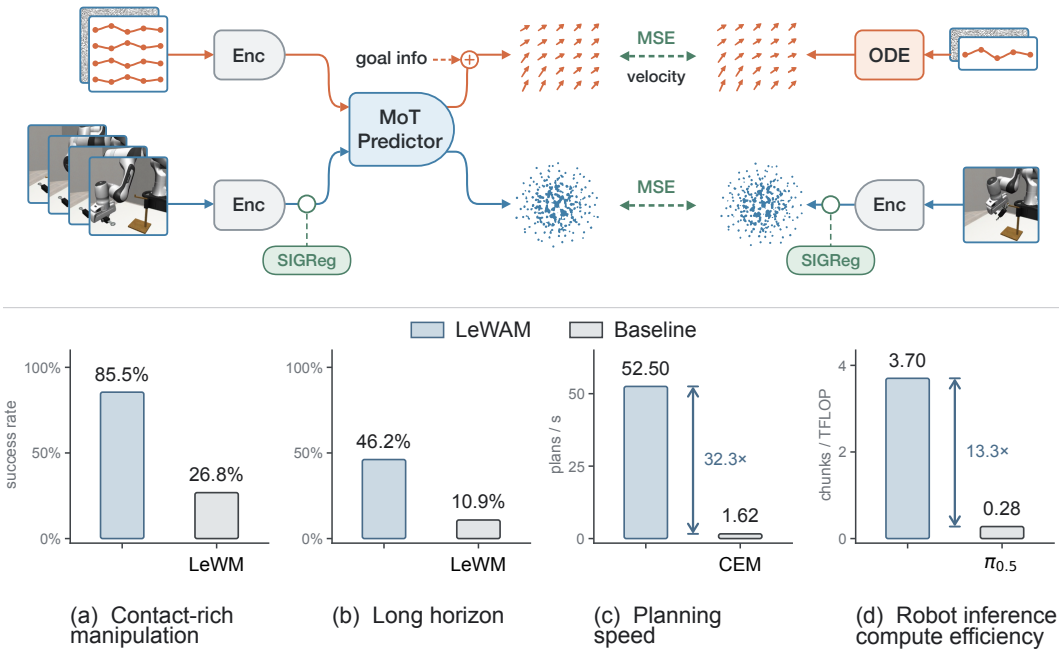


Figure 1: **LeWAM training.** A shared encoder encodes observed frames with SIGReg (Balestriero & LeCun, 2025) regularization. The Mixture-of-Transformers (Liang et al., 2025) learns latent-state prediction from clean actions and action flow matching from noisy actions with goal information. **Results overview.** The mean of (a) planning success on contact-rich tasks; (b) planning success on a 100-step goal offset; (c) plans/s versus CEM (Rubinstein, 1999); (d) action chunks per TFLOP versus $\pi_{0.5}$ (Physical Intelligence et al., 2025). Full results are in Section 4.

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

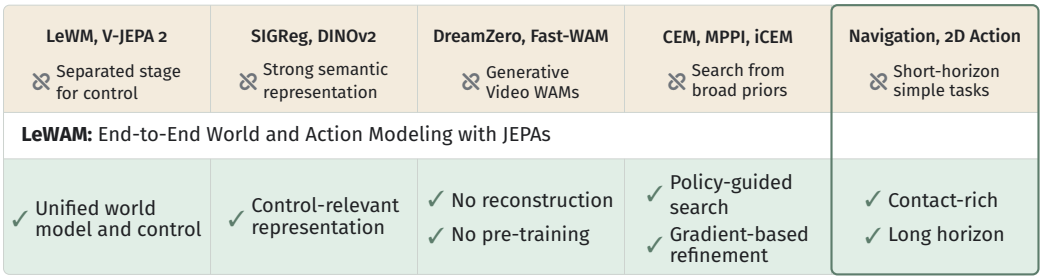


Figure 2: **Positioning of LeWAM.** LeWAM addresses key limitations in prior visual control frameworks: (1) Unlike JEPAs requiring a separate stage for control, it unifies world modeling and action policy learning end-to-end; (2) Compared to learning strong semantic representations, action prediction tailors representations to be more control-relevant; (3) In contrast to pixel-generative world action models, it operates in latent space without reconstruction; and (4) Rather than searching from broad priors, it guides planning using learned policy proposals with latent gradient refinement.

1 Introduction

Visual control requires autonomous agents to complete tasks by selecting actions based on visual observations of the world. World models address this challenge by predicting the consequences of candidate actions, allowing agents to anticipate how the world will evolve before executing them. Joint-embedding predictive architectures (JEPAs) (LeCun, 2022) offer a compact world modeling paradigm where future world states are predicted in latent space rather than by reconstructing pixels. Recent works such as DINO-WM (Zhou et al., 2024) and LeWorldModel (LeWM) (Maes et al., 2026b) demonstrate the promise of applying action-conditioned latent world models to downstream control. In world model pretraining, these pipelines use pretrained visual embeddings (Zhou et al., 2024; Assran et al., 2025) or distributional regularizers such as SIGReg (Balestriero & LeCun, 2025) to learn strong semantic representations and prevent collapse. However, these methods alone do not reliably retain control-relevant information; JEPAs can favor temporally persistent distractors (Sobal et al., 2022), while action-conditioned dynamics may retain extraneous observation information (Brandfonbrener et al., 2023).

Control with these models typically requires an online solver such as the cross-entropy method (CEM) (Rubinstein, 1999) or a policy model post-trained in a separate stage, such as GC-IDM (Nguyen et al., 2026) and PRISM (Wang et al., 2026). When initialized with a broad Gaussian prior over action sequences, CEM demands extensive rollouts to identify viable candidates, particularly as action dimensionality or planning horizon grows (Huang et al., 2021). Contact transitions also complicate local action optimization because the dynamics can switch between modes (Pang et al., 2023). Post-training a policy model over frozen JEPA embeddings leaves action supervision disconnected from visual representation learning.

We introduce **LeWAM**, an end-to-end JEPA that jointly learns representation, action-conditioned dynamics, and a stochastic policy in a single training stage (Figure 1). Training combines only two ingredients: a joint training objective over next-latent-state prediction and action flow matching, and SIGReg (Balestriero & LeCun, 2025) to regularize the latent states. A Mixture-of-Transformers (MoT) (Liang et al., 2025) predictor neatly satisfies action-conditioned dynamics modeling over clean actions and policy learning over noisy actions through flow matching, jointly shaping representations to preserve directions relevant for predicting actions and anticipating their effects. Consequently, LeWAM supports both direct policy execution and visual planning by refining action sequences sampled from its learned policy. Our contributions are threefold:

- The first JEPA that learns representation, action-conditioned latent dynamics, and stochastic policy from pixels in a single end-to-end training stage.
- A planning procedure that samples candidate action sequences from the learned action distribution and refines them with gradients through the latent state-action joint imaginations.
- Across ten heterogeneous simulated environments, LeWAM achieves a $32.3\times$ planning speedup over LeWM (CEM) (Maes et al., 2026b) alongside consistent success rate improvements, particularly on contact-rich and long-horizon tasks. On goal-agnostic task completion in simulation

and physical robot execution, LeWAM performs competitively with established behavior cloning baselines. Mechanistically, interpretability analyses show that LeWAM learns control-relevant representations and action-conditioned dynamics, while its structured action distributions yield feasible proposals that guide test-time planning.

2 Related Work

World Models. World models (Ha & Schmidhuber, 2018) learn environment dynamics to enable agents to simulate candidate action outcomes for planning and control. Early approaches like PlaNet (Hafner et al., 2019) perform planning via latent rollouts, whereas TD-MPC (Hansen et al., 2022) and TD-MPC2 (Hansen et al., 2024) ground latent dynamics in reward and value predictions. Dreamer (Hafner et al., 2020; 2023) learns generative latent dynamics with pixel reconstruction, while VideoGPT (Yan et al., 2021) and IRIS (Micheli et al., 2023) model discrete visual tokens. Within this landscape, LeWAM explicitly targets latent dynamics and policy representations directly from offline trajectories, without reward supervision or pixel reconstruction.

Joint-Embedding Predictive Architectures. JEPAs predict target states in representation space without pixel reconstruction, using variance and covariance constraints (Bardes et al., 2022), momentum targets (Assran et al., 2023; Bardes et al., 2024), or distributional regularization (Balestriero & LeCun, 2025; Wu et al., 2026) to prevent collapse. For action-conditioned world modeling, DINO-WM (Zhou et al., 2024) and V-JEPA 2-AC (Assran et al., 2025) build on pretrained representations, whereas LeWM (Maes et al., 2026b) leverages SIGReg (Balestriero & LeCun, 2025) to learn the encoder and latent dynamics from scratch. Additional action prediction objectives attempt to make representations action-aware in PLDM (Sobal et al., 2025) or as an anti-collapse signal in SMWM (Ivashkov et al., 2026). LeWAM jointly trains action-conditioned latent dynamics and policy in one predictor, allowing representations to preserve more control-relevant directions.

Control with Latent Dynamics. Conventional planning in latent world models uses population-based solvers (CEM (Rubinstein, 1999), iCEM (Pinneri et al., 2021), MPPI (Williams et al., 2017)) or gradient methods (Henaff et al., 2017; Huang et al., 2021). Population-based search can require many model rollouts as the action dimension or horizon grows (Pinneri et al., 2021; Huang et al., 2021); contact mode changes also complicate local optimization (Pang et al., 2023). GC-IDM (Nguyen et al., 2026) post-trains a goal-conditioned action predictor for direct control. PRISM (Wang et al., 2026) trains a Gaussian prior head on frozen world-model features; its predicted mean and standard deviation guide MPPI sampling. LeWAM jointly trains the visual encoder, latent dynamics, and a flow-based action generator, using its action samples for direct control or gradient-based planning (Discussion in Appendix F).

World Action Models. These models couple world prediction with action generation (Table 4, Appendix B.2). Built on pretrained video models, DreamZero (Ye et al., 2026) jointly models future video and actions, while Fast-WAM (Yuan et al., 2026) retains video co-training but skips future video generation at inference. UVA (Li et al., 2025), UWM (Zhu et al., 2025), and Cosmos 3 (NVIDIA, 2026) specifically support both forward dynamics modeling (FDM) and policy learning, but optimize only one objective per sample, rather than both jointly on the same pass. Furthermore, their results on planning remain limited. Instead of borrowing pretrained video models, LeWAM jointly trains its encoder and predictor end-to-end without pixel reconstruction. Each training sample supplies clean actions for FDM and noisy actions for stochastic policy modeling. Thus, LeWAM can learn dynamics from both demonstrations and non-expert interactions, and specifically allows action-conditioned dynamics to evaluate and refine actions at test time.

3 Methodology

Given reward-free offline trajectories of pixel observations $\boldsymbol{o}_{1:T}$ and actions $\boldsymbol{a}_{1:T}$, we jointly train a visual encoder and a predictor with action-conditioned dynamics and a flow matching policy, supporting direct control and planning with action-conditioned latent dynamics.

3.1 End-to-End Training with LeWAM

Architecture. Figure 1 summarizes the model at a high level. Let $\boldsymbol{z}_t = \mathrm{Enc}(\boldsymbol{o}_t)$ denote the latent representation of observations $\boldsymbol{o}_t$. All time indices refer to raw environment steps, such that action $\boldsymbol{a}_t$ transitions $\boldsymbol{o}_t$ to $\boldsymbol{o}_{t+1}$. We use a frameskip of K environment steps, so future observations used as dynamics targets are spaced at K-step intervals. In contrast, the policy models a contiguous action chunk $\boldsymbol{A}_{t,C} = (\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+C-1})$ with $C \geq K$. For visual planning, we assume access to goal information $\boldsymbol{c}_t$ that contains the goal image embedding $\boldsymbol{z}_g$ and its normalized remaining horizon from $\boldsymbol{z}_t$.

The encoder consists of a ResNet backbone (He et al., 2016), spatial softmax pooling, and a multilayer perceptron (MLP) with two projection layers. The same encoder processes all the pixel observations: history, future target, and goal information. The predictor is comprised of a state branch f_z and an action branch f_a. Specifically,

$$\begin{aligned} \hat{\boldsymbol{z}}_{t+K} &= f_z(\boldsymbol{q}^z_{t+K}; \boldsymbol{z}_{\leq t}, \boldsymbol{A}_{t,K}), \\ \hat{\boldsymbol{v}}_{t,C} &= f_a(\boldsymbol{A}^{(s)}_{t,C}; s, \boldsymbol{z}_{\leq t}, \boldsymbol{c}_t) \end{aligned} \tag{1}$$

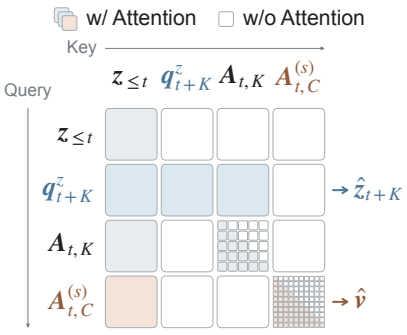


Figure 3: **MoT attention.** Filled cells permit attention, while white cells are masked. The state query reads history and clean actions, and noisy action tokens read state history and their own causal prefix. Small grids show causal attention. Arrows mark the state branch and action velocity branch.

where $\boldsymbol{A}^{(s)}_{t,C}$ is a noisy C-action chunk at flow timestep $s \in [0, 1]$ (see Equation 2) and $\boldsymbol{q}^z_{t+K}$ is a learned query token specifying the future state to predict. Integrating the predicted velocity field $\hat{\boldsymbol{v}}_{t,C}$ yields the predicted action chunk $\hat{\boldsymbol{A}}_{t,C}$.

To simultaneously balance latent-prediction and flow-matching objectives while allowing f_z and f_a to share information about states and actions, we use the Mixture-of-Transformers (MoT) (Liang et al., 2025) architecture. That is, each branch applies independent query, key, value, and feed-forward projections for its respective modality (latent states or actions) while sharing one masked self-attention. Figure 3 specifies the information available to each branch during attention. The state query $\boldsymbol{q}^z_{t+K}$ reads history latents and clean actions, and noisy action tokens $\boldsymbol{A}^{(s)}_{t,C}$ read history latents and their own causal prefix. History latents attend only to themselves, blocking indirect clean-action leakage to f_a. Goal information conditions only the noisy action tokens, leaving the dynamics unconditioned on either. Appendix B gives the full token-level mask, layer sizes, and training settings.

Training Objective. For flow matching (Lipman et al., 2023), given a recorded action chunk $\boldsymbol{A}_{t,C}$, sample s uniformly from $[0, 1]$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, then construct

$$\boldsymbol{A}^{(s)}_{t,C} = (1 - s)\boldsymbol{\epsilon} + s\boldsymbol{A}_{t,C}, \qquad \boldsymbol{v}_{t,C} = \frac{\mathrm{d}\boldsymbol{A}^{(s)}_{t,C}}{\mathrm{d}s} = \boldsymbol{A}_{t,C} - \boldsymbol{\epsilon} \tag{2}$$

The action branch regresses the path velocity $\boldsymbol{A}_{t,C} - \boldsymbol{\epsilon}$ from $\boldsymbol{A}^{(s)}_{t,C}$. The latent dynamics and action flow matching losses use mean squared error (MSE), averaged over the scalar coordinates of each prediction:

$$\begin{aligned} \mathcal{L}_{\mathrm{dyn}} &= \mathbb{E}_{\mathcal{D}}[\mathrm{MSE}(\hat{\boldsymbol{z}}_{t+K}, \boldsymbol{z}_{t+K})], \\ \mathcal{L}_{\mathrm{act}} &= \mathbb{E}_{\mathcal{D},s,\boldsymbol{\epsilon}}[\mathrm{MSE}(\hat{\boldsymbol{v}}_{t,C}, \boldsymbol{A}_{t,C} - \boldsymbol{\epsilon})]. \end{aligned} \tag{3}$$

We use SIGReg (Balestriero & LeCun, 2025) to preserve latent diversity through an isotropic Gaussian distributional constraint. Let $\boldsymbol{Z}$ collect the encoded history and target observations in a mini-batch. With M random unit directions $\boldsymbol{u}^{(m)}$ and the univariate Epps–Pulley (Epps & Pulley, 1983) test statistic T against $\mathcal{N}(0, 1)$, the regularizer is

$$\mathcal{L}_{\mathrm{sig}} = \frac{1}{M} \sum_{m=1}^{M} T\left(\boldsymbol{Z}\boldsymbol{u}^{(m)}\right). \tag{4}$$

With regularization weight λ_{sig}, the full objective is

$$\mathcal{L}_{\mathrm{LeWAM}} = \mathcal{L}_{\mathrm{dyn}} + \mathcal{L}_{\mathrm{act}} + \lambda_{\mathrm{sig}}\mathcal{L}_{\mathrm{sig}}. \tag{5}$$

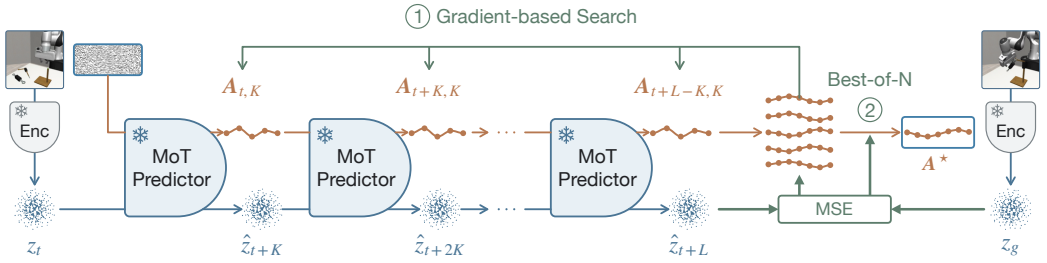


Figure 4: **Planning with LeWAM.** Each MoT predictor stage samples C actions from Gaussian noise and uses the first K, denoted $\boldsymbol{A}_{t,K}$, to advance the latent dynamics from t by K raw environment steps. Five-point curves depict action chunks ($K = 5$); stacked curves denote N sampled plans of length L. Step 1 refines all candidates through the frozen dynamics using the goal-state MSE. Step 2 selects the lowest-cost refined iterate, shown as $\boldsymbol{A}^\star$. Appendix C.3 gives the full procedure.

3.2 Efficient Control with LeWAM

LeWAM supports direct control through its learned policy and planning through imagined futures.

Direct Control. With history and optional goal context fixed, one C-action chunk is sampled by integrating along the learned velocity field over the interval $[0, 1]$ from Gaussian noise:

$$\frac{\mathrm{d}\boldsymbol{A}^{(s)}_{t,C}}{\mathrm{d}s} = f_a(\boldsymbol{A}^{(s)}_{t,C}; s, \boldsymbol{z}_{\leq t}, \boldsymbol{c}_t), \qquad \boldsymbol{A}^{(0)}_{t,C} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \hat{\boldsymbol{A}}_{t,C} = \boldsymbol{A}^{(1)}_{t,C} \tag{6}$$

The first K actions are executed and resampling begins from the resulting observation.

Planning. LeWAM uses a gradient-based refinement strategy over a population of action candidates sampled from the learned action distribution (Figure 4). To simplify notation, let $\boldsymbol{A} = (\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+L-1})$ denote a plan to reach the goal $\boldsymbol{z}_g$ after L raw environment steps from time t, where L is a multiple of frameskip K.

Starting from the observed history $\boldsymbol{z}_{\leq t}$, one population of N candidate plans is generated from the learned action distribution by alternating action sampling and latent dynamics. Specifically, at each step, Eq. (6) generates N action chunks from the observation history and goal. The forward dynamics branch uses the first K actions of each chunk to predict the N corresponding next states. Each predicted state is appended to the fixed-length history of its candidate, the planning horizon is updated, and the process continues for L total steps. This leaves N action sequence candidates $\{\boldsymbol{A}^{(n)}\}_{n=1}^{N}$. Temporally coherent policy rollouts provide structured starting points from which we assume that nearby improvements can be found via gradient descent through the latent dynamics (Henaff et al., 2017; Jyothir S V et al., 2023). Starting from $\{\boldsymbol{A}^{(n)}\}_{n=1}^{N}$, we optimize each candidate with respect to the latent cost-to-goal

$$\mathcal{C}(\boldsymbol{A}) = \mathrm{MSE}(\hat{\boldsymbol{z}}_{t+L}(\boldsymbol{A}), \boldsymbol{z}_g) \tag{7}$$

With learning rate η, the goal-cost update takes the gradient-descent form

$$\boldsymbol{a}_{t+k} \leftarrow \boldsymbol{a}_{t+k} - \eta \frac{\partial \mathcal{C}}{\partial \boldsymbol{a}_{t+k}}, \qquad k = 0, \ldots, L-1, \tag{8}$$

The descent iterate with the lowest $\mathcal{C}$ across all candidates is chosen. In practice, we add a small proximity penalty to discourage large changes from the sampled plans. Optionally, the first K actions of a plan are executed, and the entire two-stage process repeats until the goal is reached, as in receding-horizon model predictive control. Appendix C.3 details this implementation.

4 Experiments

Our experiments aim to address the following questions: (1) Can a single LeWAM architecture learn stably from offline data and support visual planning across heterogeneous tasks and settings? (2) Does LeWAM jointly learn informative action-conditioned latent dynamics, control-relevant representations, and multi-modal distributions over actions? (3) Can LeWAM further support behavior cloning in simulated environments and physical-robot control?

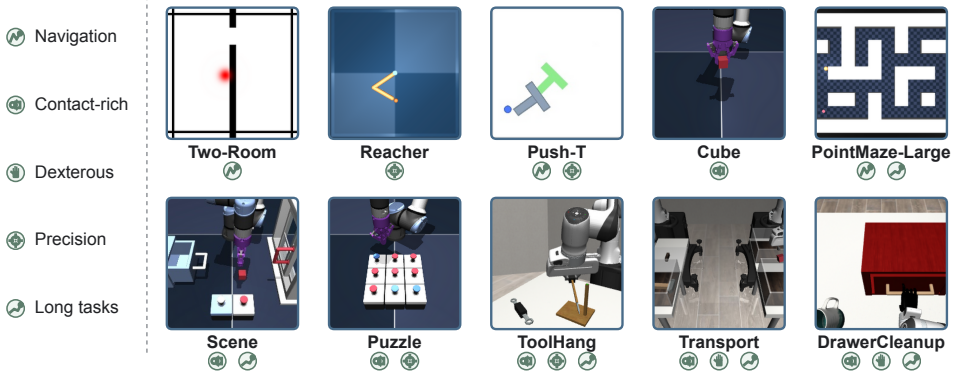


Figure 5: **Simulation environments.** Symbols beneath each frame denote overlapping task attributes, with their meanings shown in the legend to the left.

4.1 Evaluation Protocol

Heterogeneous Simulated Environments We evaluate control across ten simulated navigation and manipulation environments (Figure 5). To ensure the evaluation is reliable, the environments vary in task functionality, temporal horizon, and collection regime, which cover expert demonstrations, random-policy interactions, and mixed-quality data. Four environments adapt the benchmarks used in LeWM (Maes et al., 2026b). The other six add challenging contact-rich manipulation and navigation. Appendix C describes the tasks, data properties, construction procedure, and success criterion.

Evaluation Protocols. We distinguish two separate evaluation settings. Visual planning assumes access to a goal image, and a goal-conditioned policy or planner must reach the depicted state from its current (random) starting position within a fixed step budget. Task completion provides no goal image and evaluates behavior cloning from demonstrations: policies execute actions from start to finish to complete one specific task. Within each setting, methods share training and evaluation data, observations, action limits, goal offsets, execution budgets, and success predicates. Appendix C.2 details goal construction, evaluation conditions, and success criteria.

4.2 Visual Planning with LeWAM

We compare with four goal-conditioned baselines spanning latent-dynamics planning, an inverse-dynamics policy on a frozen world model, and direct policy execution:

- **LeWM (CEM)** (Maes et al., 2026b). Learning compact action-conditioned latent dynamics end-to-end with SIGReg, planning with CEM over learned latent dynamics.
- **LeWM (IDM)** (Nguyen et al., 2026). Post-trains a goal-conditioned IDM on frozen LeWM.
- **DINO-WM** (Zhou et al., 2024). Learns dynamics on a frozen DINOv2 encoder with CEM planner.
- **GCBC** (Ghosh et al., 2021). Trains a goal-conditioned policy on frozen DINOv2 features.

Figure 6 reports goal-reaching success across ten environments. Across the ten tasks, LeWAM averages 91.9% success versus 46.9% for LeWM (CEM) (Maes et al., 2026b), a 44.9 percentage-point gap. LeWAM exceeds that baseline on every task, with particularly large gaps on contact-rich `ToolHang` (94.7 versus 35.4) and `Transport` (83.3 versus 2.8). On the logically structured `Puzzle` task, LeWAM reaches 80.7% success versus 44.0% for goal-conditioned GCBC (Ghosh et al., 2021), highlighting the advantages of dynamics-based planning over model-free execution.

Long-horizon planning. We evaluate planning robustness with the same checkpoints, over varying horizons using four manipulation tasks with diverse visual goals: `Push-T`, `Cube`, `Scene`, and `Puzzle`. Unlike simple navigation benchmarks, these tasks introduce significantly higher behavioral diversity as the planning horizon grows. As shown in Figure 7, LeWAM exhibits less performance degradation than LeWM with increasing horizon length, highlighting its stability in long-horizon planning.

Planning speed. Appendix Table 8 compares the time for one planning call on four visual-goal tasks. At the reported solver settings, LeWAM averages 52.50 plans/s across tasks, versus 1.62 plans/s for LeWM (CEM) (Maes et al., 2026b) with a 32.3× acceleration, as shown in Figure 1(c).

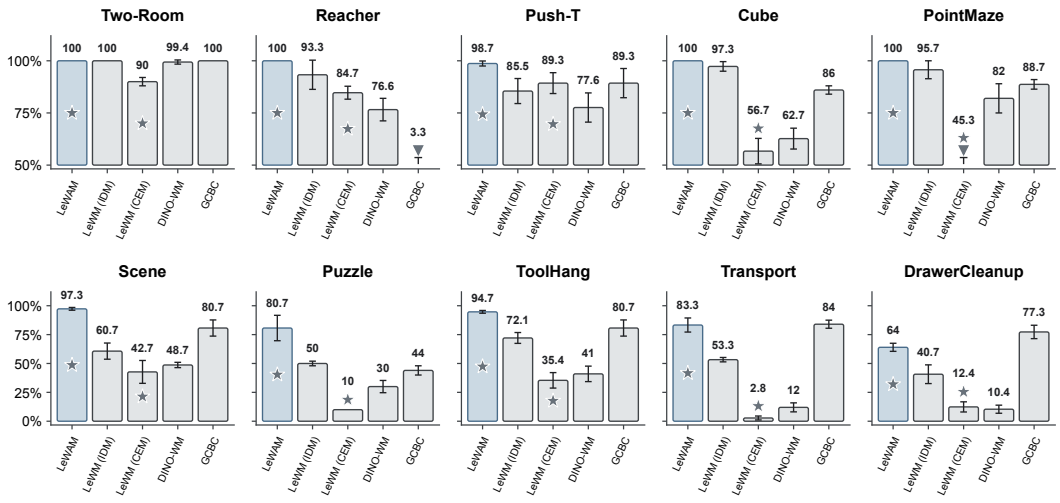


Figure 6: **Visual planning across ten simulated environments.** Each panel reports the success rate under a goal offset of $L = 25$ environment steps and permits $2L = 50$ actions across 3 seeds. Details are present in Appendix C.2. Error bars denote one standard deviation; ★ indicates models with end-to-end training. Linear axes start at 50% for the first five panels and 0% elsewhere.

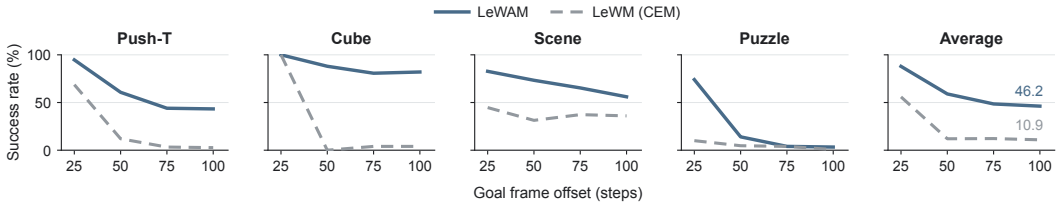


Figure 7: **Visual planning success across different horizons.** The task panels show mean success rates across 3 seeds. Each sweep starts at the first trajectory frame but with different goal offsets.

4.3 Understanding LeWAM's Dynamics, Representations, and Actions

Latent representation probing. To better measure what information is learned by our joint-training objective, we evaluate how well linear probes recover physical states, proprioceptive information, and actions from LeWAM's representations. As highlighted in Table 1 and Appendix D.2, training with an action modeling objective noticeably improves LeWAM's linear action and proprioception decodability compared to LeWM, increasing average proprioception, action, and IDM R^2 by $+0.21, +0.20, +0.19$ respectively (see Table 11, Appendix D.2).

The role of SIGReg. Furthermore, while SIGReg (Balestriero & LeCun, 2025; Maes et al., 2026b) is primarily used in JEPAs to prevent representation collapse during training, we observe that in our setup, LeWAM's joint prediction objective over latent states and actions is sufficient in preventing collapse without SIGReg or any other tricks such as variational constraints or stop-gradients (see Table 9, Appendix D.1). *However*, we observe that SIGReg itself is invaluable to planning: the planning gain on `Push-T` is 14.7 percentage points with SIGReg, compared to 2.7 without it (Table 9). The probing results of Table 1 and Table 11 demonstrate that, beyond ensuring anti-collapse, SIGReg improves accessibility of object information in the latent state. We hypothesize that this translates to improved planning by making plan-selection metrics such as latent distance-to-goal more sensitive to changes in the underlying environment state.

Figure 8(a) compares full LeWAM with its no-SIGReg ablation. Removing SIGReg lowers planning success in the three tasks shown, most strongly on `Push-T` (25.3 percentage points), followed by `Puzzle` (5.3) and `Scene` (4.0).

Why train the visual encoder end to end? Evaluated under the same downstream setup, joint encoder training improves direct control over a frozen DINOv2 (Oquab et al., 2023) encoder (Figure 8(b)). This suggests strong semantic representations are suboptimal for control, and that action

Table 1: Linear probing and the role of SIGReg. All linear probes report R^2 score using ridge regression. *Object* denotes the average R^2 over all object ground truth information and *Proprio* the ground truth proprioceptive state. *Action (IDM)* predicts the action chunk from latent z_t (both z_t, z_{t+K} for IDM). *Eff. rank* denotes effective rank (Roy & Vetterli, 2007).

Environment	Model	SIGReg	Eff. rank	Object ↑	Proprio ↑	Action (IDM) ↑
Push-T	LeWAM	Yes	42.90	0.90	**0.86**	**0.40(0.75)**
		No	8.20	0.69	**0.86**	0.38(0.73)
	LeWM	Yes	94.2	**0.95**	0.80	0.29(0.71)

prediction can help refine policy-friendly representations. Furthermore, planning improvements indicate that training the encoder from scratch still preserves useful action-conditioned dynamics for downstream planning.

Why model stochastic action distributions? We contrast action flow matching with sampling from a Gaussian centered at a deterministic action prediction obtained by training our action branch with standard MSE loss. Under the same dynamics objective, flow matching yields larger planning gains across tasks (Figure 8(c)). In this comparison, the learned stochastic action model provides more effective planning proposals than Gaussian perturbations around a point prediction. Appendix D.3 further demonstrates the ability of flow matching to learn multimodal action distributions in a two-door navigation problem.

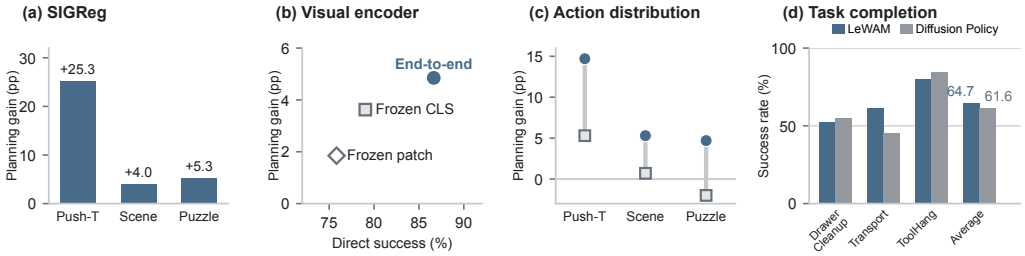


Figure 8: (a) Planning gain (percentage points) from SIGReg. (b) Direct success and planning gain for visual encoders, averaged over four tasks. (c) Planning gain for flow (circles) and MSE (squares) action heads. (d) Task-completion success for LeWAM and Diffusion Policy.

4.4 Behavior Cloning and Physical Robot Control

While visual planning relies on goal images to optimize actions online, goal-agnostic tasks require executing behaviors directly. In this section, the robot must implicitly infer task progress and react to changing object poses and contacts without explicit goal conditioning. We train LeWAM without goal context $\boldsymbol{c}_t$ while retaining other objectives. At test time, the policy directly samples action chunks from observation history for execution, bypassing the test-time planning.

Task completion in simulation. We benchmark LeWAM on three simulated tasks requiring complex manipulation without goal prompts: `DrawerCleanup`, `Transport`, and `ToolHang`. Figure 8(d) compares task-completion success with Diffusion Policy (Chi et al., 2023). LeWAM achieves competitive performance on two tasks and improves success by 16 percentage points on `Transport`. These results show that LeWAM supports goal-agnostic behavior cloning alongside goal-conditioned planning. Per-task standard deviations appear in Appendix Table 3.

Physical-robot control. Several prior JEPAs, such as DINO-WM (Zhou et al., 2024), PLDM (Sobal et al., 2025), and LeWM (Maes et al., 2026b), evaluate control only in simulation, while V-JEPA 2-AC (Assran et al., 2025) plans on physical robots over a frozen pretrained encoder. We evaluate LeWAM on a real robotic platform through direct policy execution, testing whether an end-to-end JEPA can handle real-world contact dynamics and visual variation in closed-loop control.

We evaluate both the end-to-end LeWAM and a variant that supplies pretrained features from a frozen V-JEPA 2.1 (Mur-Labadia et al., 2026) backbone on `Pick and Place T`, `Push-T`, and `Tape Stack`. The physical-robot baselines are Diffusion Policy (Chi et al., 2023) and $\pi_{0.5}$ (Physical Intelligence et al., 2025), a vision-language-action policy. All methods are trained or finetuned

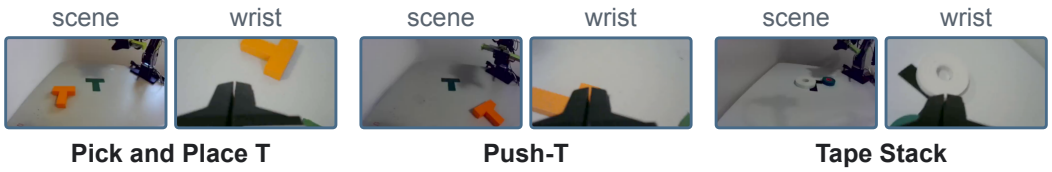


Figure 9: Paired scene and wrist view examples from three physical-robot tasks.

on our collected teleoperation data and directly execute the first half of each predicted action chunk during inference. Implementation details appear in Appendix E.

Table 2: Physical robot evaluation. Successful trials out of 30 on Pick and Place T, Push-T, and Tape Stack. GFLOPs count one inference for a full action chunk, including all denoising steps. One full LeWAM action chunk uses 270.2 GFLOPs versus 3603.6 GFLOPs for $\pi_{0.5}$. Figure 1(d) expresses the reciprocal as action chunks per TFLOP, a $13.3\times$ difference in operation-count efficiency.

Method	Pick and Place T	Push-T	Tape Stack	Average	GFLOPs
Diffusion Policy (Chi et al., 2023)	19/30	16/30	0/30	38.9%	270.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	14/30	3/30	9/30	28.9%	3603.6
LeWAM (Frozen V-JEPA 2.1 Encoder)	15/30	12/30	5/30	35.6%	270.2
LeWAM	16/30	8/30	6/30	33.3%	270.2

As seen in Table 2, our experiments show that LeWAM is competitive with both baselines, indicating that a JEPA-style world-modeling objective can be trained jointly with action prediction and deployed closed-loop on a *physical robot*. Notably, training the visual encoder end-to-end performs roughly the same as using a pretrained V-JEPA 2.1 encoder frozen across training and inference. This suggests end-to-end training can also learn useful representations for downstream control in a sample-efficient way, despite collapse pressure from the future latent prediction training objective.

Under a reconstruction probe on held-out episodes, both LeWAM variants decode rollouts to comparable visual quality, clearly tracking the manipulated object and robot's coarse motion while disregarding high-frequency detail (Appendix Figure 15).

5 Conclusion and Limitations

In this work we introduce LeWAM, an end-to-end JEPA for joint prediction of future latent states and actions, that supports control through both planning and policy execution. On goal-reaching environments, LeWAM demonstrates improved performance in dynamics-based planning over previous JEPA-based world models. On behavioral cloning from expert demonstrations on both simulated and physical robots, LeWAM is competitive with established policy baselines.

Limitations and future work. Our results do not establish robustness to distribution shift or transfer across tasks and embodiments. Our datasets and task suite are limited in scale; whether joint end-to-end training realizes larger benefits with substantially more diverse and multi-task data remains to be tested. Appendix A discusses these limits, data coverage, and future directions.

REPRODUCIBILITY STATEMENT

The experiments in Section 4 use the architectures and baseline configurations in Appendix B, with datasets and evaluation protocols in Appendix C. Anonymous links release all code, configurations, data-processing scripts, result logs, training data, evaluation recordings and results, and redistributable checkpoints.

ETHICS STATEMENT

The experiments collect no new human-subject data. Reused environments, datasets, models, and software remain subject to their original licenses; restricted assets will be accompanied by acquisition instructions. Physical deployment (Section 4.4) requires independent safety mechanisms.

AI USE STATEMENT

LLM-based assistants were primarily used to aid in manuscript polishing and language refinement. This work's research and manuscript were fully designed, overseen, and executed by its authors. All experimental results and claims were verified manually; the authors take full responsibility for the content of the paper.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

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

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 66953–66978, 2023. doi: 10.52202/075280-2925. URL https://doi.org/10.52202/075280-2925.

Lara Brudermüller, Brandon Hung, Xinghao Zhu, Jiuguang Wang, Nick Hawes, Preston Culbertson, and Simon Le Cleac'h. Generative models from and for sampling-based MPC: A bootstrapped approach for adaptive contact-rich manipulation. *arXiv preprint arXiv:2510.14643*, 2025. URL https://arxiv.org/abs/2510.14643.

Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Mustafa Shukor, Jess Moss, Alexander Soare, Dana Aubakirova, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. LeRobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Thomas W Epps and Lawrence B Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(3):723–726, 1983. doi: 10.1093/biomet/70.3.723.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

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

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Mikael Henaff, William F. Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Kevin Huang, Sahin Lale, Ugo Rosolia, Yuanyuan Shi, and Anima Anandkumar. CEM-GD: Cross-entropy method with gradient descent planner for model-based reinforcement learning. *arXiv preprint arXiv:2112.07746*, 2021. URL https://arxiv.org/abs/2112.07746.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930, 2025. doi: 10.1109/ICRA55743.2025.11127809.

Jyothir S V, Siddhartha Jalagam, Yann LeCun, and Vlad Sobal. Gradient-based planning with world models. *arXiv preprint arXiv:2312.17227*, 2023.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf. Position paper, version 0.9.2.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-Transformers: A sparse and scalable architecture for multi-modal foundation models. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2411.04996.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a.

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

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations (ICLR)*, 2023.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

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

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics*, 39 (6):4691–4711, 2023. doi: 10.1109/TRO.2023.3300230.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

Physical Intelligence, Kevin Black, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference (EUSIPCO)*, pp. 606–610, 2007. URL https://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/a5p-h05.pdf.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.

Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. In *NeurIPS Workshop on Self-Supervised Learning: Theory and Practice*, 2022. URL https://arxiv.org/abs/2211.10831.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

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

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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.

Haiyu Wu, Randall Balestriero, and Morgan Levine. VISReg: Variance-invariance-sketching regularization for JEPA training. *arXiv preprint arXiv:2606.02572*, 2026.

Jun Yamada, Shaohong Zhong, Jack Collins, and Ingmar Posner. D-Cubed: Latent diffusion trajectory optimisation for dexterous deformable manipulation. *arXiv preprint arXiv:2403.12861*, 2024. URL https://arxiv.org/abs/2403.12861.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

Appendix

*Supplementary material for **LeWAM: End-to-End World and Action Modeling with JEPAs***

Contents

A Limitations and Future Work p. 15

B Implementation Details p. 15

Baseline Implementations p. 16

System-Level Simplicity and Cost p. 19

C Benchmarks and Evaluation p. 19

Tasks and Datasets p. 19

Goals and Success Criteria p. 20

LeWAM Planning p. 22

Goal-Horizon Evaluation and Timing p. 23

D Additional Results p. 24

Component and Visual-Encoder Ablations p. 24

Multimodal Action Distributions p. 25

E Physical-Robot Setup and Evaluation p. 25

Platform and Data Collection p. 25

Model Implementation p. 26

Closed-Loop Evaluation p. 26

Representation Diagnostics and Evaluation p. 26

F Joint Prediction for Planning and Representation p. 29

A Limitations and Future Work

- **Data coverage.** Complex contact-rich manipulation still relies on expert demonstration data. Whether incorporating suboptimal interaction data improves dynamics and physical understanding remains unverified.
- **Representation scope.** Physical decodability does not ensure reliable control under distribution shift. Distinctions absent from both recorded actions and their consequences may be compressed; joint supervision does not exclude partial collapse or contextual shortcuts.
- **Single-task and single-embodiment learning.** Our model is trained from scratch on single-task, single-embodiment datasets. Extending LeWAM to multitask datasets across diverse embodiments and conditioning signals (*e.g.*, language) raises questions about the scalability of JEPA-based world action modeling and the learned representations.
- **Integration with hierarchical planning.** LeWAM provides low-level control and does not include a global high-level planner for very long-horizon tasks. Future work could integrate it with an external planner that decomposes such tasks into subgoal sequences, with LeWAM executing each subgoal through continuous control.

B Implementation Details

Temporal Sampling. Given observation-history stride Δ, the predictor in Section 3.1 uses two history latents $\boldsymbol{z}_{\leq t} = (\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t)$, a dynamics stride of $K = 5$ raw steps, and an action chunk of $C = 2K = 10$ raw steps. We use a stride of $\Delta = K$ for goal-reaching tasks and $\Delta = 1$ for task-completion frames. To predict $\boldsymbol{z}_{t+5}$, the dynamics branch receives all five actions $\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+4}$; the flow policy models all ten actions $\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+9}$. Only frames are subsampled. During imagined rollouts, the history update retains the two most recent latent states.

Inner Architecture. Unless a task-specific deviation is stated, the observation encoder is a ResNet-18-style network (He et al., 2016) trained from scratch. Its convolutional backbone uses GroupNorm with 16 groups. Input images are cropped from 224×224 to 202×202, with random offsets during training and a center crop at evaluation. A 1×1 convolution produces 32 spatial maps; spatial softmax pooling converts these to 32 expected normalized image-coordinate pairs. A two-layer projection MLP with BatchNorm maps these 64 features through a hidden width of 384 to a 192-dimensional latent state. The MoT predictor has width 192, four blocks, four attention heads, and dropout 0.1. Its token sequence contains two history latents, one learned state query, five clean action tokens, and ten action flow slots. Learned type and positional embeddings distinguish history, state query, clean action, and action flow tokens. The state readout uses LayerNorm followed by a linear map to the 192-dimensional latent, while the action readout maps each action flow slot to a velocity vector in raw-action space.

Goal information $\boldsymbol{c}_t$ is optional and is omitted for goal-free task completion. For goal-conditioned experiments, $\boldsymbol{c}_t$ contains the goal latent $\boldsymbol{z}_g$ and the normalized remaining horizon. The goal latent is supplied as an additional token in the action branch; only noisy action tokens can attend to it. The horizon h between the current timestep and the goal is normalized into a $[0, 1]$ range relative to a maximal value $H_{\max}$ which dictates the longest distance-from-goal seen during training. h modulates the noisy action tokens using AdaLN-Zero (Peebles & Xie, 2023) blocks with hidden width 512 and a 128-dimensional sinusoidal embedding.

In total, the model has approximately 16.9M trainable parameters. For the simulated experiments, action sampling uses eight Euler steps to balance speed and quality. The reported MSE dynamics configuration uses an online encoder target, no state noise, no state flow integration, no momentum target encoder, only SIGReg with 1,024 projections and 17 knots. For the training objective in Equation 5, we set $\lambda_{\text{sig}} = 0.04$.

Training Recipe in Simulated Environments. For goal-conditioned visual planning, LeWAM runs use a batch size of 128 and train for 50 epochs, with a base learning rate of 1.5×10^{-4} and five warmup epochs. Notably, in most environments, training for about 10 epochs can converge. The goal-free task-completion runs use a batch size of 64 and train for 120 epochs, with a base learning rate of 10^{-4} and ten warmup epochs. Both use AdamW with weight decay 10^{-4} and a uniform learning rate across the encoder and MoT predictor. The learning rate increases linearly during

warmup and then follows an epochwise cosine schedule. Training uses full precision, a gradient-norm clip of 1, and seed 42. We split valid decision points into 90% training and 10% validation, then sample training points with replacement within each epoch. After each epoch, we retain the checkpoint with the lowest validation action flow-matching loss, where task success is not used for checkpoint selection.

Tokens and Conditioning. With the optional goal context $\boldsymbol{c}_t$ omitted, the MoT sequence is

$$\left[\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t, \boldsymbol{q}^z_{t+K}, \boldsymbol{A}_{t,K}, \boldsymbol{A}^{(s)}_{t,C}\right]. \tag{9}$$

Here $\boldsymbol{A}_{t,K}$ and $\boldsymbol{A}^{(s)}_{t,C}$ expand into K clean and $2K$ noisy action tokens, respectively. The first three tokens are processed by f_z, and both action groups by f_a. Each MoT branch has separate query, key, value, output, normalization, and feedforward parameters, while one global masked self-attention operates over the complete token sequence. Flow timestep s modulates only the noisy action slots through AdaLN-Zero.

Attention Mask. Let $\mathcal{V}(x)$ denote the tokens visible to query x. The mask below describes the sequence without optional goal context $\boldsymbol{c}_t$. When this context is provided, the goal token is additionally visible to the noisy action tokens only. Action prefixes begin at the current chunk's start t; padded history keys are excluded:

$$\begin{aligned}
\mathcal{V}(\boldsymbol{z}_i) &= \{\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t\}, && i \in \{t-\Delta, t\},\\
\mathcal{V}(\boldsymbol{q}^z_{t+K}) &= \{\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t, \boldsymbol{q}^z_{t+K}, \boldsymbol{A}_{t,K}\},\\
\mathcal{V}(\boldsymbol{a}_j) &= \{\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t, \boldsymbol{a}_{\le j}\}, && j \in \{t, \dots, t+K-1\},\\
\mathcal{V}(\boldsymbol{a}^{(s)}_j) &= \{\boldsymbol{z}_{t-\Delta}, \boldsymbol{z}_t, \boldsymbol{a}^{(s)}_{\le j}\}, && j \in \{t, \dots, t+2K-1\}.
\end{aligned} \tag{10}$$

Clean action tokens therefore cannot relay target information to history or action flow tokens across layers. A visualization of the attention mask is provided in Figure 10.

B.1 Baseline Implementations

The baselines in Section 4.1 use the released Stable World Model codebase (Maes et al., 2026a), retaining its environment wrappers, dataset interface, action bounds, and evaluation loop. Unless stated below, each method follows its official implementation and hyperparameters.[1] These configurations describe goal-conditioned comparisons. Methods share observations, actions, start and goal pairs, and success predicates. We list deviations from released configurations.

- **LeWM (IDM).** This is the GC-IDM (Nguyen et al., 2026) configuration: a pretrained LeWM checkpoint trained by us is frozen, then a goal-conditioned inverse-dynamics model is post-trained. We follow its official two-stage implementation. Goal reaching samples a remaining horizon uniformly from 1 to 50 and clips it to the episode boundary.
- **LeWM (CEM).** We follow the official LeWM training and CEM evaluation settings (Maes et al., 2026b). On the four inherited task types, our reproduced checkpoints perform similarly to the released checkpoints but were unable to match the reported scores. All reported experiments using LeWM as a baseline use checkpoints trained by us. The `Reacher` data differs from the released random-policy collection, as detailed in Appendix C.1.
- **DINO-WM.** We evaluate the released patch-token model and a matched CLS-token variant; tokenization is their only architectural difference. The six added tasks use ten training epochs, while the original four retain their four-epoch checkpoints.
- **GCBC.** We use the goal-conditioned behavior-cloning baseline from the released codebase (Maes et al., 2026a), with its pretrained DINOv2 encoder frozen and the policy trained for 20 rather than 100 epochs. Short-horizon goal reaching samples a uniformly later frame from the same episode. At evaluation, the policy executes its predicted five-step action chunks without planning.

[1] Official repositories: DINO-WM, Stable World Model, and GC-IDM.

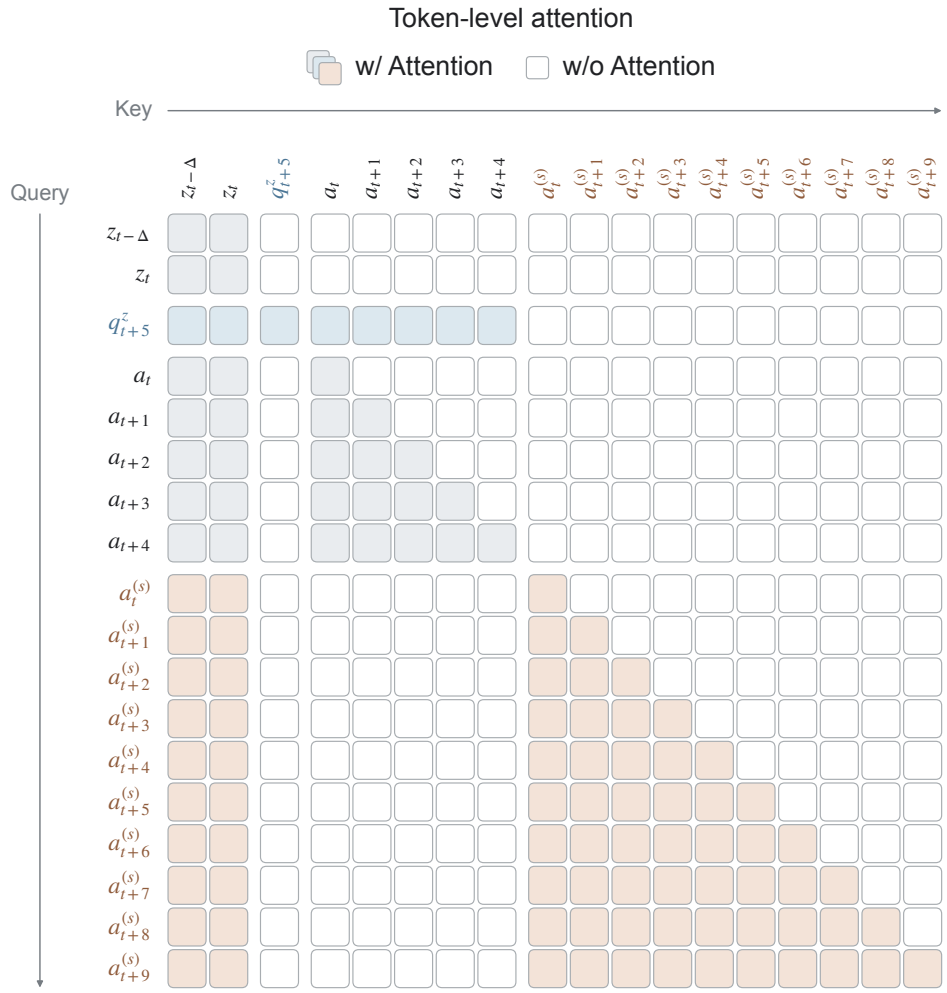


Figure 10: **Token-level MoT attention.** Expansion of Figure 3 for two history latents, one learned state query, five clean actions, and ten interpolated action tokens. Rows are queries; columns are keys. Filled cells permit attention; white cells are masked. Blue and brown mark latent state and action velocity prediction queries; gray marks context. Wider gaps separate token groups. Optional goal conditioning $\boldsymbol{c}_t$ is omitted. Δ is the history stride, and action indices remain at the raw environment rate.

Behavior-cloning baseline in simulation. The task-completion comparison in Table 3 uses a policy trained on demonstrations from each task, conditioned on observation history without a goal image or goal-horizon input.

- **Diffusion Policy** (Chi et al., 2023). We use the convolutional variant (DP-C), with a ResNet-18 encoder, GroupNorm, and spatial softmax for each camera. The image features condition a one-dimensional U-Net with channel widths $(512, 1024, 2048)$, kernel size five, and diffusion-step embedding dimension 128. Training minimizes noise-prediction MSE with a 100-step squared-cosine diffusion schedule; inference uses 100 DDPM steps. The policy takes two consecutive observations, predicts a 16-action sequence, and executes eight actions starting at the current timestep before resampling.

`ToolHang` uses the scene camera; `DrawerCleanup` and `Transport` use the scene camera and both wrist cameras. Inputs are 224×224 RGB images, with random 202×202 crops during training and center crops at evaluation, normalized to $[-1, 1]$. The policy uses visual observations only. Training windows advance at the raw frame rate, with boundary frames and actions repeated

for padding. We randomly split windows into 90% training and 10% validation using seed 42. Each policy trains for 120 epochs with batch size 64 and AdamW, learning rate 10^{-4}, betas $(0.95, 0.999)$, and weight decay 10^{-6}. The learning rate follows cosine decay after 500 warmup updates. Actions retain their recorded scale for `ToolHang` and `Transport`; `DrawerCleanup` uses min-max scaling to $[-1, 1]$. We select the checkpoint with the lowest validation denoising loss and evaluate its raw weights without exponential moving averaging. Each checkpoint is evaluated with seeds 0, 1, and 42, using 50 episodes per seed.

Table 3: Task completion. Mean success rates (%) $\pm$ standard deviation across three evaluation seeds, with 50 episodes per seed. Bold marks the better result.

Method	DrawerCleanup	Transport	ToolHang	Average
Diffusion Policy (Chi et al., 2023)	$\mathbf{54.7 \pm 1.2}$	45.3 ± 9.0	$\mathbf{84.7 \pm 3.1}$	61.6
LeWAM	52.7 ± 5.0	$\mathbf{61.3 \pm 9.9}$	80.0 ± 3.5	**64.7**

Behavior-cloning baselines on physical robots. For the comparison in Section 4.4, both baselines use the LeRobot implementations (Cadene et al., 2026), adapted to the scene and wrist cameras and seven-dimensional joint-target actions. Each policy returns 36 future actions and executes the first 18 at 30 Hz before requesting a new chunk, matching the LeWAM variants. The returned and executed chunks span 1.2 and 0.6 seconds, respectively. Appendix E describes the demonstrations and evaluation procedure, with 30 trials per task and method.

The `Pick and Place T` and `Push-T` training configurations resize both camera streams to 256×256 and retain seven joint torques from the recorded proprioceptive state. Actions and torques are normalized by mapping their training-set first and 99th percentiles to -1 and 1, respectively. Training examples start every three recorded frames; the observation offsets and action targets retain their 30 Hz timing. We hold out 10% of episodes using split seed zero. The configurations specify 20,000 optimizer updates with effective batch size 256, a 500-update linear learning-rate warmup, and cosine decay. Validation runs every 200 updates; the trainer saves both the final checkpoint and the checkpoint with the lowest validation loss.

- **Diffusion Policy** (Chi et al., 2023). The robot implementation uses a separate ImageNet-initialized ResNet-18 with spatial softmax for each camera and a conditional one-dimensional U-Net with channel widths $(336, 672, 1344)$. It conditions on two consecutive image and torque observations. Images are randomly cropped to 243×243 during training and center-cropped at evaluation. The internal diffusion horizon is 40 action positions, aligned from one step before the current observation; the policy returns positions corresponding to the next 36 actions. This 40-step internal sequence satisfies the U-Net's temporal downsampling constraint. Training uses noise-prediction MSE with a squared-cosine schedule, and inference uses 100 DDPM steps. AdamW uses learning rate 10^{-4}, betas $(0.95, 0.999)$, and weight decay 10^{-6}. Exponential moving averaging with decay 0.999 supplies the weights used for validation and saved checkpoints.
- $\pi_{0.5}$ (Physical Intelligence et al., 2025). The policy conditions on the current scene and wrist images together with the task instruction and current torque vector. Its image frontend resizes the inputs to 224×224. The wrapper discretizes the normalized torque values into 256 bins and includes them in the language prompt. The action expert predicts a 36-action chunk by flow matching, using ten Euler steps at inference. The expert-only configuration freezes the vision and language backbone while training the action expert and action/time projections. Its task configurations use AdamW with learning rate 2.5×10^{-5} and weight decay 10^{-4}, and gradient accumulation to reach effective batch size 256.

B.2 System-Level Simplicity and Cost

Table 4 complements Section 2 with a comparison of training requirements and prediction capabilities.

Table 4: **Modeling and control capabilities.** FDM denotes forward dynamics conditioned on clean actions; Policy denotes learned action prediction. Entries describe reported training recipes or the goal-reaching configurations used here.

Method	FDM	Policy	Joint FDM and Policy	End-to-end from scratch
World models				
DINO-WM (Zhou et al., 2024)	✓	✗	✗	✗
V-JEPA 2-AC (Assran et al., 2025)	✓	✗	✗	✗
LeWM (CEM) (Maes et al., 2026b)	✓	✗	✗	✓
World action models				
UVA (Li et al., 2025)	✓	✓	✗	✗
UWM (Zhu et al., 2025)	✓	✓	✗	✗
DreamZero (Ye et al., 2026)	✗	✓	✗	✗
Fast-WAM (Yuan et al., 2026)	✗	✓	✗	✗
Cosmos 3 (NVIDIA, 2026)	✓	✓	✗	✗
LeWAM (ours)	✓	✓	✓	✓
Additional goal-reaching baselines				
LeWM (IDM) (Nguyen et al., 2026)	✓	✓	✗	✗
GCBC (Ghosh et al., 2021)	✗	✓	✗	✗

Joint requires clean-action FDM and policy losses on the same sample in one training pass. From scratch includes visual encoding and prediction without pretrained or frozen components. UWM uses a frozen pretrained variational autoencoder even when its robot backbone is trained from scratch.

C Benchmarks and Evaluation

C.1 Tasks and Datasets

Task selection. We retain the four environments used by LeWM (Maes et al., 2026b) for direct comparison, then add six tasks that broaden navigation and manipulation to larger mazes, interacting objects, coupled button states, precise tool placement, bimanual transfer, and articulated drawers (Figure 5). The ten tasks also span different action dimensions, observation views, trajectory lengths, and data-collection regimes. Together, they test the same model design under varied control and data conditions.

Data provenance. `DrawerCleanup` and `Transport` use generated demonstrations from `DexMimicGen` (Jiang et al., 2025). `ToolHang` uses human demonstrations from `robomimic` (Mandlekar et al., 2022). These task-focused behavior-cloning datasets may offer limited diversity of sampled visual goals. We include them in the visual-planning comparison to probe complex contact-rich behaviors at higher action dimensionality: 7, 14, and 24 raw action dimensions for `ToolHang`, `Transport`, and `DrawerCleanup`, respectively (Table 5). `OGBench` (Park et al., 2025) supplies `Cube`, `PointMaze-Large`, `Scene`, and `Puzzle`. The released Stable World Model (Maes et al., 2026a) `Scene` collector renders one view of `swm/OGBScene-v0` and uses a noisy, closed-loop expert oracle. `Puzzle` uses the 3×3 single-view task. `Push-T` uses non-expert interactions. The released Stable World Model (Maes et al., 2026a) `Reacher` collector samples independent uniform actions. In this paper, the `Reacher` row instead describes separately collected goal-directed trajectories in a visible-target environment. The original random-policy trajectories average 201 frames. We keep the random, goal-directed, and equally mixed collections distinct.

Table 5 summarizes all ten tasks, including observation views, raw action dimensions, and dataset scale. Eight rows have audited source counts; `Scene` and `Puzzle` use author-reported statistics. Counts for the random and mixed `Reacher` variants are not included. Recorded lengths describe the data, while Appendix C.2 specifies evaluation horizons and execution budgets. Recorded duration uses the native sampling rate before temporal subsampling; synchronized views of the same trajectories are counted once. `ToolHang` uses one oblique scene view (about $45°$), while `Transport` and `DrawerCleanup` use a scene view and both wrist views.

Table 5: **Simulation benchmark datasets.** Ten tasks in Figure 5 order. Views gives camera-stream count; Dim. is the raw action dimension. Statistics cover complete source datasets before splits. Eight rows are audited; `Scene` and `Puzzle` are author reported. Total hours are computed before rounding per-task values.

Task	Collection regime	Views	Dim.	Traj.	Mean frames	Total frames	Hz	Hours
`Two-Room`	Noisy heuristic	1	2	10,000	92	920,809	10	25.58
`Reacher`	Goal-directed policy	1	2	17,504	143	2,498,208	25	27.76
`Push-T`	Non-expert	1	2	18,685	125	2,336,736	10	64.91
`Cube`	Scripted expert	1	5	10,000	201	2,010,000	20	27.92
`PointMaze-Large`	Scripted navigation	1	2	1,000	1,000	1,000,000	10	27.78
`Scene`	Scripted expert	1	5	10,024	200	2,004,800	20	27.84
`Puzzle` (3×3)	Scripted expert	1	5	10,024	200	2,004,800	20	27.84
`ToolHang`	Human demos	1	7	200	480	95,962	20	1.33
`Transport`	Generated demos	3	14	1,029	408	419,712	20	5.83
`DrawerCleanup`	Generated demos	3	24	1,026	291	298,235	20	4.14
Reported total (10)				79,492		13,589,262		240.93

C.2 Goals and Success Criteria

For the comparisons in Section 4.1, unless stated otherwise, we report mean success and standard deviation over three seeds with 50 episodes per seed. Methods within a protocol share observations, action limits, execution budgets, and success predicates.

Goal-conditioned visual planning. In Section 4.2, the initial state is sampled from an offline trajectory and the frame 25 steps later supplies the visual goal. Each controller receives 50 environment actions across all ten environments. We report first-hit success for comparison with the released LeWM evaluation (Maes et al., 2026b); the `Cube` predicate correction is detailed below. `Scene` and `Puzzle` follow the same protocol; their random-policy and zero-action controls appear alongside those for the other environments in Table 6.

Policy-floor controls. Table 6 also reports zero-action and random-policy success under the full-trajectory task-completion evaluation. These blind controls cover the benchmark environments; the learned task-completion policies in Section 4.4 are compared on `DrawerCleanup`, `Transport`, and `ToolHang` only.

Cube success predicate. Specifically, we add an end-effector (EEF) check to the `Cube` success criterion to exclude spurious successes. The original `Cube` criterion checks whether the cube center lies within 0.04 m of the target, regardless of the gripper pose. For 25-step goals, this credits zero action on 42.7% of episodes (Table 6). The corrected criterion additionally requires the end effector to lie within 0.04 m of its goal-frame position, reducing this floor to 2.7%. Figure 11 illustrates the object-only failure case.

Environment	Raw action dim.	25-step goal reaching		Task Completion	
		Zero-action	Random	Zero-action	Random
Two-Room	2	7.3 ± 0.9	25.3 ± 5.0	0.0 ± 0.0	2.7 ± 2.5
Reacher	2	3.3 ± 1.9	12.0 ± 1.6	0.0 ± 0.0	12.0 ± 4.9
Push-T	2	0.0 ± 0.0	1.3 ± 0.9	0.0 ± 0.0	0.0 ± 0.0
Cube	5	42.7 ± 6.6	44.0 ± 1.6	4.7 ± 2.5	6.0 ± 1.6
Cube (+EEF)	5	2.7 ± 0.9	4.7 ± 0.9	4.7 ± 2.5	6.0 ± 1.6
PointMaze	2	6.0 ± 0.0	9.3 ± 1.9	0.7 ± 0.9	6.0 ± 3.3
Scene	5	2.7 ± 2.3	2.7 ± 2.3	0.0 ± 0.0	0.0 ± 0.0
Puzzle	5	0.7 ± 1.2	0.7 ± 1.2	0.0 ± 0.0	0.0 ± 0.0
ToolHang	7	22.7 ± 2.5	26.0 ± 1.6	0.0 ± 0.0	0.0 ± 0.0
Transport	14	3.3 ± 0.9	2.7 ± 0.9	0.0 ± 0.0	0.0 ± 0.0
DrawerCleanup	24	0.0 ± 0.0	6.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

Darker shading indicates a higher control success rate.

Table 6: **Action dimensions and policy-floor controls.** Rows cover all ten environments in Figure 5, plus a `Cube` (+EEF) row isolating the corrected end-effector predicate. Raw action dimensionality is reported beside zero-action and random-policy success rates (%) for short-horizon goal reaching and full-trajectory task completion, averaged over three seeds. These are control floors, not additional learned-policy experiments. `Cube` has the same task-completion values in both rows because the end-effector correction applies only to short-horizon goal reaching. `DrawerCleanup` exposes 24 raw coordinates; the policy predicts 22 active dimensions after removing two constant joints. `PointMaze` uses the `PointMaze-Large` layout.



Figure 11: **A degenerate `Cube` success under short-horizon goal reaching.** Each row shows a policy rollout (left, frames over time) beside its goal frame (far right, after the divider). Throughout the execution the gripper never reaches the goal end-effector pose, yet the cube stays within the 0.04 m object-only threshold of the goal, so OGBench's object-position-only predicate scores the episode a success. Our end-effector term (Table 6) rejects these degenerate successes by additionally requiring the gripper to reach the goal pose.

C.3 LeWAM Planning

Optimization details. The goal cost $\mathcal{C}$ is defined in Eq. (7). During refinement, model parameters remain frozen and only the normalized action plans receive gradients. For each sampled plan $\boldsymbol{A}^{\text{sample}}$, Adam minimizes the goal cost with a quadratic proximity penalty:

$$\mathcal{J}_{\text{prox}}(\boldsymbol{A}; \boldsymbol{A}^{\text{sample}}) = \mathcal{C}(\boldsymbol{A}) + \lambda_{\text{prox}} \left\| \boldsymbol{A} - \boldsymbol{A}^{\text{sample}} \right\|_F^2 . \tag{11}$$

The penalty affects the updates, while candidate retention and final selection use the unpenalized goal cost $\mathcal{C}$. We clip normalized actions to $[-a_{\max}, a_{\max}]$ after each update.

Algorithm 1 Candidate refinement and selection for LeWAM planning

Require: $\boldsymbol{z}_{\le t}, \boldsymbol{c}_t, \boldsymbol{z}_g, L, N, J, \eta, \lambda_{\text{prox}}, a_{\max}$

Sample candidates from the learned policy

$\boldsymbol{A}^{\text{sample}}_{1:N} \leftarrow N$ policy rollouts from $(\boldsymbol{z}_{\le t}, \boldsymbol{c}_t)$ for L/K blocks

$c_{1:N} \leftarrow \mathcal{C}(\boldsymbol{A}^{\text{sample}}_{1:N})$

$\boldsymbol{A}^{\star}_{1:N} \leftarrow \boldsymbol{A}^{\text{sample}}_{1:N}, \quad \boldsymbol{A}_{1:N} \leftarrow \boldsymbol{A}^{\text{sample}}_{1:N}$

$c^{\star}_{1:N} \leftarrow c_{1:N}$

Refine each candidate

for $j = 0, \ldots, J-1$ **do**

  $\boldsymbol{A}_{1:N} \leftarrow \text{AdamStep}\left(\boldsymbol{A}_{1:N}, \nabla_{\boldsymbol{A}_{1:N}} \mathcal{J}_{\text{prox}}(\boldsymbol{A}_{1:N}; \boldsymbol{A}^{\text{sample}}_{1:N}), \eta\right)$

  $\boldsymbol{A}_{1:N} \leftarrow \text{clip}(\boldsymbol{A}_{1:N}, -a_{\max}, a_{\max})$

  $c_{1:N} \leftarrow \mathcal{C}(\boldsymbol{A}_{1:N})$

  $\boldsymbol{A}^{\star}_n \leftarrow \boldsymbol{A}_n, \quad c^{\star}_n \leftarrow c_n \quad \forall n$ such that $c_n < c^{\star}_n$

end for

Select the lowest-cost iterate

$n^{\star} \leftarrow \arg\min_{n=1,\ldots,N} c^{\star}_n$

return $\boldsymbol{A}^{\star}_{n^{\star}}$

Unless specified otherwise, Adam uses learning rate 0.05, gradient-norm clipping at 10, $a_{\max} = 3$, and $\lambda_{\text{prox}} = 0.01$.

Planner sensitivity. In the `Push-T` sweep in Figure 12, success plateaus after five refinement updates while latency continues to rise. Increasing the number of sampled candidates beyond eight gives no consistent success gain in this sweep.

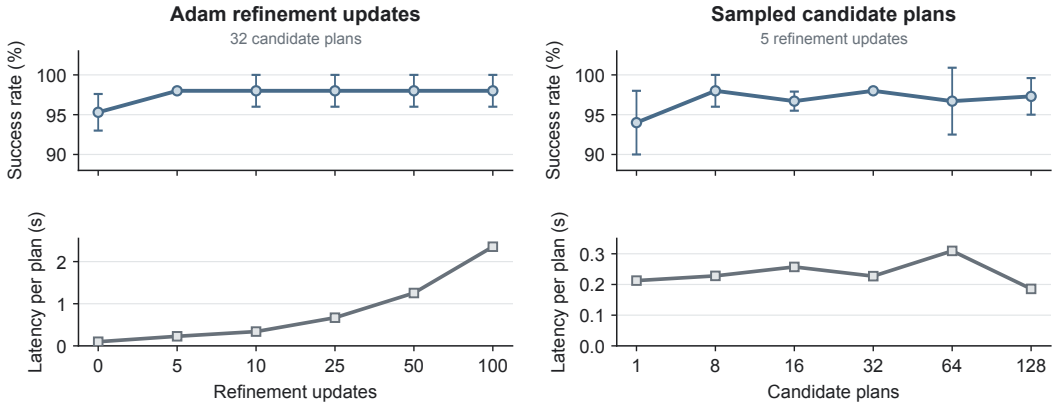


Figure 12: **Planner sensitivity on `Push-T`.** Goal-reaching success (mean and standard deviation across three seeds, 50 episodes each) and per-plan latency versus (left) Adam refinement updates with 32 candidate plans and (right) sampled candidate plans with five refinement updates.

C.4 Goal-Horizon Evaluation and Timing

Goal-horizon sweep. Table 7 gives the measurements in Figure 7 (Section 4.2). On `Push-T`, LeWAM uses SIGReg and the sampling and gradient-refinement procedure in Section 3.2; LeWM (CEM) (Maes et al., 2026b) uses a checkpoint trained by us with the reported 300×30 CEM configuration. For `Push-T`, the goal is the recorded frame 25, 50, 75, or 100 steps after the initial frame. Each episode starts at frame 0, receives twice the goal horizon in planning budget (total actions executed), and replans every 25 steps. These are the reported `Push-T` solver configurations; matched-compute comparisons remain pending. `Cube` uses a separately reported horizon sweep with five gradient steps and a fixed starting point. Its 25-step LeWM (CEM) result differs from the sampled-start evaluation in Figure 6, so the two protocols should not be conflated. All four Cube horizon points now have reported means and standard deviations. For `Scene`, the author-reported sweep gives four goal offsets; its start is the first trajectory frame, while its action budget and replanning interval are not specified in the supplied result summary. For `Puzzle`, both reported sweeps cover the same four goal offsets from the first trajectory frame; LeWM (CEM) uses the same baseline settings as the other evaluations.

Table 7: **Goal-reaching success across horizons.** Success rate (%), mean $\pm$ standard deviation. Measured values for Figure 7. The `Cube` horizon sweep is distinct from the sampled-start 25-step evaluation in Figure 6.

Task	Method	Goal horizon (steps)			
		25	50	75	100
`Push-T`	LeWAM	94.7 ± 3.1	60.7 ± 3.1	44.0 ± 3.5	43.3 ± 4.6
	LeWM (CEM)	68.7 ± 3.1	12.0 ± 4.0	3.3 ± 2.3	2.7 ± 1.2
`Cube`	LeWAM	100.0 ± 0.0	88.0 ± 2.0	80.7 ± 2.3	82.0 ± 2.0
	LeWM (CEM)	100.0 ± 0.0	0.0 ± 0.0	4.0 ± 3.5	4.0 ± 3.5
`Scene`	LeWAM	82.7 ± 3.1	73.3 ± 8.1	65.3 ± 3.1	56.0 ± 4.0
	LeWM (CEM)	44.7 ± 13.0	31.3 ± 2.3	37.3 ± 5.8	36.0 ± 6.0
`Puzzle`	LeWAM	74.0 ± 12.5	14.0 ± 3.5	4.0 ± 3.5	3.3 ± 1.2
	LeWM (CEM)	10.0 ± 0.0	4.7 ± 2.3	4.0 ± 2.0	0.7 ± 1.2

Timing scope. Table 8 reports time per planner call. Figure 1(c) averages the reciprocal throughput of each of its four tasks, using $1000/\text{milliseconds}$ per plan to obtain plans/s. The compared settings are 300×30 CEM for LeWM (Maes et al., 2026b) and five gradient steps for LeWAM. Hardware and the exact timing boundaries remain to be documented.

Table 8: **Planning speed across four visual-goal tasks.** Each plan is one planner call. LeWM (CEM) (Maes et al., 2026b) uses 300×30 CEM; LeWAM uses five gradient steps. Action dimensions are raw environment action dimensions. Throughput is the reciprocal of time per plan. The mean row averages per-task throughput, and its ratio compares those two means.

		LeWM (CEM)		LeWAM		
Task	Action dim.	ms/plan	plans/s	ms/plan	plans/s	Speedup
`Scene`	5	705	1.42	18.8	53.19	37.5×
`Puzzle`	5	731	1.37	16.3	61.35	44.8×
`Cube`	5	771	1.30	21.1	47.39	36.5×
`Push-T`	2	414	2.42	20.8	48.08	19.9×
Task mean	–	–	1.62	–	52.50	32.3×

D Additional Results

D.1 Component and Visual-Encoder Ablations

Table 9 records the SIGReg and action-head comparisons in Figure 8(a,c). These comparisons use epoch 30 checkpoints. Table 10 reports the separate encoder comparison in Figure 8(b). Direct control executes the goal-conditioned policy without search. Planning performs test-time action search through the learned dynamics. The encoder panel reports equal-weight averages over `Push-T`, `Cube`, `Scene`, and `Puzzle`.

Table 9: **SIGReg and action-head ablations.** Direct and planning success rates (%) at the listed checkpoint epochs. Gain is planning minus direct success in percentage points. Standard deviations are reported where available; none is inferred for gains.

Environment	Variant	Direct	Planning	Gain (pp)
`Push-T`	LeWAM	82.7 ± 8.1	97.3 ± 3.1	$+14.7$
	w/o SIGReg	69.3 ± 10.1	72.0 ± 3.5	$+2.7$
	MSE action head	90.0 ± 4.0	95.3 ± 1.2	$+5.3$
`Scene`	LeWAM	92.0 ± 5.3	97.3 ± 1.2	$+5.3$
	w/o SIGReg	91.3 ± 2.3	93.3 ± 3.1	$+2.0$
	MSE action head	89.3 ± 4.2	90.0 ± 2.0	$+0.7$
`Puzzle`	LeWAM	76.0 ± 7.2	80.7 ± 11.0	$+4.7$
	w/o SIGReg	67.3 ± 9.5	75.3 ± 9.0	$+8.0$
	MSE action head	76.7 ± 6.1	74.7 ± 6.1	-2.0
`Cube`	LeWAM	99.3 ± 1.2	100.0	$+0.7$
	w/o SIGReg	99.3 ± 1.2	99.3 ± 1.2	0.0
	MSE action head	98.0 ± 2.0	99.3 ± 1.2	$+1.3$

Table 10: **Visual encoders for direct control and planning.** Success rate (%) with the same goal-conditioned evaluation protocol for all encoder variants. Direct executes the policy without search; planning uses test-time action search. Gain is planning minus direct success in percentage points. Averages are equal-weight means over the four tasks; gains use unrounded means.

Visual encoder	Mode	`Push-T`	`Cube`	`Scene`	`Puzzle`	Average
LeWAM (end-to-end)	Direct	85.3 ± 5.0	100.0 ± 0.0	88.0 ± 6.9	73.3 ± 5.8	86.7
	Planning	98.7 ± 1.2	100.0 ± 0.0	93.3 ± 3.1	74.0 ± 12.5	91.5
	Gain (pp)	$+13.4$	0.0	$+5.3$	$+0.7$	$+4.9$
frozen DINOv2 CLS	Direct	72.7 ± 7.6	98.7 ± 1.2	82.0 ± 4.0	62.7 ± 7.0	79.0
	Planning	78.7 ± 6.4	99.3 ± 1.2	91.3 ± 1.2	61.3 ± 13.0	82.7
	Gain (pp)	$+6.0$	$+0.6$	$+9.3$	-1.4	$+3.6$
frozen DINOv2 patch mean	Direct	66.0 ± 12.5	93.3 ± 2.3	84.0 ± 2.0	60.0 ± 7.2	75.8
	Planning	67.3 ± 8.1	98.7 ± 1.2	86.0 ± 4.0	58.7 ± 7.6	77.7
	Gain (pp)	$+1.3$	$+5.4$	$+2.0$	-1.3	$+1.9$

D.2 Probing Latent Representations

Table 11 extends the probing results presented on `Push-T` in Section 4.3 to `Cube`, `Scene`, `Puzzle`. All probes report R^2 from ridge regression on a $4000/1000$-frame train/test split. *Object* is the mean R^2 over object ground-truth variables and *Proprio* over proprioceptive variables. *Action (IDM)* predicts the action chunk from z_t (and from $[z_t, z_{t+K}]$, in parentheses). *Eff. rank* is the effective rank measure introduced in (Roy & Vetterli, 2007).

Table 11: **Effective rank and linear probing** of object, proprio, and action information across `PushT`, `Cube`, `Scene`, `Puzzle`. Action-prediction improves LeWAM's proprioceptive and action awareness relative to LeWM, while SIGReg helps LeWAM preserve more object and general state information compared to the same model trained without it.

Environment	Model	SIGReg	Eff. rank	Object ↑	Proprio ↑	Action (IDM) ↑
`PushT`	LeWAM	Yes	42.9	0.90	**0.86**	**0.40** (**0.75**)
		No	8.2	0.69	**0.86**	0.38 (0.73)
	LeWM	Yes	94.2	**0.95**	0.80	0.29 (0.71)
`Cube`	LeWAM	Yes	46.6	**0.99**	0.67	**0.37** (0.66)
		No	8.0	**0.99**	**0.70**	**0.37** (**0.68**)
	LeWM	Yes	108.1	**0.99**	0.58	0.17 (0.55)
`Scene`	LeWAM	Yes	48.9	**0.48**	**0.78**	0.39 (**0.74**)
		No	12.5	0.40	0.77	**0.43** (**0.74**)
	LeWM	Yes	111.1	0.40	0.37	0.07 (0.35)
`Puzzle`	LeWAM	Yes	46.5	0.37	**0.70**	**0.33** (**0.69**)
		No	5.8	0.22	0.66	0.31 (0.68)
	LeWM	Yes	97.4	**0.60**	0.42	0.16 (0.48)

D.3 Multimodal Action Distributions

Figure 13 compares demonstrations in a two-door navigation task with trajectories from the flow and deterministic MSE action heads.

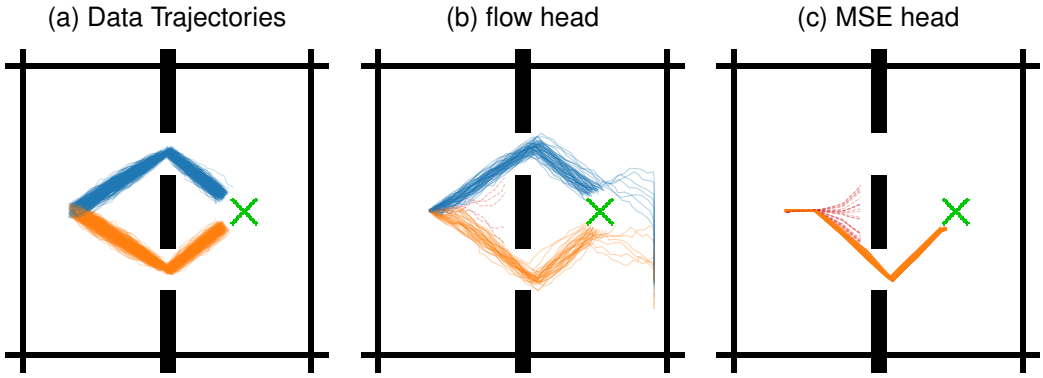


Figure 13: **Action flow matching represents multi-modality.** Demonstrations pass through the upper or lower door (a). Samples from the flow action head cover both routes (b), whereas the deterministic MSE action head concentrates on the lower route and produces some wall collisions (c). Black segments mark walls, and the green cross marks the goal. Blue and orange indicate upper- and lower-door trajectories; dashed red paths end in collision.

E Physical-Robot Setup and Evaluation

This section documents the physical-robot experiments in Section 4.4.

E.1 Platform and Data Collection

Demonstrations are collected by joint-space teleoperation from a reBot Arm 102 leader to a reBot B601-DM follower through LeRobot (Cadene et al., 2026). The follower uses impedance control with gravity feedforward. One proficient operator collects successful demonstrations without intentional exploration. Between episodes, both arms return to the calibration zero pose before recording resumes.

A ZED-Mini records the scene at 672×376 pixels per eye, and a wrist-mounted Logitech C270 records at 640×360 pixels. The model uses the left scene view and wrist view, both cropped to 256×256 pixels; the right stereo view is recorded but unused. At 30 Hz, a Raspberry Pi 5 stores the three video streams, seven-dimensional joint targets as actions, and follower position, torque, and velocity as a 21-dimensional state.

Table 12: Datasets for the physical-robot tasks in Section 4.4.

Task	Episodes	Frames	Duration	Task Description
Pick and Place T	200	52,297	29 min	"Pick up the T block and place it onto the outline."
Push-T	100	61,996	34 min	"Push the T block onto the outline."
Tape Stack	100	23,726	13 min	"Stack the green roll of tape on the white roll of tape."

E.2 Model Implementation

The LeWAM models in Section 4.4 use the state MSE and action flow matching objectives from Section 3.1. In the frozen V-JEPA 2.1 variant, attentive pooling and a BatchNorm MLP reduce each camera frame to one summary token. Proprioceptive state and previous actions are linearly projected from raw vectors to the model dimension. Learned queries initialize the future-camera and action streams.

The real world LeWAM variants take two video context frames and predict three future frames, all sampled at 2.5 Hz. They use model width 512 and depth eight, and train with AdamW at batch size 256, using gradient accumulation when needed. Both variants use a ViT-B visual backbone. The end-to-end model trains its visual encoder from scratch; the frozen variant initializes its encoder from V-JEPA 2.1 (Mur-Labadia et al., 2026) and keeps it fixed. For exact reproduction, check the published configuration on the project page.

For Table 2, batch-1 GFLOPs include all inference steps for one full action chunk: 100 DDPM steps for Diffusion Policy (Chi et al., 2023), ten Euler steps for $\pi_{0.5}$ (Physical Intelligence et al., 2025), and ten Euler steps for each physical-robot LeWAM variant.

The physical-robot Diffusion Policy and $\pi_{0.5}$ architectures, training configurations, and execution settings are given in Appendix B.1, separately from the simulated task-completion baseline.

E.3 Closed-Loop Evaluation

For Section 4.4, the robot sends its observation window to a remote inference machine, receives an action chunk, and executes it at 30 Hz under the receding-horizon schedule. A human evaluator labels each episode as success or failure, using the same criteria used during teleoperation. Each method is evaluated in 30 trials per task.

E.4 Representation Diagnostics and Evaluation

A separate probe visualizes what information remains in the visual embeddings and how it evolves under recorded actions. Learnable per-patch queries read the fixed encoder outputs through cross-attention layers and project them to pixel patches. Only this decoder is optimized; the encoder and latent dynamics receive no gradients from it. The architecture is shown in Figure 14.

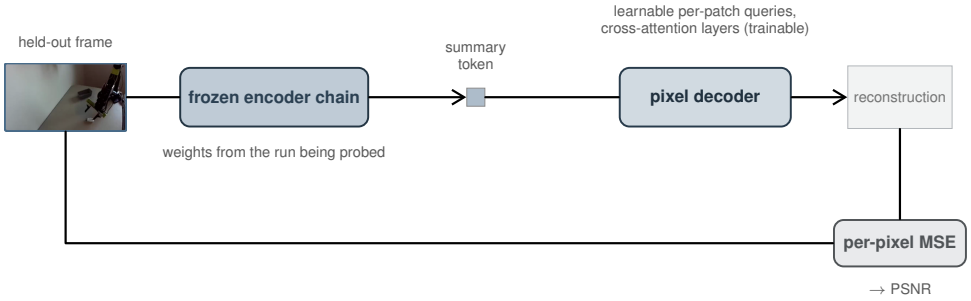


Figure 14: Frozen-encoder pixel-decoder probe. The target frame and reconstruction yield per-pixel MSE and peak signal-to-noise ratio (PSNR).

Figure 15 shows the same held-out Pick and Place T episode from the scene camera for both encoder variants. We display ten frames at indices 0, 3, 6, 9, 12, 15, 18, 21, 24, and 26, spanning 10.4 seconds at the 2.5 Hz model sampling rate. Each strip has three rows: recorded video; independent encode/decode reconstructions of the corresponding recorded frames; and decoded predictions from a latent rollout initialized by the initial observation context, advanced using the recorded action sequence, and never refreshed with intermediate ground-truth frames. The last row tests action-conditioned prediction, not action generation or closed-loop control. In both strips, the predicted orange block moves from the left side of the workspace toward the green target. The end-to-end model retains this coarse sequence without pretrained visual features, and the frozen encoder provides a matched reference. The pixel decoder blurs fine detail and longer rollouts can drift. These strips provide qualitative evidence of action following and visual information retained by the representation, rather than a quantitative measure of physical consistency.

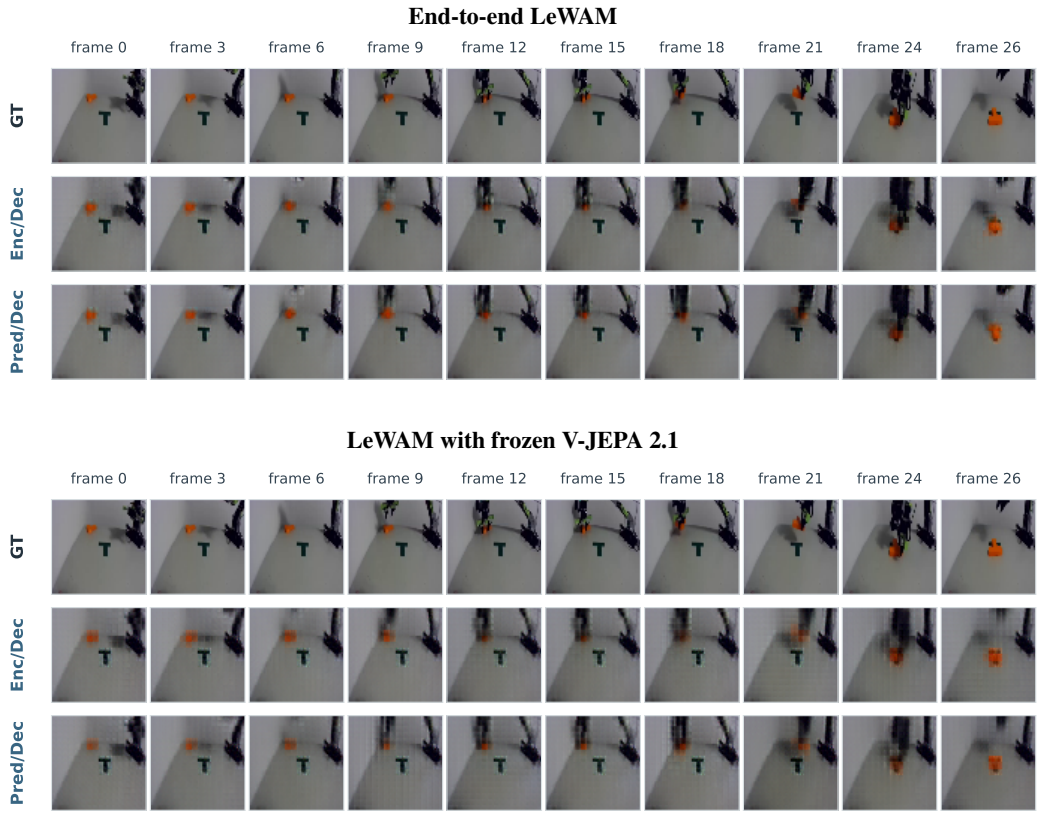


Figure 15: Matched visual reconstructions and recorded-action latent rollouts for the scene camera. Within each model, rows show the ground-truth (GT) video, independent frame reconstructions, and decoded autoregressive predictions.

F JOINT PREDICTION FOR PLANNING AND REPRESENTATION

This section expands the planning and representation mechanisms in Sections 3.2 and 4.

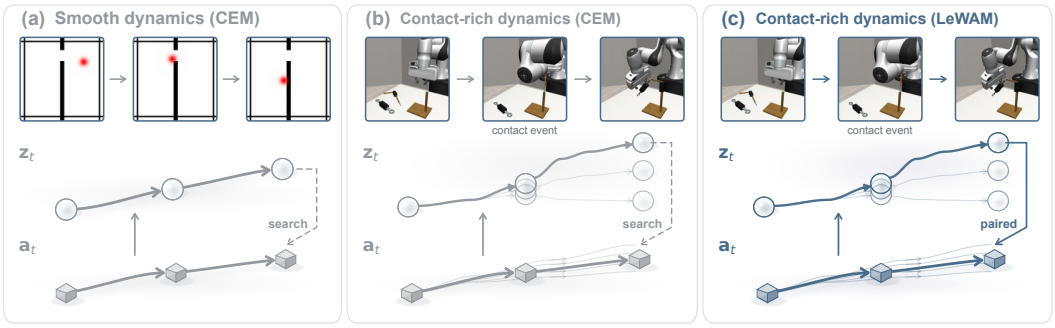


Figure 16: **Action proposals and predicted consequences.** **(a)** Smooth rollouts; **(b)** candidate rollouts branch at contact; **(c)** LeWAM pairs learned action proposals with predicted outcomes. Dashed links denote search; the solid link denotes joint prediction. This schematic illustrates finite-budget search, not a universal CEM failure.

Behavior and consequence. A contact event can switch the active mode, so nearby actions yield distinct outcomes (Pang et al., 2023). If sampled plans miss useful contact, an object-based goal cost provides little ranking signal (Yamada et al., 2024). CEM (Rubinstein, 1999) is derivative-free and can cross nonsmooth transitions, but a finite population cannot favor a useful branch it has not sampled; more iterations increase rollout cost. Learned proposals can improve contact-rich planning under constrained budgets (Brudermüller et al., 2025). LeWAM samples coherent plans from its action distribution, predicts their consequences, and refines plans through frozen dynamics (Figure 16).

Representation shaping. SIGReg (Balestriero & LeCun, 2025; Maes et al., 2026b) regularizes the latent marginal; action supervision encourages control-relevant distinctions (Figure 17). If visual history changes the conditional action velocity, discarding it increases optimal action regression risk. Latent prediction adds pressure to retain action-conditioned consequences. The diagram is conceptual: action prediction alone does not rule out every form of collapse.

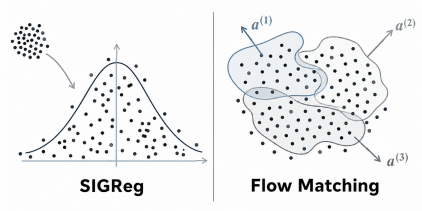


Figure 17: Conceptual roles of SIGReg and action flow matching in shaping the latent space.

Conditional action risk. In Equation 2, squared flow matching makes the optimal velocity the conditional mean of $\boldsymbol{a}_j - \boldsymbol{\epsilon}_j$. Let $\mathcal{I}_j$ include the noisy action prefix, flow time, and other conditioning, but exclude visual history $\boldsymbol{z}_{\leq t}$. With $\mathcal{R}_{\mathrm{FM}}$ denoting optimal velocity MSE, the conditional projection identity gives

$$\begin{aligned}&\mathcal{R}_{\mathrm{FM}}(\text{constant}) - \mathcal{R}_{\mathrm{FM}}(\boldsymbol{z}_{\leq t}) \\ &= \frac{1}{C}\sum_{j=t}^{t+C-1} \mathbb{E}[\mathrm{MSE}(\mathbb{E}[\boldsymbol{a}_j - \boldsymbol{\epsilon}_j \mid \mathcal{I}_j, \boldsymbol{z}_{\leq t}], \mathbb{E}[\boldsymbol{a}_j - \boldsymbol{\epsilon}_j \mid \mathcal{I}_j])] \geq 0.\end{aligned} \tag{12}$$

When visual history changes the conditional velocity, a constant code incurs higher optimal risk. This pressure can vanish if other inputs already determine the velocity. Thus action prediction cannot exclude partial collapse; LeWAM also uses SIGReg to regularize the global latent distribution.