SEGMENT TO FOCUS: GUIDING LATENT ACTION MODELS IN THE PRESENCE OF DISTRACTORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent action models (LAMs) offer a promising path to pre-training embodied agents on large amounts of action-free video. They infer latent actions between consecutive observations that can later be decoded to ground-truth actions using a small number of labels. However, recent work has shown that this recipe fails in the presence of action-correlated visual distractors, such as dynamic backgrounds, camera shake, or other moving objects. In these scenarios, the reconstruction objective drives latent actions to encode exogenous motion instead of agent-controlled dynamics, resulting in policies that underperform when fine-tuned. We observe that endogenous and exogenous factors are typically spatially separated in pixel space: control-relevant change is concentrated on the agent, while distractor motion occurs elsewhere. We exploit this observation by restricting the reconstruction objective to agent pixels, while the model still receives the full observation as input. We call this method **MaskLAM**; it obtains the agent mask zero-shot from off-the-shelf segmentation foundation models (e.g., SAM) and requires no architectural changes, auxiliary losses, or action labels during pre-training. Across two continuous-control benchmarks (Distracting Control Suite, Distracting Meta-World), MaskLAM reduces normalized linear-probe MSE by up to $3.51\times$ and improves normalized return by up to $4.97\times$ over LAPO. On real robot video (DROID), it yields the most action-predictive latents among all compared LAMs.

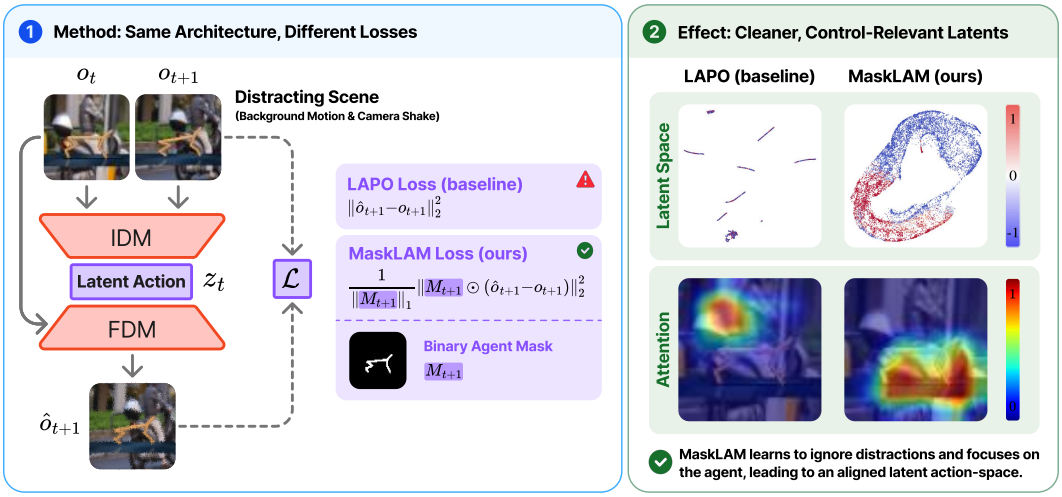


Figure 1: Overview of MaskLAM. Under visual distractors, the reconstruction loss of latent action models such as LAPO [37] pressures the latent action z_t to encode exogenous motion, and downstream policies fail [31]. MaskLAM restricts the loss to agent pixels via a mask M_{t+1} obtained zero-shot from SAM [36] (1), so distractor pixels contribute no gradient to z_t, while the model still receives the full observation. (2) On cheetah-run with distractors, the latent space of LAPO collapses into entangled clusters, while MaskLAM forms a smooth manifold over the ground-truth action (top), and the IDM's Eigen-CAM [30] saliency shifts from the background to the agent (bottom).

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.

1 INTRODUCTION

The scalability of reinforcement and imitation learning [2, 27] is heavily limited by the need for action-labeled data. Although the internet contains an abundance of unlabeled video demonstrations, using this resource is still an unsolved problem when ground-truth control annotations are missing. Latent Action Models (LAMs) [12, 37] have emerged as a promising way to overcome this bottleneck: by jointly training an inverse and forward dynamics model on observation-only video, they infer a *latent action* between successive frames, and a lightweight decoder maps these to actual controls using only a small set of labeled examples. On simple benchmarks (static backgrounds and a single moving agent) this approach works consistently well.

This no longer holds once the video contains action-correlated distractors. Standard LAMs rely on *global* reconstruction objectives that force the model to encode the entire scene to predict future frames. With moving backgrounds, camera shake, or additional moving agents, pixel reconstruction has no reason to prefer agent dynamics over background variation, and the latent encodes whatever varies most [49]. The Exogenous Block MDP (Ex-BMDP) framework [13] formalizes this failure mode: observations entangle a control-endogenous factor (driven by agent actions) with a control-exogenous factor (evolving independently of the agent), and standard reconstruction losses pressure the latent to encode both. As a result, the latent action space of LAPO [37], the basis for many subsequent LAMs, no longer recovers the true action, and downstream policies fail [31]. Prior work addresses this through architectural or modeling interventions: information bottlenecks [47], improved feature extractors [23], auxiliary losses [31, 7], or explicit distractor models [44], at the cost of model complexity, action-label dependence, or extra hyperparameters.

Our key observation is that the Ex-BMDP factorization, although abstract in latent space, is predominantly *spatial* in pixel space: agent actions have immediate, local effect on the *agent's* pixels, while exogenous changes are spatially disjoint from them. The contamination of the latent action therefore originates in the prediction target, and we remove it there. **MaskLAM** leaves the LAPO recipe unchanged and restricts the forward-dynamics loss to agent pixels, which segmentation foundation models [36] localize zero-shot from a single prompt. The inverse and forward dynamics models still receive the full observation, so the mask determines which pixels the latent action must explain, not which pixels the model may use to infer it. The difference matters when the environment moves the agent, for example a log carrying a frog across a river: masking the input removes the context that explains the agent's motion, while masking the loss keeps it available (Section 6.1).

Following prior work on latent actions [31, 23, 7], we study this problem on controlled distractor benchmarks, which allow us to vary the type and strength of distractors systematically. Our goal is to identify where the distractor problem originates in LAM training and to address it at this point. Learning from uncurated real-world video is the broader goal of this line of research, and we view our work as a step towards it.

Our contributions are:

- We identify spatial separability of agent and distractor pixels as the property that turns the Ex-BMDP factorization into a training-time intervention, and propose **MaskLAM**, which restricts the forward-dynamics loss to agent pixels using zero-shot segmentation masks.
- We show that masking the input can only remove action-relevant context (Proposition 1), while masking the loss keeps it. When the environment moves the agent, loss masking improves normalized return over input masking.
- We demonstrate that, on the Distracting Control Suite and Distracting Meta-World, MaskLAM reduces linear-probe error by up to $3.51\times$ and improves normalized return by up to $4.97\times$ over LAPO, and attains lower probe error than LAOM-Labels without action labels, trailing it in return on DCS but exceeding its success rate on DMW.
- We further find that MaskLAM admits compact latent spaces and lower label budgets, degrades gracefully under segmentation errors, and achieves the lowest probe error among all LAMs on real robot video (DROID).

2 Related work

Latent action models. Latent action models (LAMs) aim to recover action-relevant representations from unlabeled state transitions. Early work by ILPO [12] learned discrete latent actions through a forward dynamics objective, but mode collapse and poor scalability [41] limited its reach. LAPO [37] resolved both issues by jointly training an inverse and forward dynamics model, and the resulting framework now underpins large-scale robot manipulation pre-training [9, 46] and interactive world models [5]. It assumes that agent actions explain the state transition; under visual distractors, reconstruction-based LAMs entangle agent dynamics with scene variation [31, 49].

Latent action learning under distractors. Prior work counters distractors with additional supervision or model components. LAOM-Labels [31] decodes ground-truth actions during pre-training, which reintroduces action labels; UniVLA [6] and Nikulin et al. [32] steer latent actions toward task-relevant content with vision-language models. LAOF [7] is the closest prior method: it also uses segmentation masks, but on an auxiliary optical-flow target, while the reconstruction target still contains the distractors. Object-centric LAMs [23, 28] train on probe-selected agent slots, i.e., they mask the input.

Distractors in world models. Model-based RL isolates uncontrollable dynamics in separate latent branches [33, 43, 44] and uses agent or task-relevant masks as supervision [15, 22]. These methods train on action- and reward-labeled transitions. MaskLAM brings target masking to action-free video: unlike LAOF, it masks the reconstruction target itself, so no gradient from distractor pixels reaches the latent action, and unlike object-centric LAMs, it keeps the context outside the mask visible (Proposition 1).

3 Background and Problem Formulation

Exogenous Block MDP. We adopt the Ex-BMDP framework [13] as the formal foundation for reasoning about visual distractors. Each observation $o_t \sim Q(\cdot \mid s_t, e_t)$ entangles an endogenous state s_t, which evolves with the agent's actions, and an exogenous state e_t, which evolves independently of them and represents visual distractors. We further partition the endogenous state as $s_t = (s_t^{\text{agent}}, s_t^{\text{ctx}})$, where s_t^{agent} is the physical state of the agent and s_t^{ctx} the context relevant for inferring or explaining its actions, such as objects it manipulates or surfaces it stands on [45, 24]. For self-actuated agents, the agent transition alone may suffice for inverse dynamics; more generally, the context provides additional information about the action. If external, unobserved influences directly affect the agent transition, the observed agent motion need not uniquely identify the executed action [29]. The goal of latent action learning is therefore to recover a latent action $z_t \in \mathbb{R}^{d_z}$ that captures the endogenous transition while discarding exogenous variation.

Latent action models. Most reinforcement and imitation learning methods require action-labeled trajectories $\tau_n = \{(o_i^n, a_i^n)\}_{i=1}^T$, but the vast majority of video data on the internet provides only observation-only trajectories $\tau_n = \{o_i^n\}_{i=1}^T$. LAMs [12, 37] bridge this gap by inferring latent actions z_i from observation-only data, which produces $\mathcal{D}_z = \{(o_i, z_i)\}$ for downstream policy learning. We build on LAPO [37], the basis for many subsequent LAMs [19, 10, 46]. LAPO jointly trains an *inverse dynamics model* (IDM) $z_t \sim p_{\text{IDM}}(\cdot \mid o_t, o_{t+1})$ and a *forward dynamics model* (FDM) $\hat{o}_{t+1} \sim p_{\text{FDM}}(\cdot \mid o_t, z_t)$ to minimize the next-observation prediction loss:

$$\mathcal{L}_{\text{FDM}}^{\text{LAPO}} = \mathbb{E}\left[\|\hat{o}_{t+1} - o_{t+1}\|_2^2\right]. \tag{1}$$

The latent acts as an information bottleneck: since both models observe o_t but only the IDM sees o_{t+1}, the IDM is forced to compress transition information between consecutive frames into z_t. When endogenous variation dominates the transition, the bottleneck aligns with the true actions [39, 24].

Latent action learning pipeline. The standard pipeline [37] has three stages. Stage 1 trains the IDM and FDM on the full unlabeled dataset. Stage 2 relabels all transitions with the trained IDM, producing $\mathcal{D}_z$, and trains a latent policy $\pi_{\text{latent}}(z_t \mid o_t)$ by behavior cloning. Stage 3 trains a decoder $d\colon \mathcal{Z} \to \mathcal{A}$ on a small labeled set; the deployable policy is $\pi = d \circ \pi_{\text{latent}}\colon \mathcal{O} \to \mathcal{A}$.

Why LAPO fails under distractors. Since $o_{t+1} \sim Q(\cdot \mid s_{t+1}, e_{t+1})$, the FDM in (1) must predict changes from *both* the action-relevant and action-irrelevant visual transitions. Given o_t, z_t is the only varying input to the FDM, so the optimization pressures it to encode action-irrelevant

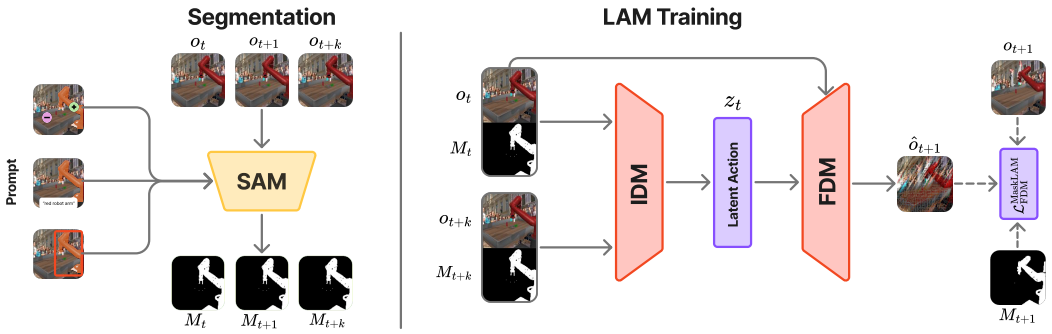


Figure 2: Simplified MaskLAM stage 1 pipeline. Conditioned on a target agent prompt, a pre-trained SAM model extracts agent masks M_t, M_{t+1}, M_{t+k} from observations o_t, o_{t+1}, o_{t+k}. The IDM consumes $(o_t, o_{t+k}, M_t, M_{t+k})$ and predicts the latent action z_t; the FDM then reconstructs $\hat{o}_{t+1}$ from (o_t, z_t, M_t). The forward dynamics loss is spatially restricted to agent pixels (white) via M_{t+1}, so distractor pixels (black) contribute no gradient.

transition information whenever this reduces prediction error on distractor pixels [49]. As recently shown analytically by Lee et al. [25], standard reconstruction can explicitly place future exogenous information into latent actions. Nikulin et al. [31] show that this contamination persists even with a multi-step IDM, larger latents, a latent-space FDM, and augmentations.

Problem statement. We seek a method that suppresses reconstruction pressure from action-irrelevant visual distractors while preserving action-relevant context, without requiring ground-truth action supervision or modifying the LAM architecture.

4 Method

As shown in Section 3, the FDM loss (1) reconstructs the full observation and thereby forces z_t to encode exogenous distractors alongside the endogenous transition. Instead of modifying the encoder or adding auxiliary losses (Section 2), MaskLAM modifies the *prediction target*. Prior methods separate endogenous from exogenous variation in latent space [13, 24, 23, 7]; MaskLAM performs this separation in pixel space, where pre-trained segmentation models localize the agent zero-shot. The separation applies only to the target; the context outside the agent stays visible.

Spatial separability assumption. Let $M_t \in \{0,1\}^{H\times W}$ be a binary segmentation mask identifying agent pixels, obtained from a pre-trained segmentation model. MaskLAM assumes the observation emission is approximately spatially separable with respect to the agent's physical state:

$$M_t \odot o_t \approx M_t \odot Q_{\text{agent}}(\cdot \mid s_t^{\text{agent}}), \quad (1 - M_t) \odot o_t \approx (1 - M_t) \odot Q_{\text{out}}(\cdot \mid s_t^{\text{ctx}}, e_t), \tag{2}$$

where $\odot$ denotes the Hadamard (element-wise) product, and Q_{agent} and Q_{out} denote the agent and contextual/exogenous components of the emission function. This holds under a *visual independence* assumption: the rendering of pixels within M_t must be governed mainly by s_t^{agent}, independent of e_t. Spatial overlap between agent and distractor is allowed. The assumption requires a visible, segmentable agent, as in third-person video; we discuss egocentric video in Section 7.

Masked forward dynamics loss. Given the spatial separability assumption, the most direct intervention is to restrict the FDM loss to agent pixels:

$$\mathcal{L}_{\text{FDM}}^{\text{MaskLAM}} = \frac{1}{\|M_{t+1}\|_1} \|M_{t+1} \odot (\hat{o}_{t+1} - o_{t+1})\|_2^2, \tag{3}$$

where M_{t+1} is the binary mask indicating agent occupancy in the target frame. The loss is zero on distractor pixels, and normalization by $\|M_{t+1}\|_1$ keeps gradient magnitude consistent across varying mask sizes. To see why this eliminates exogenous pressure on z_t, consider the gradient:

$$\frac{\partial \mathcal{L}_{\text{FDM}}^{\text{MaskLAM}}}{\partial z_t} = \frac{2}{\|M_{t+1}\|_1} \left(M_{t+1} \odot (\hat{o}_{t+1} - o_{t+1})\right)^\top \frac{\partial \hat{o}_{t+1}}{\partial z_t}. \tag{4}$$

Where $M_{t+1} = 0$, the residual is zeroed before propagating into z_t. Since encoding $e_t \to e_{t+1}$ cannot reduce $\mathcal{L}_{\text{FDM}}^{\text{MaskLAM}}$, the optimization no longer encourages z_t to encode exogenous dynamics.

Mask-conditioned IDM and FDM. We concatenate the segmentation mask as an extra input channel: the IDM receives $(o_t, o_{t+k}, M_t, M_{t+k})$ and the FDM receives (o_t, z_t, M_t). The mask channel tells the IDM whose transition z_t must explain and the FDM where to apply it, but it does not restrict what either model may see: both receive unmasked observations and can use the full scene as context, including objects outside M_t that the agent interacts with (Figure 5).

Why loss masking, not input masking. Alternatively, one could mask the input and feed $M_t \odot o_t$ to the IDM and FDM, as object-centric LAMs do [23]. Both variants exclude exogenous pixels from the target, but they differ in the information available to the IDM. Let $X_t = (s_t^{\text{agent}}, s_{t+1}^{\text{agent}})$ denote the agent transition inside the mask and $C_t = (s_t^{\text{ctx}}, s_{t+1}^{\text{ctx}})$ the endogenous context outside of it. With input masking, the IDM observes only X_t; with loss masking, it additionally observes C_t.

Proposition 1 (Context can only improve inverse dynamics)**.** *For predicting a_t from X_t or from (X_t, C_t), the Bayes-optimal risks under log loss and squared error satisfy*

$$H(a_t \mid X_t) - H(a_t \mid X_t, C_t) = I(a_t; C_t \mid X_t) \geq 0, \tag{5}$$

$$\mathbb{E}\big[\mathrm{Var}(a_t \mid X_t)\big] - \mathbb{E}\big[\mathrm{Var}(a_t \mid X_t, C_t)\big] = \mathbb{E}\big[\mathrm{Var}\big(\mathbb{E}[a_t \mid X_t, C_t] \mid X_t\big)\big] \geq 0. \tag{6}$$

The first gap is positive if and only if $I(a_t; C_t \mid X_t) > 0$, the second if and only if the context changes the conditional mean of a_t.

The proof follows from the chain rule of mutual information and the law of total variance (Section T.1). Proposition 1 concerns the information available to the IDM and does not guarantee what unsupervised training recovers. It does, however, distinguish two cases. For self-actuated agents, a_t is identifiable from X_t, both gaps vanish, and input and loss masking should perform on par. If the environment moves the agent, C_t indicates which action produced the observed displacement, and input masking discards this information. Loss masking keeps C_t visible and, by (4), still removes the exogenous gradient on z_t. We evaluate both cases in Section 6.1.

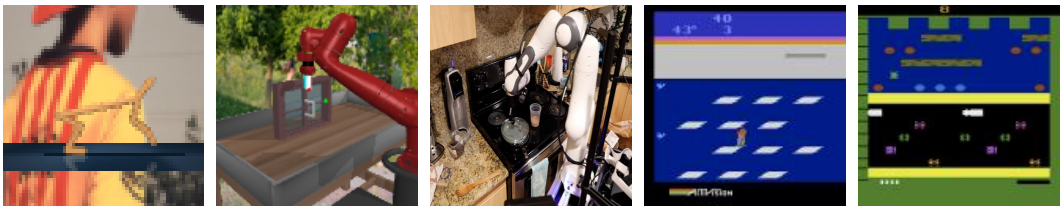
Implementation (Figure 2)**.** Following prior work [31, 24, 26], we use a multi-step IDM $z_t \sim p_{\text{IDM}}(\cdot \mid o_t, o_{t+k})$ with k sampled uniformly from $\{1, \ldots, 10\}$ during training and $k = 1$ at inference, since exogenous noise decorrelates over longer horizons [24, 13]. As in Nikulin et al. [31], we stack 3 consecutive observations as input to both the IDM and FDM. Masks are obtained zero-shot from SAM [36, 8] and pre-computed before LAM training (Section C). We use the continuous LAPO baseline from Nikulin et al. [31] (IMPALA ResNet encoder [14], ResNet decoder [17]).

5 Experiment Setup

We evaluate the masked forward dynamics loss (3) along four axes: latent action quality, downstream policy performance, robustness to segmentation failures, and the role of context outside the mask.

Environments. We evaluate on two commonly-used distractor benchmarks following prior work on latent actions [31, 23], on real robot video, and on two Atari games for a targeted ablation (Figure 3).

- *Distracting Control Suite* (DCS) [40]: augments DeepMind Control Suite [42] with dynamic video backgrounds (DAVIS [34]), randomized agent colors, and camera movement. Our main experiments use the *easy* setting, which applies only background videos and color randomization; the *hard* setting (Section I) uses stronger color randomization, more background videos, and a dynamic camera. Color randomization alters the agent itself, so distractors are not fully spatially separable.
- *Distracting Meta-World* (DMW): augments Meta-World [48] with dynamic natural video backgrounds. We evaluate the MT10 subset. DMW introduces richer agent geometry, testing mask quality under complex interactions.
- *DROID* [21]: 350 hours of cluttered, real robot manipulation across 564 scenes and 86 tasks. Observations are sampled from two external cameras at 1 Hz; actions are relative joint positions and the absolute gripper position, following Black et al. [4].
- *Atari* [3]: Frostbite and Frogger, two games in which the environment influences the agent. We use both games only for the loss- versus input-masking ablation.



(a) DCS: cheetah-run (b) DMW: window-open (c) DROID (d) Atari: Frostbite (e) Atari: Frogger

Figure 3: Environments. DCS (a) adds dynamic background videos and agent color randomization; DMW (b) adds dynamic background videos. DROID (c) contains real robot manipulation in cluttered, uncontrolled scenes. In Frostbite (d) and Frogger (e), drifting floes, logs, and turtles move the agent.

For DCS and DMW we provide both ground-truth masks (from simulation) and SAM 2.1 masks [36]. On DROID, SAM 3.1 [8] segments the robot zero-shot from the single language prompt "robot manipulator". Atari ground-truth masks come from OCAtari [11]. All datasets will be released. Unless stated otherwise, results on DCS and DMW refer to the distractor setting; distractor-free results are reported in Sections G and H.

Baselines. *LAPO* [37] trains the IDM-FDM pair without auxiliary losses; following LAOM [31] we remove the VQ-VAE for better performance on continuous control. *LAOM-Labels* adds an auxiliary linear probe loss decoding ground-truth actions during Stage 1 training, showing what privileged label access can achieve; we use 128k ground-truth labels unless stated otherwise, and 65.5k for DROID. *LAOF* [7] adds a masked optical flow auxiliary loss, the closest prior method that also uses segmentation masks, but on flow instead of reconstruction. *LAPO-Slots* [23] decomposes observations into object slots, selects the agent slot with a small labeled probe, and thereby masks the *input* instead of the loss. We report MaskLAM in two configurations, **MaskLAM@GT** with simulator ground-truth masks and **MaskLAM@SAM** with SAM masks; unsuffixed **MaskLAM** refers to both jointly. All baselines use their published hyperparameters, adapted only for convergence on our datasets, with latent dimension $d_z = 128$ unless stated otherwise (Sections D and E).

Metrics. We report mean $\pm$ standard deviation over 3 seeds (protocol in Section F). Latent action quality is measured by a linear probe that predicts ground-truth actions from frozen latent actions [1, 49]. We report its *normalized mean squared error* (NMSE), i.e., the probe error divided by the action variance, and, for the discrete Atari actions, its *accuracy*. Downstream performance is measured over 100 episodes relative to the expert: the *normalized success rate* (NSR) on DMW [48] and the *normalized return* (NR) on DCS [42] and Atari (Section F).

6 Results and Discussion

We frame our analysis through the lens of Zhang et al. [49], who show that linear latent action models trained by next-frame reconstruction behave like PCA on the change between consecutive frames:

Table 1: Latent action quality (NMSE) and downstream performance (NR/NSR) under distractors aggregated across tasks. MaskLAM attains the lowest probe error on every benchmark and the highest return of all label-free methods. Distractor-free results in Sections G and H.

	Distracting Control Suite		Distracting Meta-World		DROID
Method	**NMSE ↓**	**NR ↑**	**NMSE ↓**	**NSR ↑**	**NMSE ↓**
LAPO	$0.5114_{\pm 0.0336}$	$0.0759_{\pm 0.0092}$	$0.3093_{\pm 0.0094}$	$0.6362_{\pm 0.0147}$	$0.8479_{\pm 0.0066}$
LAOM-Labels	$0.2160_{\pm 0.0034}$	$\mathbf{0.5663_{\pm 0.0186}}$	$0.1421_{\pm 0.0077}$	$0.7810_{\pm 0.0328}$	$0.8822_{\pm 0.0028}$
LAOF	$0.5056_{\pm 0.0194}$	$0.0939_{\pm 0.0195}$	$0.4529_{\pm 0.0569}$	$0.5768_{\pm 0.0791}$	$0.8484_{\pm 0.0053}$
LAPO-Slots	$0.6570_{\pm 0.0293}$	$0.0454_{\pm 0.0078}$	$0.2450_{\pm 0.0481}$	$0.8274_{\pm 0.0229}$	$0.8544_{\pm 0.0081}$
MaskLAM@GT	$\mathbf{0.2061_{\pm 0.0093}}$	$0.3776_{\pm 0.0298}$	$\mathbf{0.0880_{\pm 0.0036}}$	$\mathbf{0.8432_{\pm 0.0211}}$	—
MaskLAM@SAM	$0.2326_{\pm 0.0372}$	$0.3581_{\pm 0.0404}$	$0.0903_{\pm 0.0049}$	$0.8318_{\pm 0.0314}$	$\mathbf{0.7979_{\pm 0.0063}}$

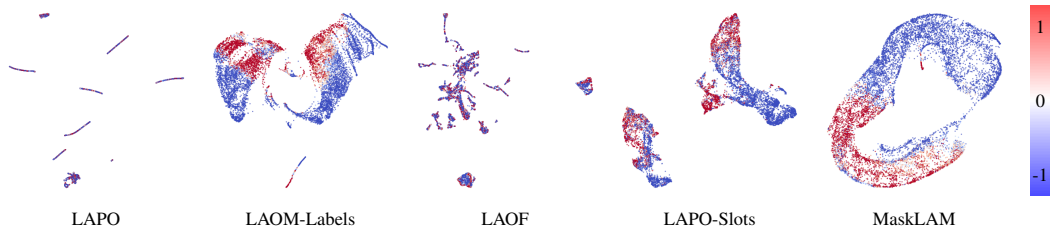


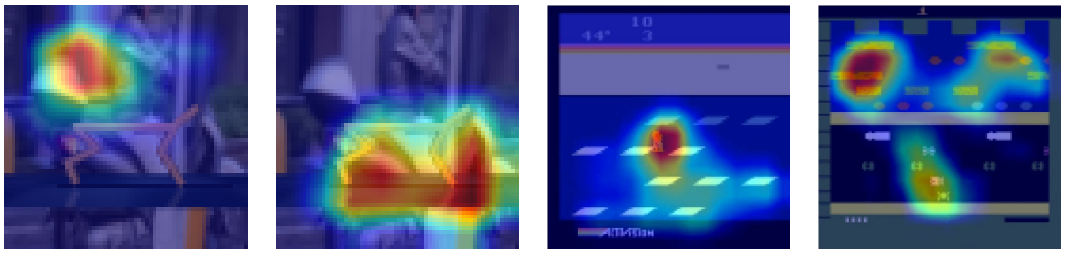
Figure 4: UMAP projection of the latent action space on DCS cheetah-run, colored by the 4th ground-truth action dimension. A well-aligned latent space exhibits a smooth color gradient along the projection. MaskLAM produces a clean, monotonically structured manifold, while LAPO and LAOF collapse into entangled clusters with no clear correspondence to the ground-truth action. Per-dimension projections for all joints are provided in Section K.

the latent encodes whatever varies most, regardless of source. Once distractors enter the frame, this is exogenous variance, which MaskLAM removes from the target.

Masking the target restores action alignment. MaskLAM@GT achieves the lowest probe error of all methods on both simulated benchmarks, including LAOM-Labels (Table 1), and keeps its lead over all label-free baselines in stronger distractor settings (Section I). MaskLAM@GT and MaskLAM@SAM are within noise, so off-the-shelf segmentation suffices. The restored alignment is visible in Figure 4: MaskLAM produces a smooth gradient over the ground-truth action, while LAPO and LAOF collapse into entangled clusters. On real robot video (DROID), MaskLAM@SAM is the only LAM that improves over LAPO, closing 30% of the gap to a supervised IDM (Section J). The restored alignment carries over to control: MaskLAM@GT exceeds every label-free baseline in downstream performance on both benchmarks (Table 1), raising normalized return on DCS from 0.08 (LAPO) to 0.38.

MaskLAM attends to the agent and its endogenous context. Eigen-CAM [30] saliency of the IDM (Figure 5) falls on the distractor background for LAPO and on the agent for MaskLAM. Part of it lies outside M_{t+1}: on cheetah-run it also covers the ground the agent pushes against (b), in Frostbite and Frogger the ice floes and logs that carry the agent (c,d). The IDM thus learns from data to attend to the agent and its context that shapes the agent's transition, which becomes decisive once this context influences the agent (Section 6.1). Time series in Sections L and T.2.

A masked target frees latent capacity. By removing the gradient pressure that ties the latent size to the prediction target, MaskLAM lets the model use a smaller latent without sacrificing alignment. Figure 6a sweeps the latent action dimension with and without the masked loss: with masking, a 64-dim latent already matches the alignment of a 256-dim latent without it; the gap is largest in the low-dimensional regime, where exogenous content would push out action information, a trade-off also reported for LAOM [31] and LAPA [46]. The cleaner latents also need fewer labels: with as few as 2k labels for Stage 3 decoder fine-tuning, MaskLAM reaches near peak performance on DMW,



(a) LAPO, cheetah-run (b) MaskLAM, cheetah-run (c) MaskLAM, Frostbite (d) MaskLAM, Frogger

Figure 5: Eigen-CAM saliency of the IDM. Warmer colors indicate higher influence on the predicted latent action. On DCS, LAPO attends to the background (a) and MaskLAM to the agent and the ground it pushes against (b). On Atari, MaskLAM also attends to the ice floes (c) and to the logs and turtles (d) that carry the agent, although they lie outside the agent mask.

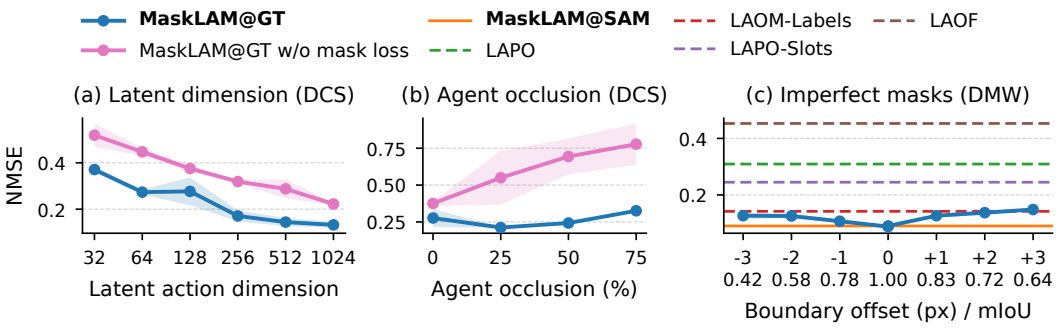


Figure 6: Compact latents and robustness to segmentation errors. (a) On DCS, masking lets a 64-dim latent match a 256-dim unmasked one, with the largest gap for small latents. (b) With masking, probe error stays nearly flat up to 75% agent occlusion, while it more than doubles without. (c) On DMW, shrinking or expanding the ground-truth masks by up to 3 pixels (mIoU down to 0.42) keeps MaskLAM@GT below LAPO, LAPO-Slots, and LAOF (horizontal references).

while LAPO and LAPO-Slots need nearly 100× more; on DCS, they fail at every budget (Figure 19). LAOM-Labels overtakes MaskLAM at larger budgets despite higher probe NMSE, so the gap is not a decoding limit; we hypothesize that action supervision yields latents more robust to rollout-induced distribution shift.

MaskLAM learns a unified latent action space. Since the mask only selects whose transition is explained, a single MaskLAM can in principle cover several embodiments. We test this by training one MaskLAM@GT on all four DCS embodiments, whose action spaces range from 4 (hopper) to 21 (humanoid) dimensions, with the data budget of a single-embodiment run, i.e., a quarter of the usual data per embodiment. Decoded per task, the unified latent space matches the four embodiment-specific models in probe NMSE (0.208 vs. 0.206) and normalized return (0.369 vs. 0.378; Section P).

MaskLAM is robust to segmentation errors. Real segmenters miss occluded parts of the agent and never produce pixel-perfect boundaries. We simulate occlusion on DCS by masking out a growing fraction of the agent during training and evaluation and measuring the resulting probe error (Figure 6b). With the masked loss, probe error stays nearly flat up to 75% occlusion; without it, the same level of occlusion more than doubles it. Occlusion erodes the target's action-to-noise ratio under the unmasked loss, whereas MaskLAM holds it steady by keeping distractor variance out of the target (Section Q). To test boundary errors, we shrink or expand ground-truth masks by 0 to 3 pixels on DMW, down to mIoU 0.42 (Figure 6c; Figure 27). Probe error stays below LAPO, LAPO-Slots, and LAOF across the entire perturbation range. With MaskLAM@GT and MaskLAM@SAM in agreement, pixel-perfect segmentation is not required.

6.1 Ablation Study

To isolate the contribution of each component, we ablate the masked loss (*w/o loss masking*), the additional mask input channel (*w/o mask channel*), both together (*w/o all masking*), and the multi-step IDM (*w/* k=1). We further compare against a variant that zeroes distractor pixels in the input instead of the loss (*w/ input masking*). Removing every component reduces MaskLAM to LAPO. Figure 7a reports the resulting action probe NMSE aggregated across DCS and DMW.

Loss masking is the principal contributor. Removing the masked loss more than doubles the probe error, and removing all components recovers LAPO. Loss masking accounts for the largest share of the gain, consistent with Zhang et al. [49]: it alone removes exogenous variance from the target. The other components refine it. Dropping the mask input channel hurts performance only when the masked loss is present, so the channel acts as a multiplier on loss masking, and setting k=1 degrades performance both with and without masking.

Loss masking beats input masking when the environment influences the agent. As predicted by Proposition 1, on DCS and DMW input masking is statistically indistinguishable from loss masking: the agents are mostly self-actuated, and pixels outside M_t add little information about the

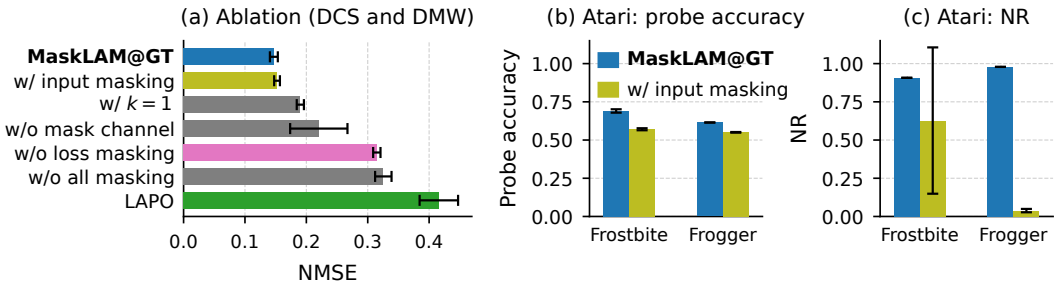


Figure 7: Ablation of MaskLAM components and loss versus input masking. (a) On DCS and DMW, loss masking contributes most; input masking is on par for these self-actuated agents. (b, c) On Frostbite and Frogger, where the environment moves the agent, loss masking improves probe accuracy (b) and NR (c); input masking fails on Frogger and is unstable across seeds on Frostbite.

action. We therefore evaluate on Frostbite and Frogger, where the agent's transition also depends on s_t^{ctx}: its displacement is the sum of its action and the motion of the floe, log, or turtle it stands on. Input masking hides this context from the IDM, so the unexplained drift is absorbed by z_t; loss masking removes it only from the target, and the IDM can account for it. Accordingly, loss masking raises probe accuracy from 0.561 to 0.653, and the gap widens downstream: normalized return rises from 0.333 to 0.943, with input masking collapsing on Frogger (0.038; Figure 7b,c). Adding the manipulated object to the target on contact-rich DMW tasks leaves probe NMSE unchanged (Section T.3), since the object remains visible as input. Agent-only masks thus suffice.

7 Limitations

There are several limitations to our work. First, MaskLAM relies on an upstream segmentation model, and segmentation errors enter the supervisory target directly. We used SAM 2.1 and SAM 3.1, but any model that produces agent masks can be used. While no study can cover every segmentation failure, we evaluated two frequent ones, partial occlusion (Section Q) and imprecise boundaries down to an mIoU of 0.42 (Section R), and off-the-shelf segmenters sufficed even on DROID (Section J). Second, the mask defines which transition the latent action must explain: an action whose only visible effect lies outside the agent provides no training signal unless the mask is extended to cover it (Section T.3). In egocentric videos, ego-motion affects every pixel, the appropriate mask covers the full frame, and MaskLAM reduces to LAPO. Third, MaskLAM inherits the limitations of pixel-level reconstruction, such as sensitivity to high-frequency noise, higher computational cost, and confounders like textures on the agent's body. Finally, following prior work [31, 23], we focused on controlled distractor benchmarks, which allow precise control over distractor type and strength, but whose noise patterns may differ from uncurated real-world video. DROID provides a first real-world test case, in which unfiltered zero-shot masks already improve latent action quality despite clutter, lighting changes, and segmentation errors; extending the evaluation to large-scale web video is the next step.

8 Conclusion

We revisited latent action learning from action-free video in the presence of action-correlated distractors. We showed that LAPO fails in this regime because of its prediction target: once exogenous variance dominates the next-frame reconstruction loss, the latent action absorbs it. We proposed MaskLAM, which restricts the forward-dynamics loss to agent pixels using zero-shot segmentation masks, requiring no auxiliary losses or action labels during pre-training. On established distractor benchmarks, this intervention outperforms all label-free baselines and is competitive with LAOM-Labels, which relies on privileged action supervision, and on real robot video it lowers probe error over LAPO. Because the mask restricts only the target, the IDM can still draw on endogenous context affecting the agent's transition. Our findings suggest that the supervision needed to separate agent from distractor dynamics can shift from action labels, which are scarce, to segmentation masks, which are cheap and available zero-shot, a step toward latent action pre-training on distractor-rich video.

Ethics Statement

This work introduces a training-time modification to latent action models and is evaluated on simulated continuous-control benchmarks, two Atari games, and the public DROID robot manipulation dataset; it involves no human subjects, no newly collected personally identifying data, and no newly released dataset of real-world imagery. The primary ethical consideration is second-order: by replacing privileged action supervision with zero-shot segmentation, MaskLAM lowers the cost of recovering control-aligned representations from passively observed video, which is beneficial for embodied pre-training but also lowers the barrier to building behavioral models of agents who did not consent to being observed. We discuss the positive and negative consequences of this shift, along with the segmenter-level failure modes that condition both, in Section A.

Reproducibility Statement

We have made the following efforts to ensure that the results in this paper can be reproduced. The full architecture of every module, the observation and frame stacking scheme, and the training procedure are specified in Section D, with complete hyperparameter tables for MaskLAM and all baselines in Section E. The evaluation protocol, including the exact definition of the normalized linear action probe MSE and the normalized return and success-rate metrics, is given in Section F. The procedure used to generate the offline datasets for DCS and DMW, and the preprocessing and splits of DROID and Atari, are described in Section B. The prompting schemes used to obtain zero-shot segmentation masks (box-prompted SAM 2.1 on DCS and DMW, language-prompted SAM 3.1 on DROID) are described in Section C; simulator and OCAtari ground-truth masks are described in Sections B.1 and B.3. All reported numbers are means over three random seeds with standard deviations, evaluated on the held-out split with 100 rollout episodes per seed for downstream performance; the aggregation procedure is stated explicitly in Section F. We will release the full training and evaluation code, the DCS, DMW, and Atari datasets with their masks, and the SAM 3.1 masks for DROID (Section D).

AI Use Statement

We have not used generative AI tools for any task with required disclosure. Theoretical model development, mathematical formulations, proof assistance, hypothesis refinement, research methodology design, implementation, data cleaning, qualitative analysis, and result interpretation were carried out by the authors. Synthetic data generation and translation are not applicable to this work. We used generative AI tools for tasks with recommended disclosure: paper editing, LaTeX formatting and layout, and minor edits to our plotting scripts. We have reviewed all AI-assisted work and checked every change against our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016. URL http://arxiv.org/abs/1610.01644. arXiv preprint arXiv:1610.01644.

[2] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890. doi: 10.1016/j.robot.2008.10.024. URL https://www.sciencedirect.com/science/article/pii/S0921889008001772.

[3] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. arXiv preprint arXiv:2410.24164.

[5] 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 *Forty-first International Conference on Machine Learning*, 2024.

[6] 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, 2025. URL https://arxiv.org/abs/2505.06111. arXiv preprint arXiv:2505.06111.

[7] Xizhou Bu, Jiexi Lyu, Fulei Sun, Ruichen Yang, Zhiqiang Ma, and Wei Li. LAOF: Robust latent action learning with optical flow constraints, 2025. URL http://arxiv.org/abs/2511.16407. arXiv preprint arXiv:2511.16407.

[8] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719. arXiv preprint arXiv:2511.16719.

[9] Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19752–19763, 2025.

[10] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=c7rM7F7jQjN.

[11] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. OCAtari: Object-centric atari 2600 reinforcement learning environments, 2024. URL https://openreview.net/forum?id=4PzxLPEGRn.

[12] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1755–1763. PMLR, 2019. URL https://proceedings.mlr.press/v97/edwards19a.html.

[13] Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RQLLzMCefQu.

[14] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

[15] Kevin Gmelin, Shikhar Bahl, Russell Mendonca, and Deepak Pathak. Efficient RL via disentangled environment and agent representations. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023. URL https://proceedings.mlr.press/v202/gmelin23a.html.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

[18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[19] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 24328–24346. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

[20] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

[21] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[22] Kyungmin Kim, JB Lanier, Pierre Baldi, Charless Fowlkes, and Roy Fox. Make the pertinent salient: Task-relevant reconstruction for visual control with distractions, 2024. URL https://arxiv.org/abs/2410.09972. arXiv preprint arXiv:2410.09972.

[23] Albina Klepach, Alexander Nikulin, Ilya Zisman, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, Igor Kiselev, and Vladislav Kurenkov. Object-centric latent action learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(27):22626–22634, 2026. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v40i27.39423. URL https://ojs.aaai.org/index.php/AAAI/article/view/39423.

[24] Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J. Foster, Lekan P. Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=TNocbXm5MZ.

[25] Jung Min Lee, Taehyun Cho, Li Zhao, and Jungwoo Lee. Why latent actions fail, and how to prevent it, 2026. URL https://arxiv.org/abs/2605.20223.

[26] Alexander Levine, Peter Stone, and Amy Zhang. Multistep inverse is not all you need. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=xyrgG4rsqY.

[27] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. ISSN 1533-7928. URL http://jmlr.org/papers/v17/15-522.html.

[28] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

[29] Agostino Martinelli. Nonlinear unknown input observability and unknown input reconstruction: The general analytical solution. *Information Fusion*, 85:23–51, 2022. ISSN 1566-2535. doi: 10.1016/j.inffus.2022.03.004. URL https://www.sciencedirect.com/science/article/pii/S156625352200029X.

[30] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-CAM: Class activation map using principal components. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2020. doi: 10.1109/IJCNN48605.2020.9206626. URL https://ieeexplore.ieee.org/abstract/document/9206626. ISSN: 2161-4407.

[31] 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. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 46427–46447. PMLR, 2025. URL https://proceedings.mlr.press/v267/nikulin25a.html.

[32] Alexander Nikulin, Ilya Zisman, Albina Klepach, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, and Vladislav Kurenkov. Vision-language models unlock task-centric latent actions, 2026. URL http://arxiv.org/abs/2601.22714. arXiv preprint arXiv:2601.22714.

[33] Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-Dream: Isolating and leveraging noncontrollable visual dynamics in world models. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=shlWc9lnzN.

[34] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation, 2017. URL http://arxiv.org/abs/1704.00675. arXiv preprint arXiv:1704.00675.

[35] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[36] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45c1f6a8cbf2da59ebf2c802b4f742cd-Abstract-Conference.html.

[37] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rvUq3cxpDF.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL http://arxiv.org/abs/1707.06347. arXiv preprint arXiv:1707.06347.

[39] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9573–9585. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html.

[40] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels, 2021. URL https://arxiv.org/abs/2101.02722v1. arXiv preprint arXiv:2101.02722.

[41] Oliver Struckmeier and Ville Kyrki. Preventing mode collapse when imitating latent policies from observations, 2023. URL https://openreview.net/forum?id=Mf9fQ0OgMzo.

[42] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: 10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

[43] Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022. URL https://proceedings.mlr.press/v162/wang22c.html.

[44] Yucen Wang, Shenghua Wan, Le Gan, Shuai Feng, and De-Chuan Zhan. AD3: Implicit action is the key for world models to distinguish the diverse visual distractors. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 51546–51568. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24bq.html.

[45] Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23151–23180. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wang22ae.html.

[46] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June 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. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45d74e190008c7bff2845ffc8e3facd3-Abstract-Conference.html.

[47] Weirui Ye, Yunsheng Zhang, Pieter Abbeel, and Yang Gao. Become a proficient player with limited data through watching pure videos. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Sy-o2N0hF4f.

[48] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

[49] Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn?, 2025. URL http://arxiv.org/abs/2506.15691. arXiv preprint arXiv:2506.15691.

Overview. Section A discusses broader impacts. Sections B to F describe the datasets, mask generation, implementation, hyperparameters, and evaluation protocol. Sections G to M extend the results on latent action quality and downstream performance, including real robot video and qualitative analyses. Sections N to R cover latent dimension, label budget, multiple embodiments, occlusion, and mask quality. Sections S and T extend the ablation, including the proof of Proposition 1 and the Atari experiments, and Section U analyzes the failure modes of LAOF and LAPO-Slots.

A Ethical Considerations and Broader Impacts

MaskLAM lowers the cost of extracting control-relevant signal from action-free video by replacing privileged action supervision with zero-shot segmentation. We summarize the most likely positive and negative consequences of this shift.

Positive impact. The principal benefit is a reduction in the action-label budget required to pre-train embodied agents on observation-only video. Action labels are scarce and expensive to collect; segmentation masks are cheap and already available zero-shot from foundation models such as SAM 2.1 [36]. By re-allocating the supervision burden from action labels to masks, MaskLAM is a step towards latent action pre-training at the data scales that have driven progress in vision and language, with potential applications in robot learning, assistive robotics, and any embodied domain where teleoperated demonstrations are difficult to obtain. The same shift broadens the pool of researchers who can experiment with latent action models without privileged hardware or large labeled datasets.

Negative impact. The same property is generic to any video source: a method that recovers control-aligned representations from passively observed video lowers the barrier to building behavioral models of agents that did not consent to that observation. The downstream risks are surveillance-adjacent (an actor with access to public video of a target task can pre-train a policy that imitates the demonstrator without ever instrumenting them, and promptable segmenters such as SAM 2.1, or SAM 3.1 with plain-language prompts, make it straightforward to single out specific individuals or assets within a scene) and automation-displacement-adjacent (any reduction in the cost of imitation from video lowers the marginal cost of replacing the imitated work). We see no technical mitigation at the level of the LAM itself; mitigation belongs at the data and deployment layer, through dataset auditing, restrictions on identifying segmentation prompts, and venue-level norms around consent for video used in embodied pre-training.

B Datasets

We use three kinds of data. On DCS and DMW we collect our own synchronized distractor-free and distracting datasets from expert policies (Section B.1). DROID provides real robot video with natural distractors, for which we generate masks without re-collecting data (Section B.2). On Atari we collect trajectories in two games in which the environment moves the agent (Section B.3).

B.1 Distracting Control Suite and Distracting Meta-World

To isolate the effect of visual distractors, we generate *synchronized* pairs of distractor-free and distracting trajectories: the underlying agent behavior and environment state are identical in both settings, so any gap in latent action quality or downstream return comes from how a method handles the visual distractors and not from a different action distribution.

Expert policies. On DCS we reuse the released expert checkpoints of LAOM [31] instead of retraining our own. Briefly, those checkpoints are PPO [38] from CleanRL [20] for cheetah-run, walker-run, and hopper-hop, and SAC [16] from Stable-Baselines3 [35] for humanoid-walk, all trained on proprioceptive states with default hyperparameters; we refer the reader to [31] for the full training schedule. On DMW we use the released single-task SAC experts of Meta-World [48], one per task, trained with the Garage configuration of 500 epochs $\times$ 40 epoch cycles $\times$ 500 gradient steps per cycle, corresponding to roughly 10M environment steps per task; full hyperparameters are reported in [48]. Across both benchmarks the experts act on proprioceptive states, not on rendered images, so visual distractors cannot influence their decisions and the resulting action sequences are deterministic functions of the seed and initial state.

Trajectory collection. We roll out each expert checkpoint and render every state twice: once with a clean background and once with the distractor configuration of Section 5, sharing the camera pose, agent state, and time step across both renders. Because the expert ignores pixels, the two renders differ only in the observation channel and not in the recorded action or the next state, which gives the synchronized distractor-free / distracting pair described above. The collected datasets contain 9M training and 1M held-out transitions per task on DCS, and 1M training and 100k held-out transitions per task on DMW. For each transition we store the rendered observation, the ground-truth simulator mask, the proprioceptive state, and the executed action.

Distractor videos. The dynamic backgrounds are sampled from DAVIS [34]. We assign DAVIS clips to our train and test splits disjointly: training trajectories sample backgrounds only from the DAVIS train split, and held-out evaluation trajectories sample backgrounds only from the DAVIS test split. This is intentionally stricter than the standard protocol in LAOM, which reuses the DAVIS clips seen during training in its in-distribution evaluation and reports a separate, harder "OOD" number on held-out clips. By construction, every evaluation in this paper corresponds to that OOD setting, so our reported metrics measure generalization to unseen distractor videos and not memorization of the training backgrounds.

Masks. Ground-truth segmentation masks are read directly from the simulator at render time, so they are tightly aligned with the rendered observation and require no additional processing. SAM 2.1 masks [36] are produced in a separate post-processing pass over the released observations using the prompt-based extraction protocol detailed in Section C. All datasets, including ground-truth and SAM masks, will be released.

B.2 DROID

Source. We use the public DROID RLDS release [21], 350 hours of teleoperated Franka Panda manipulation across 564 scenes and 86 tasks in homes, offices, and labs. Unlike DCS and DMW, the distractors are not synthetic: clutter, lighting changes, camera placement, and people moving through the frame occur naturally, and no distractor-free counterpart exists.

Episode selection and split. We use all episodes of the release, including unsuccessful ones, and hold out randomly selected episodes for evaluation. The resulting dataset contains 2.62M training and 140k held-out transitions.

Observations. We use the two external cameras and discard the wrist camera, whose view contains almost no agent pixels. Frames are kept at the recorded resolution of 184×320 RGB and subsampled from the recording rate of 15 Hz to 1 Hz. For every training sample, one of the two external cameras is drawn at random.

Actions. Following Black et al. [4], the action is the change in the 7 joint positions between consecutive subsampled frames, concatenated with the absolute gripper position, giving $d_a = 8$.

Masks. No ground-truth masks exist; all masks come from SAM 3.1 (Section C), so only MaskLAM@SAM is reported.

B.3 Atari

Games. We use Frostbite and Frogger from the Arcade Learning Environment [3] without frame skipping or sticky actions. Both games contain moving entities outside the agent that partly determine its motion: drifting ice floes in Frostbite, and logs and turtles in Frogger. Frogger additionally contains cars, which move independently of the agent; when the frog shares a lane with cars, their sprites flicker in sync (Section T.2). Frostbite uses the full action set ($|\mathcal{A}| = 18$), Frogger the minimal action set ($|\mathcal{A}| = 5$).

Expert policies and collection. Trajectories are collected with Rainbow [18] agents trained with CleanRL [20] for 40M environment steps, whose returns are listed in Table 9. The datasets contain 2M training and 200k held-out transitions per game.

Observations and masks. Observations are 128×128 RGB frames. Ground-truth agent masks are extracted with OCAtari [11], which reads object positions from the emulator RAM.

C MASK GENERATION

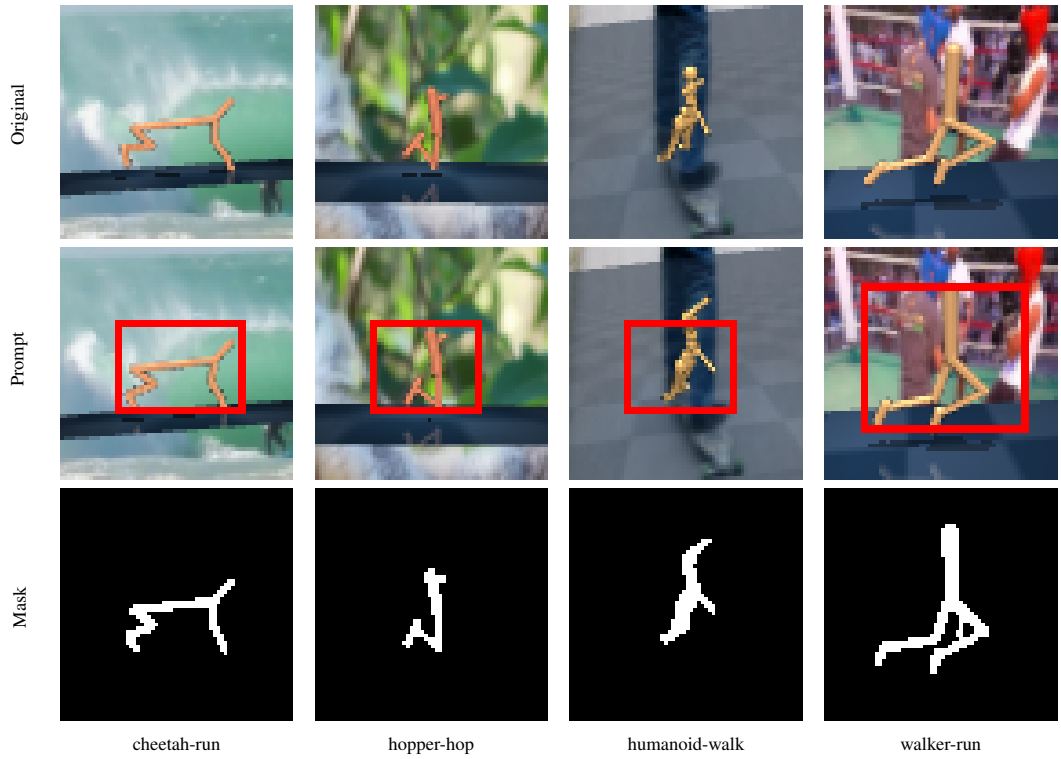


Figure 8: SAM 2.1 mask generation on DCS. Each column shows one task (cheetah-run, hopper-hop, humanoid-walk, walker-run); rows show, top to bottom, the rendered observation, the same observation with the fixed per-task bounding-box prompt overlaid, and the segmentation mask propagated by the SAM 2.1 video tracker from that single first-frame prompt. A single coarse box per task is sufficient to recover a tight agent mask across all four environments despite dynamic backgrounds, agent color randomization, and camera shake.

We pre-compute SAM segmentation masks for every trajectory of both simulated benchmarks using the SAM 2.1-hiera-tiny video model [36]. To avoid per-frame annotation, we exploit the deterministic initialization of each environment: the agent always starts at a known location in the camera frame, so a single coarse, fixed-size bounding box on the first frame of a trajectory is sufficient to identify it. We feed this bounding box as the initial prompt to SAM 2.1 and let the video tracker propagate the mask through the remaining frames. The same procedure is applied to DCS and DMW; per-task bounding boxes are reused across all trajectories of that task. Figure 8 illustrates the prompt and the resulting segmentation for the four DCS environments.

At the reported throughput of 91.5 FPS for the 38.9M-parameter tiny variant on an NVIDIA A100 [36], annotating the full corpus takes approximately 309 GPU-hours on a single A100. SAM masks are stored alongside the simulator ground-truth masks in the released datasets so that any method can be evaluated under either mask source without re-running segmentation.

DROID. Per-episode bounding boxes do not scale to DROID's 564 scenes and varying viewpoints, so we prompt SAM 3.1 [8] with the single concept "robot manipulator" on every frame of both external cameras. MaskLAM consumes masks and is agnostic to how they are produced: switching from box-prompted SAM 2.1 to language-prompted SAM 3.1 requires no change to the objective or the architecture. We do not post-process the masks, so the DROID results include the effect of real segmentation errors. Mask generation for DROID takes approximately 2,800 GPU-hours. Figure 9 shows typical results and failure cases.

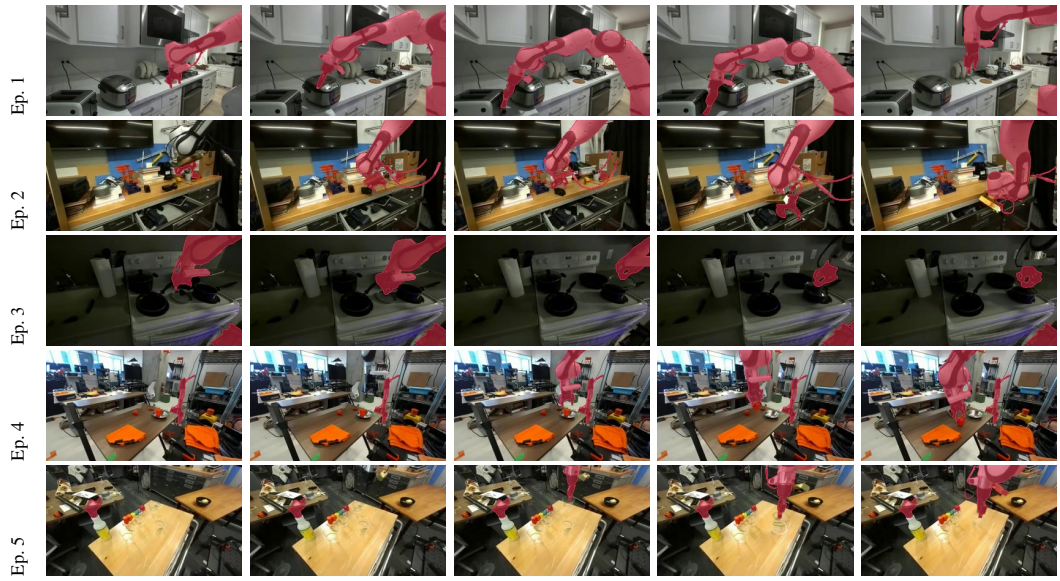


Figure 9: SAM 3.1 masks on DROID from the single prompt "robot manipulator". Each row shows frames of one episode from an external camera, sampled at 1 Hz, with the mask overlaid in pink. Episode 1 shows a typical result: the arm is segmented despite clutter and lighting. The remaining rows show failure modes. In episode 2 the white upper arm is missed while a nearby object is segmented (first frame), and the mask extends along the robot's cables. In episode 3, recorded in a dark kitchen, a false positive persists in the bottom-right corner and the arm shrinks to a fragment around the gripper (fourth and fifth frame). In episode 4 a background rack is segmented as the robot in every frame, and the arm itself is missed in the second frame. In episode 5 the mask leaks onto the head of a spray bottle. MaskLAM trains on these masks without any filtering.

D Implementation Details

Observation and stack. Observations are $64\times64\times3$ RGB on DCS, $128\times128\times3$ on DMW, $184\times320\times3$ on DROID, and $128\times128\times3$ on Atari. Following Nikulin et al. [31] we stack 3 consecutive frames as input to every module, and the IDM additionally consumes 3 future frames at offset $k \in \{1, \ldots, 10\}$ sampled uniformly per training step (Section 4).

Preprocessing. Pixel observations are centered to $[-\frac{1}{2}, \frac{1}{2}]$ via $o \leftarrow o/255 - \frac{1}{2}$, matching the FDM output range. Segmentation masks are binarized to $\{0, 1\}$ at a 0.5 threshold so that ground-truth and SAM masks share the same numerical scale. On DCS and DMW, ground-truth actions are clamped to the environment action space $[-1, 1]^{d_a}$ before being used as supervision, since the simulator already saturates commands at execution time (Section F).

Encoder backbone. Every convolutional module uses the IMPALA backbone of Espeholt et al. [14]: three encoder blocks with base widths $[16, 32, 32]$ scaled by a channel multiplier m, and 2 residual blocks per encoder block. Each encoder block applies a 3×3 convolution, a 3×3 max-pool of stride 2, and the residual stack, halving spatial resolution; a final flatten and linear projection produces a 128-dimensional embedding. Weights are initialized orthogonally.

IDM. Inputs are the 3 current and 3 future frames concatenated along the channel dimension, with each frame augmented by its segmentation mask as an extra input channel (24 input channels in total at $H\times W$). The IMPALA backbone with multiplier $m{=}4$ maps this stack to the continuous latent action $z_t \in \mathbb{R}^{d_z}$ that conditions the FDM. The latent action dimension is $d_z = 128$, if not stated otherwise. No quantizer is used between the encoder and the FDM.

FDM. Implemented as the LAOM-Labels world model of Nikulin et al. [31]: an IMPALA-style encoder with widths $[16, 32, 32]$, multiplier 4, and 2 residual blocks per block, applied to the 3 current frames stacked with their masks (12 input channels). The encoder bottleneck is concatenated channel-wise with z_t projected through a linear layer to the bottleneck spatial shape; a symmetric IMPALA

decoder of transposed-convolution upsampling blocks followed by residual blocks reconstructs back to input resolution, and a final 1×1 convolution maps the result to the 3 RGB channels of $\hat{o}_{t+1}$, squashed to $[-\frac{1}{2}, \frac{1}{2}]$ by $\tanh/2$.

Stage 2 latent policy. A fresh IMPALA backbone with multiplier $m{=}4$ takes the 3 current frames only as input (9 channels; no mask, no future frames) and is trained to regress the frozen IDM output via mean-squared error, producing a d_z-dimensional latent action.

Stage 3 BC actor. The Stage 2 latent-policy backbone is loaded and frozen, and a two-layer MLP of hidden size 256 with ReLU activations followed by a linear action head is appended on top of its d_z-dimensional embedding. Only the MLP and the action head are trained, with mean-squared error against the ground-truth action. The actor is therefore fully deterministic at inference.

Baselines. For LAPO, LAOM-Labels, LAOF, and LAPO-Slots we adopt each method's published latent-action backbone unchanged. To keep the comparison fair we set the latent action dimension to $d_z = 128$ across all methods and use the shared Stage 2 latent policy and Stage 3 BC actor described above for behavior cloning and action decoding. Every reported policy is therefore fully deterministic at inference, and any gap in downstream return reflects the quality of the upstream latent action and not differences in policy architecture.

Training pipelines. LAPO [37], MaskLAM, LAOM-Labels [31], and LAOF [7] all follow the standard three-stage latent-action-learning pipeline of Section 3: Stage 1 trains the LAM on observation-only data, Stage 2 distills a latent policy via behavior cloning against the frozen IDM output, and Stage 3 fine-tunes a small action decoder on a labeled subset. The four methods differ only in their Stage 1 design choices, for which we refer the reader to the original papers; per-stage hyperparameters are listed in Section E.

LAPO-Slots pipeline. LAPO-Slots [23] expands the standard pipeline to five stages by inserting an object-centric pre-training step before LAM training:

- *Stage 1* (slot pre-training): a VideoSAUR encoder is pre-trained on the unlabeled video corpus to convert each frame into a set of object-centric slot representations that separate controllable objects from background distractors.
- *Stage 2* (slot probing): on a small labeled subset, a linear probe regresses ground-truth actions from each slot, and the most action-predictive slot per task is retained.
- *Stage 3* (LAM pre-training): a LAM is trained on the selected slots, inferring latent actions z_t from transitions and learning a forward dynamics model in slot space.
- *Stage 4* (relabeling and behavior cloning): the inferred latent actions serve as pseudo-labels for a behavior-cloning policy that predicts z_t from observations.
- *Stage 5* (action decoder fine-tuning): a small labeled set fine-tunes a decoder that maps the latent policy's output to executable actions.

Hardware. All experiments run on a single NVIDIA A100 40 GB GPU. On DCS, DMW, and Atari, peak memory stays below 24 GB across every method and stage, so these experiments reproduce on consumer GPUs; the higher-resolution DROID runs require more memory. We use bfloat16 mixed precision and `torch.compile` where applicable.

Wall-clock per training run. Table 2 reports the total wall-clock for a single end-to-end training run, summed across all stages of the pipeline, for one seed on one task of each benchmark.

Total compute budget. Table 3 reports the total GPU-hours used in this work, aggregated over every seed, task, ablation, and sweep that contributed to the reported numbers. The MaskLAM row is the largest single contribution because of the additional latent-dimension sweep (Section N), the occlusion sweep (Section Q), the mask-mIoU sweep (Section R), and the ablation study (Section S). Mask pre-computation with SAM 2.1 and SAM 3.1 is accounted for separately as a one-time cost shared by all MaskLAM@SAM runs (Section C).

Code and data release. We will release the full training and evaluation code, the synchronized distractor-free and distracting datasets of both simulated benchmarks with ground-truth and SAM masks, the Atari datasets, and the SAM 3.1 masks for DROID.

Table 2: Wall-clock per end-to-end training run on a single A100 40 GB GPU, summed across all stages of the pipeline for one seed on one task. DROID runs only Stage 1 of the pipeline. LAPO-Slots and LAOM-Labels are the most expensive, LAPO-Slots because of the additional VideoSAUR pre-training stage; MaskLAM costs about as much as LAOF and at most half as much as LAOM-Labels.

Method	DCS	DMW, Atari	DROID (Stage 1 only)
LAPO	≈ 6 h 21 m	≈ 5 h 20 m	≈ 10 h 23 m
LAOM-Labels	≈ 23 h 6 m	≈ 22 h 0 m	≈ 44 h 11 m
LAOF	≈ 11 h 10 m	≈ 7 h 7 m	≈ 14 h 10 m
LAPO-Slots	≈ 37 h 31 m	≈ 18 h 41 m	≈ 37 h 1 m
MaskLAM	≈ 11 h 8 m	≈ 7 h 8 m	≈ 14 h 18 m

Table 3: Total GPU-hours used in this work on a single A100 40 GB GPU, aggregated across every seed, task, ablation, and sweep that contributed to the reported numbers. Mask generation with SAM 2.1 and SAM 3.1 is reported as a separate one-time pre-computation cost.

Method	Runs	GPU-hours
LAPO	986	1,268
LAOM-Labels	615	4,481
LAOF	772	1,295
LAPO-Slots	699	3,516
MaskLAM	2,695	7,339
SAM 2.1 mask generation	—	309
SAM 3.1 mask generation (DROID)	—	2,800
Total	5,767	21,008

E HYPERPARAMETERS

Tables 4 to 8 list per-stage hyperparameters for all five methods. We initialized each baseline from its published hyperparameters and adjusted learning rates, batch sizes, and training budgets where necessary to ensure convergence on our datasets. Pipeline descriptions are in Section D. Where two values appear separated by a slash, the first applies to DCS and the second to DMW/Atari/DROID; otherwise the value is shared. A dash (—) indicates the hyperparameter does not apply to that stage. The action probe listed for MaskLAM and LAPO is a linear probe on detached latent actions, used for monitoring only: none of its gradient reaches the LAM or the latent policy, so neither method uses action labels before Stage 3.

Table 4: Per-stage hyperparameters for MaskLAM (see Section D for the pipeline description).

	Hyperparameter	Stage 1: LAM	Stage 2: BC	Stage 3: Decoder
Optimization	Optimizer	AdamW	AdamW	AdamW
	Learning rate	6×10^{-4} / 3×10^{-4}	4×10^{-4} / 2×10^{-4}	1×10^{-3}
	LR schedule	Cosine annealing	Cosine annealing	Cosine annealing
	Batch size	512 / 128	512 / 128	512 / 128
	Gradient clipping	—	—	1.0
	Mixed precision	bf16	bf16	bf16
Training budget	Epochs / steps	2 / 4 epochs	2 / 4 epochs	25,000 steps
	Block size	13	13	3
	Dataset subset	full	full	128,000
Architecture	Visual encoder	IMPALA	IMPALA	IMPALA
	Channel multiplier	4	4	4
	Encoder channels	6	4	4
	Encoder res blocks	2	2	—
	Latent action dimension d_z	128	128	128
	Frame stack	3	3	3
	Hidden dim (actor)	—	—	256
MaskLAM-specific	Future obs. sampling	✓	×	—
	Future obs. offset	10	10	—
Loss weights	Obs. reconstruction MSE	1.0	—	—
	Action probe MSE (detached)	0.01	0.01	—
	Lat. behavior cloning MSE	—	1.0	—
	Behavior cloning MSE	—	—	1.0

Table 5: Per-stage hyperparameters for LAPO [37] (see Section D for the pipeline description).

	Hyperparameter	Stage 1: LAM	Stage 2: BC	Stage 3: Decoder
Optimization	Optimizer	AdamW	AdamW	AdamW
	Learning rate	6×10^{-4} / 3×10^{-4}	4×10^{-4} / 2×10^{-4}	1×10^{-3}
	LR schedule	Cosine annealing	Cosine annealing	Cosine annealing
	Batch size	512 / 128	512 / 128	512 / 128
	Gradient clipping	—	—	1.0
	Mixed precision	bf16	bf16	bf16
Training budget	Epochs / steps	2 / 4 epochs	2 / 4 epochs	25,000 steps
	Block size	4	4	3
	Dataset subset	full	full	128,000
Architecture	Visual encoder	IMPALA	IMPALA	IMPALA
	Channel multiplier	4	4	4
	Encoder channels	6	4	4
	Encoder res blocks	2	2	—
	Latent action dimension d_z	128	128	128
	Hidden dim (actor)	—	—	256
LAPO-specific	Number of latents	4	—	—
	Policy block size	—	3	—
	Future obs. offset	1	—	—
Loss weights	Obs. reconstruction MSE	1.0	—	—
	Action probe MSE (detached)	0.01	0.01	—
	Lat. behavior cloning MSE	—	1.0	—
	Behavior cloning MSE	—	—	1.0

Table 6: Per-stage hyperparameters for LAOM-Labels [31] (see Section D for the pipeline description).

	Hyperparameter	Stage 1: LAM	Stage 2: BC	Stage 3: Decoder
Optimization	Optimizer	AdamW	AdamW	AdamW
	Learning rate	6×10^{-4} / 3×10^{-4}	3×10^{-4} / 2×10^{-4}	1×10^{-3}
	LR schedule	Piecewise linear	Piecewise linear	Piecewise linear
	Batch size	512 / 128	512 / 128	512 / 128
	Warmup (epochs)	1	0	0
	Weight decay	0	0	0
Training budget	Epochs / steps	3 / 5 epochs	2 / 5 epochs	20,000 steps
	Frame stack	3	3	—
	Image augmentation	✓	—	—
Architecture	Visual encoder	IMPALA	IMPALA	IMPALA
	Encoder scale	6	32	—
	Encoder res blocks	2	2	—
	Hidden dim (MLP)	—	—	256
	Latent action dimension d_z	128	—	—
	Obs. head dim	1024	—	—
LAOM-Labels-specific	Labeled subset size	128,000	—	128,000
	Labeled batch size	512 / 128	—	—
	Labeled loss coef.	1×10^{-3}	—	—
	Future obs. sampling	✓	×	—
	Future obs. offset	10	—	—
	EMA target τ	1×10^{-3}	—	—

Table 7: Per-stage hyperparameters for LAOF [7] (see Section D for the pipeline description).

	Hyperparameter	Stage 1: LAM	Stage 2: BC	Stage 3: Decoder
Optimization	Optimizer	AdamW	AdamW	AdamW
	Learning rate	3×10^{-4} / 1.5×10^{-4}	2×10^{-4} / 1×10^{-4}	1×10^{-3}
	LR schedule	Piecewise linear	Piecewise linear	Piecewise linear
	Batch size	512 / 128	512 / 128	512 / 128
	Gradient clipping	2.0	—	2.0
	Mixed precision	bf16	bf16	bf16
Training budget	Epochs / steps	2 / 4 epochs	2 / 4 epochs	25,000 steps
	Dataset subset	128,000	128,000	128,000
	Frame stack	—	3	3
Architecture	Encoder backbone	IMPALA	IMPALA	IMPALA
	Optical-flow encoder	RAFT	—	—
	World-model scale	24	—	—
	IDM encoder scale	4	—	—
	Policy encoder scale	—	4	4
	Latent action dimension d_z	128	128	128
	Hidden dim (actor)	—	—	256
LAOF-specific	Future obs. offset	1	—	—

Table 8: Per-stage hyperparameters for LAPO-Slots [23] (see Section D for the pipeline description). Stage 2 (slot probing) has no learnable parameters.

	Hyperparameter	Stage 1: VideoSAUR	Stage 2: Probing	Stage 3: LAM
Optimization	Optimizer	AdamW	—	AdamW
	Learning rate	3×10^{-4}	—	3×10^{-4} / 1.5×10^{-4}
	LR schedule	exp. + warmup	—	—
	Warmup	1,250 steps	—	0 epochs
	Batch size	256 / 128	—	256 / 128
	Gradient clipping	0.05	—	1.0
Training budget	Epochs / steps	25,000 steps	—	1 / 4 epochs
	Dataset subset	full	—	full
Architecture	DINO backbone	ViT-B/14 (DINOv2)	—	ViT-B/14 (DINOv2)
	Slot dim	128	—	128
	Num. slots	4 / 3	—	4 / 3
	Slot iterations	3	—	—
	Sim. loss / temp.	0.1 / 0.075	—	—
	Hidden dim	—	—	1024
	Residual blocks	—	—	3
	Latent action dimension d_z	—	—	128
	Frame stack	—	—	1
LAPO-Slots-specific	PCA components	—	32	—
	Probing model	—	Linear regression	—
	Cross-validation	—	5-fold	—
	Future obs. offset	—	—	10

	Hyperparameter	Stage 4: BC	Stage 5: Decoder
Optimization	Optimizer	AdamW	AdamW
	Learning rate	3×10^{-4} / 1.5×10^{-4}	1×10^{-3}
	LR schedule	—	—
	Warmup	0 epochs	0 epochs
	Batch size	256 / 128	256 / 128
	Gradient clipping	1.0	—
Training budget	Epochs / steps	1 / 4 epochs	25,000 steps
	Dataset subset	128,000	128,000
Architecture	Visual encoder	IMPALA	IMPALA
	Channel multiplier	4	—
	Residual blocks	2	—
	Latent action dimension d_z	128	128
	Hidden dim (actor)	256	256

F Evaluation Details

All numbers reported in the paper are computed on the held-out evaluation split of the released datasets (Section B), averaged over three random seeds, with 100 rollout episodes per seed for downstream performance. The remainder of this section makes that protocol explicit, gives the formal definitions of the metrics summarized in Section 5, and lists the per-task action variances that normalize the probe MSE; hardware and wall-clock times are reported in Section D.

Evaluation data. Every metric is computed on the held-out split of the released dataset (Section B): 1M transitions per task on DCS, 100k per task on DMW, 140k on DROID, and 200k per game on Atari. On DROID, the held-out split consists of randomly selected episodes. On DCS and DMW, held-out trajectories sample distractor backgrounds exclusively from the DAVIS test split, while training trajectories sample exclusively from the DAVIS train split, so the two distractor-video sets are disjoint at the clip level. LAOM [31] reports two numbers per task, an in-distribution number that reuses training DAVIS clips at evaluation time and a separate, harder "OOD" number on held-out clips; every number we report corresponds to that OOD setting. We treat this as the only fair protocol, since an in-distribution evaluation conflates memorization of the background videos with generalization to distractor variation, which is the property the method is supposed to provide.

Action clipping. On DCS and DMW, we clip ground-truth actions to the environment action space $[-1, 1]^{d_a}$ prior to probe regression and behavior-cloning fine-tuning. The simulator already clamps commands at execution time, so any two distinct unclipped values that share a clipped image produce identical state transitions; regressing against unclipped targets therefore introduces a many-to-one ambiguity that penalizes models that correctly recover the executed action. Our action-probe MSE values are consequently not directly comparable to LAOM [31], which evaluates on unclipped targets.

Normalized linear action probe MSE (NMSE). Let d_a denote the ground-truth action dimension and d_z the latent action dimension. For each task τ we fit a linear probe consisting of a weight matrix $W \in \mathbb{R}^{d_a \times d_z}$ and a bias vector $b \in \mathbb{R}^{d_a}$ that maps a frozen latent action $z_t \in \mathbb{R}^{d_z}$ to the clipped ground-truth action $a_t \in \mathbb{R}^{d_a}$, with no gradient flowing back into the latent action model. The probe is fit on the training split and evaluated on the held-out split; we report

$$\mathrm{NMSE}(\tau) \;=\; \frac{\mathbb{E}_{\mathcal{D}^{\tau}_{\mathrm{eval}}}\left[\|W z_t + b - a_t\|_2^2\right]}{\mathrm{Var}_{\mathcal{D}^{\tau}}(a)}, \qquad \mathrm{Var}(a) \;=\; \frac{1}{d_a}\sum_{i=1}^{d_a} \mathrm{Var}\Big(a^{(i)}\Big), \tag{7}$$

where the denominator is the mean per-dimension variance of the clipped ground-truth actions on that task's training split, listed in Table 10. Under this normalization, $\mathrm{NMSE} = 1$ corresponds to a probe that predicts the mean action and $\mathrm{NMSE} = 0$ to perfect recovery, so the metric is comparable across tasks with different action scales.

Normalized return (NR). On DCS we generate rollouts with the decoded policy of Stage 3 for 100 episodes and report

$$\mathrm{NR}(\tau) \;=\; \frac{R_{\pi}(\tau)}{R_{\mathrm{exp}}(\tau)}, \tag{8}$$

where $R_{\pi}(\tau)$ is the mean undiscounted episode return of the learned policy and $R_{\mathrm{exp}}(\tau)$ is the expert return from Table 9. A value of 1 matches the data-collection expert.

Normalized success rate (NSR). On DMW we generate rollouts with the decoded policy of Stage 3 for 100 episodes and report

$$\mathrm{NSR}(\tau) \;=\; \frac{S_{\pi}(\tau)}{S_{\mathrm{exp}}(\tau)}, \tag{9}$$

where $S_{\pi}(\tau)$ is the success fraction of the learned policy and $S_{\mathrm{exp}}(\tau)$ the expert success rate from Table 9.

Probe accuracy. Atari actions are discrete, so NMSE is undefined. We instead fit a linear classifier on frozen latent actions under the same protocol as the probe in (7) and report its accuracy on the held-out split. Chance level is $1/|\mathcal{A}|$, i.e., 0.056 on Frostbite and 0.200 on Frogger. Normalized return on Atari follows (8) with the expert returns of Table 9.

Expert results. Table 9 lists the performance of the expert policies that collected the offline trajectories (Section B), measured with the evaluation protocol used for the downstream policies.

These numbers are the reference point for the Normalized Return (NR) on DCS and Atari and the Normalized Success Rate (NSR) on DMW: a value of 1.0 matches expert performance on a given task, and 0.0 corresponds to zero return or zero success.

Table 9: Expert performance of the data collection policies used for normalization: average return on DCS and Atari, and average success rate on DMW.

DCS

Task	Return
cheetah-run	837.67
hopper-hop	307.33
humanoid-walk	616.52
walker-run	738.37

Atari

Game	Return
Frostbite	3076.00
Frogger	278.00

DMW

Task	Success rate
button-press-topdown	1.000
door-open	0.917
drawer-close	1.000
drawer-open	1.000
pick-place	1.000
peg-insert-side	0.772
push	1.000
reach	1.000
window-open	1.000
window-close	1.000

Aggregation. For every task we first compute the mean and standard deviation across the three seeds. Per-benchmark numbers are obtained by averaging the per-task means, and separately averaging the per-task standard deviations, across the tasks of that benchmark. When we report a single number across both benchmarks, we average the two per-benchmark aggregates, so that DCS and DMW contribute equally regardless of the difference in the number of tasks.

Per-task action variance. Table 10 lists $\mathrm{Var}(a)$ for every evaluated task, which serves as the denominator of (7). Variances are computed on the clipped ground-truth actions of the released dataset and agree to numerical precision between the training and held-out splits, so the split subscript is dropped in (7); we report the value computed on the larger training split. They are also shared between the distractor-free and distracting variants of each benchmark, since the action stream is identical by construction in our synchronized data collection (Section B).

Table 10: Per-task action variance $\mathrm{Var}(a) = \frac{1}{d_a}\sum_i \mathrm{Var}(a^{(i)})$, computed on clipped ground-truth actions over the training split of the released dataset. Values are shared between the distractor-free and distracting variants of each benchmark, because the action stream is identical by construction (Section B). Used as the denominator of NMSE in (7).

Task	$\mathrm{Var}(a)$
DCS	
cheetah-run	0.6787
hopper-hop	0.7454
humanoid-walk	0.4221
walker-run	0.6729
DROID	
all episodes	0.0645

Task	$\mathrm{Var}(a)$
DMW	
button-press-topdown	0.2085
door-open	0.4266
drawer-close	0.2466
drawer-open	0.1483
peg-insert-side	0.2497
pick-place	0.2286
push	0.1311
reach	0.0372
window-close	0.3650
window-open	0.2979

G Per-Task Latent Action Quality

Per-task NMSE on DCS and DMW in the distractor and distractor-free settings; metric definition in Section F. The distractor results break down the aggregates of Table 1; the distractor-free results are reported only here.

Distractor setting. MaskLAM@GT and MaskLAM@SAM attain the lowest NMSE among label-free baselines on all four DCS tasks (Table 11) and all ten DMW tasks (Table 13), often by a 2–3× margin, and outperform LAOM-Labels on most tasks. The two MaskLAM variants differ by at most

0.032 NMSE on every task except hopper-hop, where MaskLAM@SAM has a high variance across seeds.

Distractor-free setting. On DCS (Table 11, columns 5–8), MaskLAM@GT outperforms every label-free baseline on average (0.205 vs. 0.297 for LAPO) and is on par with the privileged LAOM-Labels (0.211). On DMW (Table 12), MaskLAM ties LAPO, which is the strongest baseline in the absence of distractors. Without exogenous variance dominating the reconstruction target, the unmasked loss already places most gradient weight on agent-driven change.

Distracting Control Suite (NMSE ↓)

	Distractor				Distractor-free			
Method	**cheetah-run**	**hopper-hop**	**humanoid-walk**	**walker-run**	**cheetah-run**	**hopper-hop**	**humanoid-walk**	**walker-run**
LAPO	$0.3836_{\pm 0.0220}$	$0.6433_{\pm 0.0671}$	$0.6766_{\pm 0.0382}$	$0.3423_{\pm 0.0071}$	$0.1484_{\pm 0.0019}$	$0.1959_{\pm 0.0036}$	$0.5298_{\pm 0.2591}$	$0.3142_{\pm 0.2031}$
LAOM-Labels	$0.1652_{\pm 0.0020}$	$0.1933_{\pm 0.0067}$	$\mathbf{0.3112_{\pm 0.0025}}$	$0.1943_{\pm 0.0026}$	$0.1478_{\pm 0.0009}$	$\mathbf{0.1904_{\pm 0.0156}}$	$\mathbf{0.3379_{\pm 0.0072}}$	$\mathbf{0.1663_{\pm 0.0006}}$
LAOF	$0.3236_{\pm 0.0143}$	$0.4863_{\pm 0.0303}$	$0.5719_{\pm 0.0174}$	$0.6408_{\pm 0.0157}$	$0.5546_{\pm 0.0162}$	$0.7345_{\pm 0.0443}$	$0.6322_{\pm 0.0503}$	$0.6348_{\pm 0.0696}$
LAPO-Slots	$0.6615_{\pm 0.0212}$	$0.6950_{\pm 0.0051}$	$0.7270_{\pm 0.0052}$	$0.5444_{\pm 0.0859}$	$0.2892_{\pm 0.0314}$	$0.3613_{\pm 0.0066}$	$0.4673_{\pm 0.0088}$	$0.2884_{\pm 0.0020}$
MaskLAM@GT	$0.1171_{\pm 0.0007}$	$\mathbf{0.1853_{\pm 0.0256}}$	$0.3531_{\pm 0.0059}$	$\mathbf{0.1692_{\pm 0.0049}}$	$0.0910_{\pm 0.0017}$	$0.2093_{\pm 0.0031}$	$0.3380_{\pm 0.0041}$	$0.1811_{\pm 0.0163}$
MaskLAM@SAM	$\mathbf{0.1148_{\pm 0.0040}}$	$0.2601_{\pm 0.1434}$	$0.3852_{\pm 0.0007}$	$0.1704_{\pm 0.0008}$	$\mathbf{0.0859_{\pm 0.0028}}$	$0.2147_{\pm 0.0245}$	$0.3555_{\pm 0.0076}$	$0.1956_{\pm 0.0035}$

Table 11: Per-task NMSE on DCS in distractor (columns 1–4) and distractor-free (columns 5–8) settings. MaskLAM@GT and MaskLAM@SAM attain the lowest NMSE among label-free baselines under distractors, and MaskLAM@GT surpasses LAOM-Labels on three of four tasks; without distractors MaskLAM outperforms every label-free baseline on average and is on par with LAOM-Labels.

Distracting Meta-World - Distractor-free (NMSE ↓)

Method	**reach**	**push**	**pick-place**	**door-open**	**drawer-open**	**drawer-close**	**button-press-topdown**	**peg-insert-side**	**window-open**	**window-close**
LAPO	$\mathbf{0.0773_{\pm 0.0086}}$	$0.1332_{\pm 0.0076}$	$\mathbf{0.1105_{\pm 0.0015}}$	$\mathbf{0.0557_{\pm 0.0009}}$	$\mathbf{0.0399_{\pm 0.0018}}$	$0.0124_{\pm 0.0012}$	$0.0358_{\pm 0.0034}$	$0.2174_{\pm 0.0068}$	$0.0466_{\pm 0.0024}$	$\mathbf{0.0129_{\pm 0.0014}}$
LAOM-Labels	$0.1130_{\pm 0.0120}$	$0.1583_{\pm 0.0146}$	$0.1273_{\pm 0.0073}$	$0.1047_{\pm 0.0103}$	$0.0986_{\pm 0.0193}$	$0.0931_{\pm 0.0065}$	$0.1068_{\pm 0.0097}$	$0.1983_{\pm 0.0100}$	$0.0789_{\pm 0.0034}$	$0.0584_{\pm 0.0058}$
LAOF	$0.1254_{\pm 0.0049}$	$0.3574_{\pm 0.0437}$	$0.2276_{\pm 0.0235}$	$0.2055_{\pm 0.1099}$	$0.1483_{\pm 0.0398}$	$0.1249_{\pm 0.0104}$	$0.1680_{\pm 0.0855}$	$0.3118_{\pm 0.0223}$	$0.1192_{\pm 0.0191}$	$0.0605_{\pm 0.0113}$
LAPO-Slots	$0.1117_{\pm 0.0046}$	$0.2166_{\pm 0.0118}$	$0.1675_{\pm 0.0049}$	$0.1245_{\pm 0.0081}$	$0.0757_{\pm 0.0031}$	$0.0540_{\pm 0.0041}$	$0.0447_{\pm 0.0031}$	$\mathbf{0.1979_{\pm 0.0095}}$	$0.0678_{\pm 0.0053}$	$0.0292_{\pm 0.0058}$
MaskLAM@GT	$0.0837_{\pm 0.0063}$	$0.1376_{\pm 0.0047}$	$0.1290_{\pm 0.0093}$	$0.0763_{\pm 0.0017}$	$0.0439_{\pm 0.0019}$	$0.0137_{\pm 0.0026}$	$0.0341_{\pm 0.0018}$	$0.2417_{\pm 0.0079}$	$\mathbf{0.0399_{\pm 0.0014}}$	$0.0155_{\pm 0.0030}$
MaskLAM@SAM	$0.0794_{\pm 0.0057}$	$\mathbf{0.1271_{\pm 0.0018}}$	$0.1259_{\pm 0.0041}$	$0.0667_{\pm 0.0029}$	$0.0468_{\pm 0.0021}$	$\mathbf{0.0108_{\pm 0.0008}}$	$\mathbf{0.0292_{\pm 0.0021}}$	$0.2354_{\pm 0.0056}$	$0.0448_{\pm 0.0038}$	$0.0131_{\pm 0.0012}$

Table 12: Per-task NMSE on DMW (distractor-free). MaskLAM ties LAPO, the strongest label-free baseline without distractors, confirming that loss masking does not degrade alignment in the absence of exogenous variance.

Distracting Meta-World - Distractor (NMSE ↓)

Method	**reach**	**push**	**pick-place**	**door-open**	**drawer-open**	**drawer-close**	**button-press-topdown**	**peg-insert-side**	**window-open**	**window-close**
LAPO	$0.2906_{\pm 0.0087}$	$0.4812_{\pm 0.0028}$	$0.5470_{\pm 0.0110}$	$0.2312_{\pm 0.0063}$	$0.2612_{\pm 0.0087}$	$0.1715_{\pm 0.0067}$	$0.2978_{\pm 0.0131}$	$0.5117_{\pm 0.0176}$	$0.2021_{\pm 0.0140}$	$0.0987_{\pm 0.0051}$
LAOM-Labels	$0.1610_{\pm 0.0027}$	$0.2065_{\pm 0.0060}$	$0.1409_{\pm 0.0030}$	$0.1407_{\pm 0.0124}$	$0.1143_{\pm 0.0183}$	$0.1335_{\pm 0.0138}$	$0.1075_{\pm 0.0029}$	$0.2496_{\pm 0.0065}$	$0.0916_{\pm 0.0067}$	$0.0751_{\pm 0.0050}$
LAOF	$0.3392_{\pm 0.0244}$	$0.5003_{\pm 0.0645}$	$0.4891_{\pm 0.0497}$	$0.4487_{\pm 0.0802}$	$0.4893_{\pm 0.1146}$	$0.5076_{\pm 0.0681}$	$0.4376_{\pm 0.0176}$	$0.5331_{\pm 0.0436}$	$0.4662_{\pm 0.0529}$	$0.3180_{\pm 0.0530}$
LAPO-Slots	$0.2174_{\pm 0.0049}$	$0.4550_{\pm 0.2032}$	$0.2691_{\pm 0.0081}$	$0.2444_{\pm 0.0319}$	$0.2927_{\pm 0.0616}$	$0.1666_{\pm 0.0047}$	$0.2083_{\pm 0.1377}$	$0.3184_{\pm 0.0070}$	$0.1600_{\pm 0.0045}$	$0.1183_{\pm 0.0179}$
MaskLAM@GT	$\mathbf{0.1140_{\pm 0.0066}}$	$0.1441_{\pm 0.0080}$	$\mathbf{0.1291_{\pm 0.0005}}$	$\mathbf{0.0738_{\pm 0.0077}}$	$0.0548_{\pm 0.0021}$	$\mathbf{0.0156_{\pm 0.0003}}$	$\mathbf{0.0391_{\pm 0.0018}}$	$0.2538_{\pm 0.0068}$	$\mathbf{0.0409_{\pm 0.0008}}$	$\mathbf{0.0148_{\pm 0.0016}}$
MaskLAM@SAM	$0.1246_{\pm 0.0077}$	$\mathbf{0.1366_{\pm 0.0061}}$	$0.1316_{\pm 0.0077}$	$0.0789_{\pm 0.0068}$	$\mathbf{0.0524_{\pm 0.0046}}$	$0.0184_{\pm 0.0015}$	$0.0462_{\pm 0.0007}$	$\mathbf{0.2381_{\pm 0.0100}}$	$0.0605_{\pm 0.0022}$	$0.0161_{\pm 0.0018}$

Table 13: Per-task NMSE on DMW (distractor). MaskLAM@GT reduces NMSE by at least 2× relative to LAPO on every task, and a MaskLAM variant attains the lowest NMSE on all ten tasks, including the contact-rich pick-place, peg-insert-side, and push.

H Per-Task Downstream Performance

Per-task downstream performance on DCS and DMW in the distractor and distractor-free settings; metric definitions in Section F. The distractor results break down the aggregates of Table 1; the distractor-free results are reported only here. All numbers are mean ± standard deviation across three seeds.

Distractor setting. MaskLAM@GT and MaskLAM@SAM exceed every label-free baseline on every DCS task (Table 14), and MaskLAM@GT matches or exceeds LAOM-Labels on 7 of 10 DMW tasks (Table 16) despite using no action supervision in Stage 1.

Distractor-free setting. On DCS (Table 14, columns 5–8), LAOM-Labels is the top-performing method, but MaskLAM stays in reach and more than doubles the normalized return of LAPO (0.51 vs. 0.24). We hypothesize that even in clean scenes the unmasked FDM loss diverts part of the latent's

capacity toward reconstructing background structure instead of action-driven change, an effect the latent-dimension sweep of Figure 6a makes explicit. On DMW (Table 15), MaskLAM ties LAPO, which is the strongest baseline in the absence of distractors.

Distracting Control Suite (Mean NR ↑)

	Distractor				Distractor-free			
Method	**cheetah-run**	**hopper-hop**	**humanoid-walk**	**walker-run**	**cheetah-run**	**hopper-hop**	**humanoid-walk**	**walker-run**
LAPO	$0.2057_{\pm 0.0211}$	$0.0043_{\pm 0.0005}$	$0.0375_{\pm 0.0115}$	$0.0562_{\pm 0.0038}$	$0.6828_{\pm 0.0431}$	$0.1729_{\pm 0.0180}$	$0.0022_{\pm 0.0001}$	$0.0876_{\pm 0.0496}$
LAOM-Labels	$\mathbf{0.9415_{\pm 0.0018}}$	$\mathbf{0.5425_{\pm 0.0120}}$	$0.0448_{\pm 0.0030}$	$\mathbf{0.7364_{\pm 0.0578}}$	$0.9219_{\pm 0.0091}$	$\mathbf{0.7114_{\pm 0.0292}}$	$0.0019_{\pm 0.0001}$	$\mathbf{0.7981_{\pm 0.0275}}$
LAOF	$0.2721_{\pm 0.0529}$	$0.0561_{\pm 0.0189}$	$0.0082_{\pm 0.0022}$	$0.0391_{\pm 0.0040}$	$0.2131_{\pm 0.0403}$	$0.0281_{\pm 0.0155}$	$0.0022_{\pm 0.0000}$	$0.0452_{\pm 0.0119}$
LAPO-Slots	$0.1055_{\pm 0.0141}$	$0.0020_{\pm 0.0007}$	$0.0247_{\pm 0.0052}$	$0.0494_{\pm 0.0113}$	$0.1801_{\pm 0.0146}$	$0.1211_{\pm 0.0234}$	$0.0020_{\pm 0.0002}$	$0.2023_{\pm 0.0437}$
MaskLAM@GT	$0.7588_{\pm 0.0309}$	$0.2736_{\pm 0.0382}$	$\mathbf{0.0758_{\pm 0.0121}}$	$0.4022_{\pm 0.0381}$	$\mathbf{0.9362_{\pm 0.0286}}$	$0.6337_{\pm 0.0451}$	$0.0024_{\pm 0.0000}$	$0.4782_{\pm 0.0407}$
MaskLAM@SAM	$0.7384_{\pm 0.0342}$	$0.2164_{\pm 0.1050}$	$0.0645_{\pm 0.0086}$	$0.4130_{\pm 0.0136}$	$0.9167_{\pm 0.0151}$	$0.5628_{\pm 0.0733}$	$\mathbf{0.0024_{\pm 0.0002}}$	$0.4919_{\pm 0.0548}$

Table 14: Per-task NR on DCS in distractor (columns 1–4) and distractor-free (columns 5–8) settings. MaskLAM is the top-performing label-free method on every task in both settings; without distractors, all methods fail on humanoid-walk, and LAOM-Labels leads on hopper-hop and walker-run.

Distracting Meta-World - Distractor-free (Mean NSR ↑)

Method	**reach**	**push**	**pick-place**	**door-open**	**drawer-open**	**drawer-close**	**button-press-topdown**	**peg-insert-side**	**window-open**	**window-close**
LAPO	$0.5648_{\pm 0.1118}$	$\mathbf{0.9364_{\pm 0.0348}}$	$0.7194_{\pm 0.0555}$	$0.9212_{\pm 0.0212}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.4011_{\pm 0.0187}$	$0.9668_{\pm 0.0146}$	$\mathbf{1.0000_{\pm 0.0000}}$
LAOM-Labels	$0.6133_{\pm 0.0633}$	$0.1983_{\pm 0.0451}$	$0.0750_{\pm 0.0328}$	$0.9400_{\pm 0.0350}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.9967_{\pm 0.0058}$	$0.9900_{\pm 0.0000}$	$\mathbf{0.4567_{\pm 0.0961}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$
LAOF	$\mathbf{0.7297_{\pm 0.2191}}$	$0.4307_{\pm 0.2137}$	$0.3347_{\pm 0.1592}$	$\mathbf{0.9690_{\pm 0.0494}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.8317_{\pm 0.2916}$	$0.4140_{\pm 0.0233}$	$0.4507_{\pm 0.2192}$	$\mathbf{1.0000_{\pm 0.0000}}$
LAPO-Slots	$0.5076_{\pm 0.0798}$	$0.9242_{\pm 0.0284}$	$0.6704_{\pm 0.0571}$	$0.9468_{\pm 0.0150}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.2811_{\pm 0.1385}$	$0.9963_{\pm 0.0042}$	$\mathbf{1.0000_{\pm 0.0000}}$
MaskLAM@GT	$0.5430_{\pm 0.0188}$	$0.9171_{\pm 0.0379}$	$0.7054_{\pm 0.0556}$	$0.9213_{\pm 0.0389}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.3102_{\pm 0.0812}$	$0.9507_{\pm 0.0304}$	$0.9281_{\pm 0.1012}$
MaskLAM@SAM	$0.5375_{\pm 0.0591}$	$0.9143_{\pm 0.0100}$	$\mathbf{0.7220_{\pm 0.0368}}$	$0.9378_{\pm 0.0161}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.2369_{\pm 0.0570}$	$0.9372_{\pm 0.0324}$	$0.9958_{\pm 0.0039}$

Table 15: Per-task NSR on DMW (distractor-free). MaskLAM ties LAPO, the strongest baseline in the absence of distractors, indicating that loss masking induces no penalty when its target assumption (exogenous contamination) is absent.

Distracting Meta-World - Distractor (Mean NSR ↑)

Method	**reach**	**push**	**pick-place**	**door-open**	**drawer-open**	**drawer-close**	**button-press-topdown**	**peg-insert-side**	**window-open**	**window-close**
LAPO	$0.2100_{\pm 0.0211}$	$0.1715_{\pm 0.0461}$	$0.0560_{\pm 0.0207}$	$0.9118_{\pm 0.0247}$	$0.9979_{\pm 0.0029}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.9934_{\pm 0.0080}$	$0.0535_{\pm 0.0140}$	$0.9698_{\pm 0.0093}$	$0.9981_{\pm 0.0005}$
LAOM-Labels	$0.5250_{\pm 0.0527}$	$0.5250_{\pm 0.0482}$	$0.4433_{\pm 0.1079}$	$\mathbf{0.9483_{\pm 0.0153}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{0.3683_{\pm 0.1040}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$
LAOF	$0.3927_{\pm 0.0264}$	$0.3560_{\pm 0.1863}$	$0.2853_{\pm 0.1513}$	$0.9340_{\pm 0.0245}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.9200_{\pm 0.1213}$	$0.5333_{\pm 0.2197}$	$0.0847_{\pm 0.0133}$	$0.2617_{\pm 0.0480}$	$\mathbf{1.0000_{\pm 0.0000}}$
LAPO-Slots	$0.5321_{\pm 0.0531}$	$0.7736_{\pm 0.1015}$	$0.7330_{\pm 0.0292}$	$0.9124_{\pm 0.0058}$	$0.9965_{\pm 0.0061}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.9945_{\pm 0.0095}$	$0.3347_{\pm 0.0212}$	$0.9970_{\pm 0.0029}$	$\mathbf{1.0000_{\pm 0.0000}}$
MaskLAM@GT	$\mathbf{0.5451_{\pm 0.0455}}$	$\mathbf{0.9286_{\pm 0.0222}}$	$0.7771_{\pm 0.0511}$	$0.9076_{\pm 0.0319}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.2812_{\pm 0.0542}$	$0.9921_{\pm 0.0062}$	$\mathbf{1.0000_{\pm 0.0000}}$
MaskLAM@SAM	$0.4861_{\pm 0.0460}$	$0.9073_{\pm 0.0469}$	$\mathbf{0.8574_{\pm 0.0293}}$	$0.9133_{\pm 0.0347}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$\mathbf{1.0000_{\pm 0.0000}}$	$0.2966_{\pm 0.0696}$	$0.8578_{\pm 0.0865}$	$0.9994_{\pm 0.0010}$

Table 16: Per-task NSR on DMW (distractor). MaskLAM@GT matches or exceeds LAOM-Labels on 7 of 10 tasks and attains the highest average NSR of all methods, despite using no action supervision in Stage 1.

I Stronger Distractors and Distribution Shift

Our main DCS experiments use the *easy* distractor setting, following prior work on latent actions [31]. Here we probe two questions that this setting leaves open. First, does masking remain effective when the distractors become stronger and harder to separate from the agent, putting more pressure on the spatial separability assumption in (2)? Second, does a latent action model trained under milder distractors still produce aligned latent actions when the distractors at test time are stronger than those seen during training?

Setting. The *hard* dynamic setting of DCS [40] uses stronger color randomization, draws backgrounds from more DAVIS videos, and adds a dynamic camera. Because color randomization acts on the agent itself, the hard setting entangles agent appearance with the distractors more strongly than the easy setting of the main experiments. We evaluate two protocols. *Hard*: all stages are trained and evaluated on hard data. *Easy* $\rightarrow$ *Hard*: the LAM is trained on easy data only; its frozen IDM then labels hard data, on which the linear probe is fit and evaluated. This tests whether the latent action space still carries the action information under stronger, unseen distractors.

Results. Table 17 and Figure 10 report mean probe NMSE over the four DCS tasks. *Stronger distractors.* Masking remains effective in the hard setting: both MaskLAM variants attain the lowest error (0.236), ahead of LAOM-Labels (0.267), which uses ground-truth labels, and at less than half

of LAPO's error (0.531). Compared to the easy setting, MaskLAM@GT degrades only moderately (0.206 → 0.236), so the spatial separability assumption holds well enough even when distractors entangle more strongly with the agent's appearance. *Distribution shift.* When the LAM is trained on easy data only and evaluated on hard data, MaskLAM@GT (0.311) and MaskLAM@SAM (0.340) stay within noise of LAOM-Labels (0.325), while every label-free baseline remains above 0.55. Latent actions learned under mild distractors therefore remain informative under stronger ones, without action labels. Per-task results in Figure 11.

Table 17: Mean action-probe NMSE (↓) on DCS in the hard distractor setting. *Hard*: trained and evaluated on hard data. *Easy → Hard*: LAM trained on easy data, evaluated on hard data. MaskLAM leads in the hard setting and matches LAOM-Labels under shift without action labels.

Method	Hard	Easy → Hard
LAPO	$0.5307_{\pm 0.0497}$	$0.6212_{\pm 0.0144}$
LAOM-Labels	$0.2673_{\pm 0.0100}$	$0.3254_{\pm 0.0057}$
LAOF	$0.5774_{\pm 0.0294}$	$0.5520_{\pm 0.0156}$
LAPO-Slots	$0.8031_{\pm 0.0634}$	$0.7301_{\pm 0.0150}$
MaskLAM@GT	$0.2357_{\pm 0.0068}$	$\mathbf{0.3113_{\pm 0.0191}}$
MaskLAM@SAM	$\mathbf{0.2354_{\pm 0.0059}}$	$0.3398_{\pm 0.0493}$

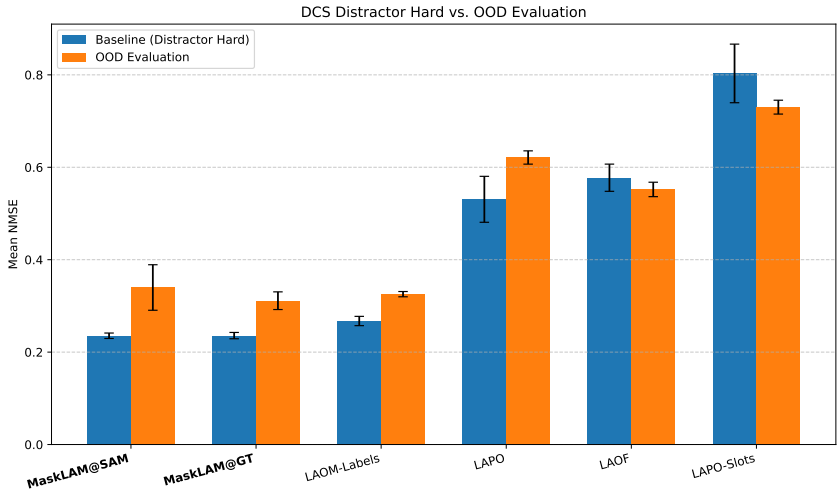


Figure 10: Mean action-probe NMSE on DCS (hard) in-distribution (blue) and after the easy-to-hard shift (orange). Error bars show the standard deviation over 3 seeds. Both MaskLAM variants stay at the level of LAOM-Labels under shift, while the label-free baselines remain above 0.55.

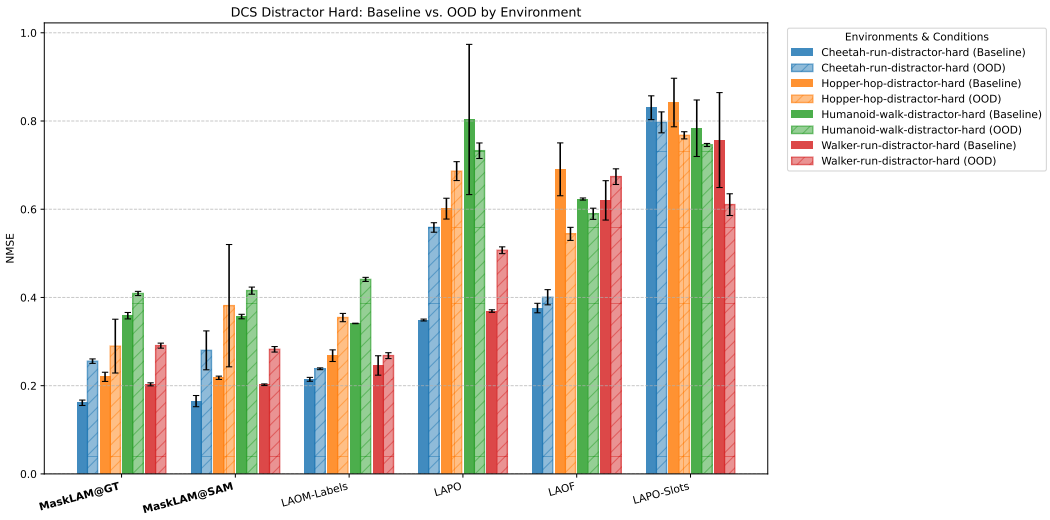


Figure 11: Per-task action-probe NMSE on DCS (hard), in-distribution and after the easy-to-hard shift. In the hard setting, MaskLAM attains the lowest error on cheetah-run, hopper-hop, and walker-run and is on par with LAOM-Labels on humanoid-walk. Under the easy-to-hard shift, MaskLAM and LAOM-Labels remain close on every task, while the label-free baselines stay well above both.

J Real-World Evaluation on DROID

Our main experiments follow the distractor benchmarks established in the latent action literature, where distractors are deliberately strong and precisely controlled. We additionally evaluate on DROID to test whether the benefit of masking carries over to real robot footage, where distractors such as clutter, changing lighting, and people moving through the frame arise naturally, and where masks come from a zero-shot segmenter applied to real images.

Protocol. All methods share the Stage 1 protocol of Section 5 and were trained to convergence. LAOM-Labels receives 2.5% of the ground-truth labels (65.5k) in Stage 1, as in its original paper. As a reference for the attainable probe error, we train a supervised IDM with the MaskLAM IDM architecture to regress ground-truth actions directly. DROID is an offline dataset, so we report probe NMSE only.

Results. Table 18 lists all methods. MaskLAM@SAM is the only LAM that improves on LAPO and closes 30% of the gap between LAPO and the supervised IDM. LAOF and LAPO-Slots are within noise of LAPO, and LAOM-Labels falls behind it. Absolute probe errors are high for every method, the supervised IDM included; we attribute this to the small backbones used throughout the paper, which we keep for comparability. Scaling the backbones is orthogonal to the masking objective.

Table 18: Action-probe NMSE on DROID (mean ± std over 3 seeds). The supervised IDM regresses ground-truth actions and serves as a reference. MaskLAM@SAM is the only LAM that improves on LAPO.

Method	**NMSE ↓**
Supervised IDM (reference)	$0.6837_{\pm 0.0019}$
LAPO	$0.8479_{\pm 0.0066}$
LAOM-Labels	$0.8822_{\pm 0.0028}$
LAOF	$0.8484_{\pm 0.0053}$
LAPO-Slots	$0.8544_{\pm 0.0081}$
MaskLAM@SAM	$\mathbf{0.7979}_{\pm \mathbf{0.0063}}$

K Extended UMAP Results

Figure 12 extends the single-dimension comparison of Figure 4 to all six action coordinates of cheetah-run on DCS (distractor). Each row corresponds to one method and each column to the ground-truth action dimension used to color the projection; a well-aligned latent exhibits a smooth color gradient along the manifold for every column.

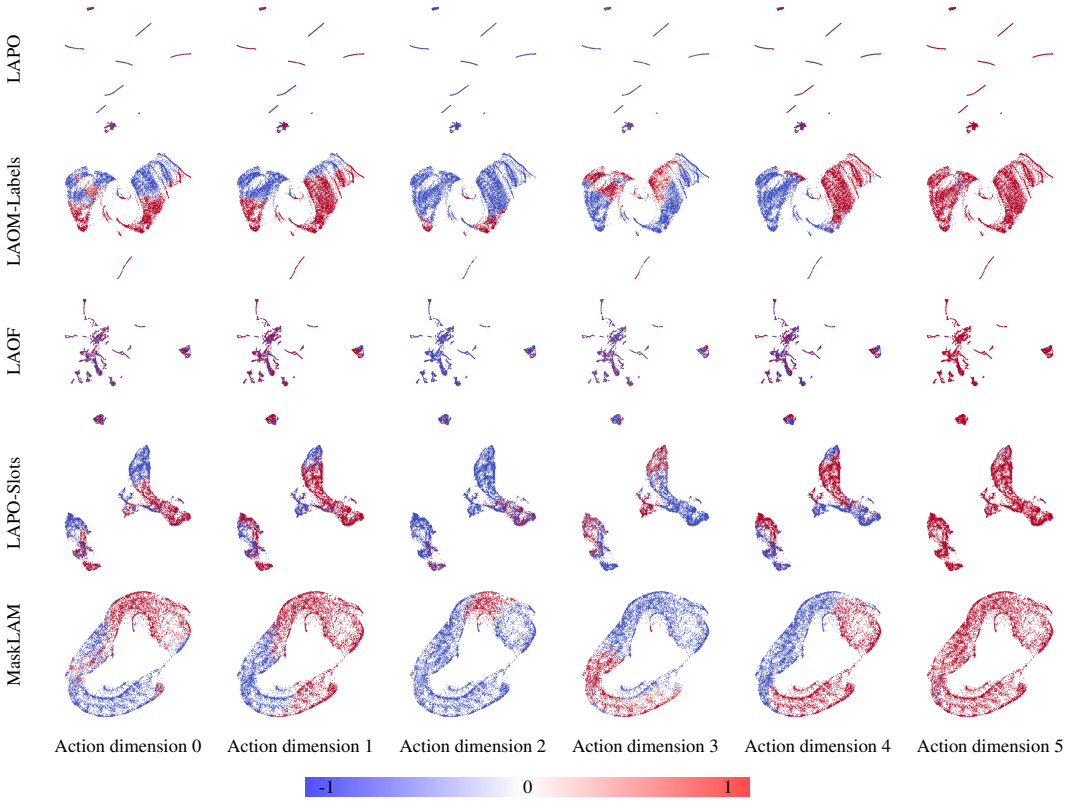


Figure 12: UMAP projections of learned latent actions on the DCS (distractor) dataset. Rows show LAPO, LAOM-Labels, LAOF, LAPO-Slots, and MaskLAM, respectively; columns show action dimensions 0–5.

MaskLAM recovers a coherent gradient on every action dimension. MaskLAM's manifold shows a clean red-to-blue transition for all six coordinates, with the gradient direction rotating from column to column as expected from a latent that is linearly aligned with the action vector. The per-dimension consistency confirms that the single-dimension view in the main text is representative.

LAOM-Labels matches the structure using action supervision. LAOM-Labels also forms a connected manifold with visible color separation on every dimension, consistent with its use of ground-truth actions as an auxiliary supervisory signal during pre-training (Section 5). MaskLAM reaches comparable per-dimension structure without consuming any action labels at the latent action learning stage.

LAPO and LAOF collapse into entangled fragments with no action structure. LAPO's projection breaks into thin disjoint strands with no visible gradient on any dimension, and LAOF mirrors this pattern at finer fragmentation. Under distractors, the unmasked reconstruction loss steers both latents toward exogenous variance instead of agent-controlled dynamics (Section 6), and the per-dimension view shows that this happens uniformly across the action vector.

LAPO-Slots recovers partial structure on a subset of dimensions. LAPO-Slots produces a single connected manifold with locally visible red-blue regions on some coordinates, but the gradient is non-monotonic or washes out on the remaining ones. Object-centric decomposition isolates the agent

at the input, yet the unmasked reconstruction loss still allocates the latent's capacity unevenly across action coordinates.

The per-dimension picture matches the aggregate probe error reported in Table 1: only MaskLAM and LAOM-Labels recover a latent action space that is linearly decodable to the full action vector, and MaskLAM does so without any ground-truth action supervision during pre-training.

L Extended Eigen-CAM Results

Eigen-CAM [30] produces a saliency map by taking the principal component of the activations of a chosen convolutional layer, highlighting the spatial regions that drive the network's prediction without requiring class labels or gradient signals. We apply it to the final convolutional block of the IDM and overlay the resulting heatmap on the input frame; warmer colors mark pixels that contribute most to the predicted latent action z_t. Since the IDM is the only module that consumes observations to produce z_t, its saliency directly exposes which pixels shape the latent action.

Figure 13 extends the single-frame comparison of Figure 5 to ten consecutive frames of a cheetah-run rollout in the distracting setting. LAPO's saliency drifts across the distractor background throughout the trajectory and rarely overlaps with the agent, confirming that without the masked loss the IDM grounds z_t in exogenous variation. MaskLAM's saliency stays locked onto the cheetah body and onto the ground-plane region directly under its feet across all ten frames, even though the supervisory mask covers only the cheetah itself (see Figure 8). The ground-plane response is informative: it is endogenous context, the surface the agent interacts with to produce its motion, and the IDM has learned to attend to it without ever being told to. On Frostbite and Frogger the same effect extends to the platforms that carry the agent (Figure 31).

This is the behavior we designed for in Section 4. The endogenous context varies from video to video and from frame to frame: the floor under a cheetah, a cup approached by a gripper, a door touched by a hand, a tool brought into contact with a workpiece. Specifying these regions a priori for every observation would require dense, task-aware annotation that defeats the purpose of label-free latent action learning, and any conservative mask drawn at input time would either erase useful interaction surfaces or admit distractors back in. Loss masking avoids this: the IDM still sees the full observation and is free to attend wherever reconstruction demands, while z_t is shaped only by reconstruction errors inside M_{t+1}. The boundary of endogenous context is learned from data, without a hand-crafted mask.

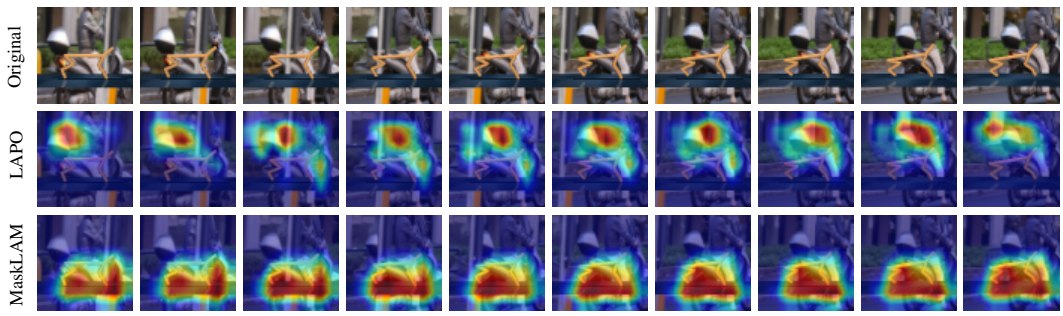


Figure 13: Eigen-CAM saliency on ten consecutive frames of a cheetah-run rollout on DCS (distractor). Warmer colors indicate higher influence on the predicted latent action. Top: Original frames. Middle: LAPO drifts across the distractor background. Bottom: MaskLAM stays on the cheetah and the ground-plane region under its feet, despite the supervisory mask covering only the cheetah.

M World Model Visualizations

The FDM (world model) in Stage 1 of the three-stage MaskLAM and LAPO training pipeline is trained jointly with the IDM (Section 4) and provides the only learning signal that shapes the latent action z_t: z_t must explain whatever the world model is asked to reconstruct beyond what the prior

frames already determine. Visualizing what each method's world model actually predicts therefore exposes what z_t has been forced to encode.

Figures 14 and 15 show LAPO's world model on DCS (distractor) and DMW (distractor). Reconstructions are nearly indistinguishable from the ground-truth next observation: the agent, the dynamic distractor video behind it, and the color randomization are all recovered at the pixel level. This is exactly what the unmasked reconstruction loss rewards. The world model has no way to know that the distractor video is exogenous, so it spends representational capacity on tracking it, and the IDM is pushed to write frame-to-frame distractor variation that the prior frames cannot predict into z_t. The pixel-perfect reconstruction is direct visual evidence of the failure mode that motivates our method.

Figures 16 and 17 show MaskLAM's world model on the same benchmarks. The agent and its immediate vicinity are reconstructed cleanly, while everything outside the supervisory mask collapses into a low-frequency texture with visible artifacts at the agent boundary. This is the intended behavior: with the loss masked to M_{t+1}, the world model receives no gradient for distractor pixels and stops modeling them, so z_t is no longer required to carry distractor information. The artifacts outside the mask are not a defect but the visible signature of the bottleneck we impose on the latent action.



Figure 14: LAPO world model predictions on DCS (distractor). Top row: predicted next observation $\hat{o}_{t+1}$. Bottom row: ground-truth next observation o_{t+1}. Ten consecutive transitions sampled from training. The unmasked reconstruction loss produces a pixel-accurate reproduction of the dynamic distractor video and the agent color randomization, evidence that z_t must carry distractor information for the world model to satisfy its objective.

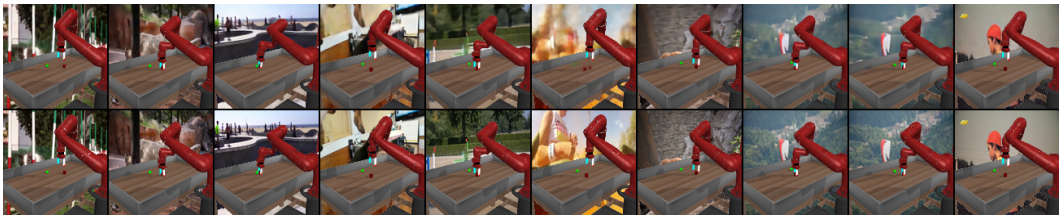


Figure 15: LAPO world model predictions on DMW (distractor). Layout matches Figure 14. Reconstructions track the distractor background with the same fidelity as the agent, confirming that the failure mode is benchmark-independent.



Figure 16: MaskLAM world model predictions on DCS (distractor). Top row: predicted next observation $\hat{o}_{t+1}$. Bottom row: ground-truth next observation o_{t+1}. With the reconstruction loss restricted to the supervisory mask M_{t+1}, the world model recovers the agent sharply while everything outside the mask collapses into a low-frequency texture with boundary artifacts.

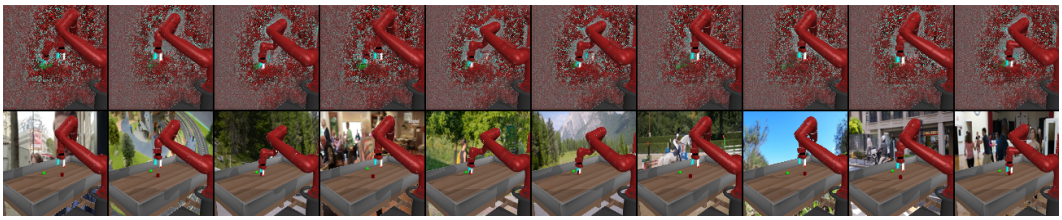


Figure 17: MaskLAM world model predictions on DMW (distractor). Layout matches Figure 16. The agent and its contact surfaces are reconstructed cleanly; the masked-out background is reconstructed only to a coarse extent, mirroring the DCS behavior.

N Latent Action Dimension

Per-environment breakdown of Figure 6a: NMSE on DCS (distractor) as a function of latent action dimension $d_z \in \{32, 64, 128, 256, 512, 1024\}$, with and without the masked loss (all other hyperparameters held fixed).

Compression at fixed alignment. A 256-dim masked latent matches or exceeds a 1024-dim unmasked latent on all four tasks (Figure 18), a $4\times$ reduction with no loss in alignment. The unmasked loss must allocate dimensions to both agent and distractor dynamics; the masked loss restricts the supervisory variance to agent pixels, shrinking the effective rank of the prediction target.

Low-dimensional regime. The gap widens at $d_z \leq 64$, where exogenous content under the unmasked loss crowds out the action signal.

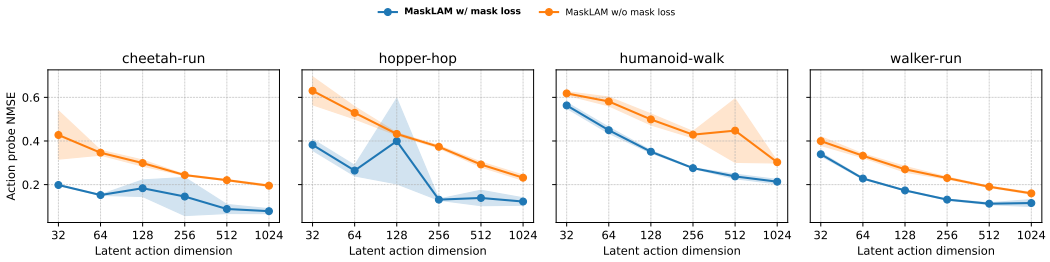


Figure 18: Per-task NMSE on DCS (distractor) as a function of latent action dimension (log scale). Blue: MaskLAM@GT; orange: same architecture with the masked loss removed. The masked loss gives lower NMSE at every d_z. A 256-dim masked model matches a 1024-dim unmasked one on all four tasks.

O Action-Label Budget

Per-environment breakdown of Figure 19: NR on DCS (distractor) and NSR on DMW (distractor) as a function of the Stage 3 label budget, swept from 2k to 128k labels with the Stage 1 latent action model held fixed. All numbers in this section are reported in the distractor setting.

DMW. MaskLAM@GT reaches near peak NSR with 2k labels on six of ten tasks (reach, push, pick-place, drawer-open, drawer-close, window-open), while LAPO and LAPO-Slots need nearly two orders of magnitude more labels to close the gap (Figure 20).

DCS. LAPO and LAPO-Slots fail to recover useful policies at any budget we tested; MaskLAM@GT reaches competitive returns from 16k–32k labels (Figure 21). LAOM-Labels remains ahead on hopper-hop and walker-run, although MaskLAM has lower probe NMSE there; as in the main text, we hypothesize that action supervision produces latents that hold up better under rollout-induced distribution shift.

Mask source. MaskLAM@SAM tracks MaskLAM@GT across all environments and label budgets, so swapping ground-truth for SAM-predicted masks costs no sample efficiency.

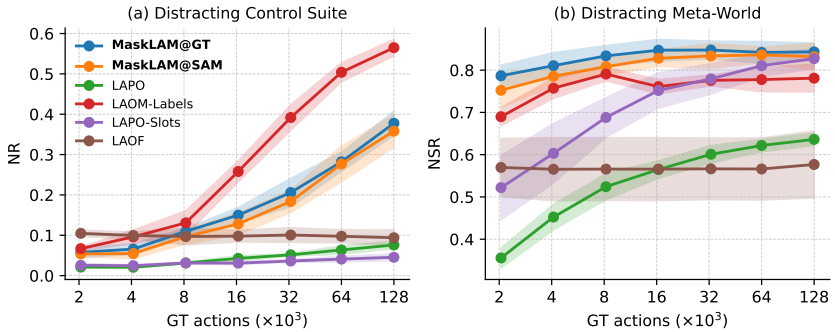


Figure 19: Sample efficiency as a function of the number of ground-truth action labels (log scale) available for action decoder fine-tuning. MaskLAM@GT reaches near peak performance with as few as 2k labels on DMW (b), while LAPO and LAPO-Slots require nearly two orders of magnitude more labels to close the gap, and on DCS (a) they fail to recover useful policies at any label budget.

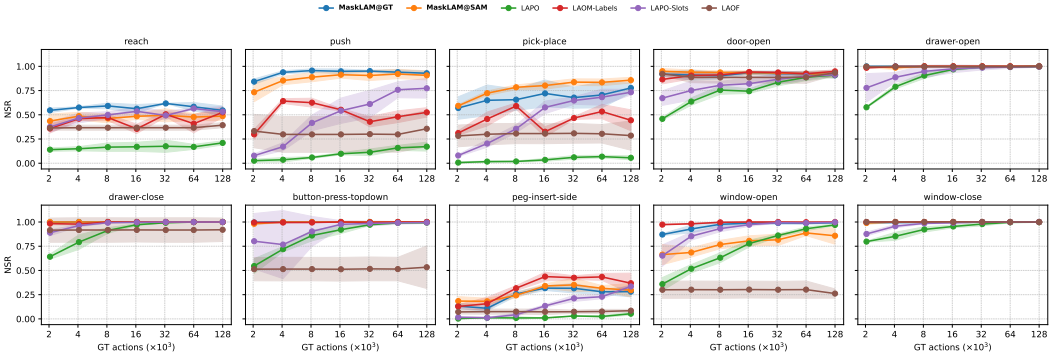


Figure 20: Per-task NSR on DMW (distractor) as a function of the Stage 3 label budget (log scale). MaskLAM reaches peak performance with 2k labels on six of ten tasks; LAPO and LAPO-Slots require nearly two orders of magnitude more labels to close the gap.

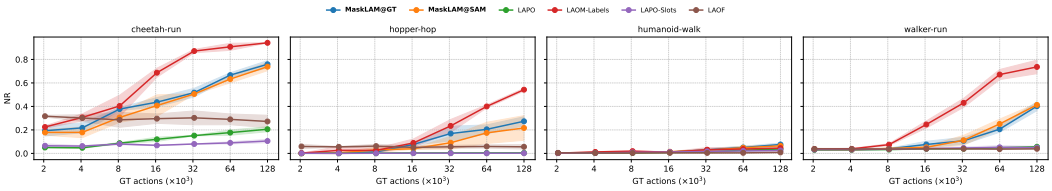


Figure 21: Per-task NR on DCS (distractor) as a function of the Stage 3 label budget (log scale). MaskLAM@GT saturates earlier than every label-free baseline; LAPO and LAPO-Slots fail to recover useful policies at any budget.

P Unified Latent Action Space Across Embodiments

Motivation. Pre-training on data from a single embodiment limits latent action models to the few domains where such data is plentiful. The broader goal is a LAM that learns from diverse video across many morphologies and captures their dynamics in one unified latent action space, leaving embodiment-specific adaptation to fine-tuning [46, 9]. MaskLAM fits this goal naturally: the mask specifies whose transition the latent must explain, while the objective itself is identical for every embodiment. We therefore test whether a single MaskLAM, pre-trained on a mixture of embodiment data, learns a unified latent action space that can be fine-tuned to each of them.

Protocol. We pre-train a single Stage 1 LAM, MaskLAM@GT (unified), on a mixture of all four DCS (distractor) embodiments: cheetah, hopper, humanoid, and walker. Their action spaces differ in size, from 4 dimensions (hopper) to 21 (humanoid). The total transition budget equals that of one single-embodiment run, so each embodiment contributes a quarter of its usual data. The unified IDM then labels the data of every embodiment with latent actions, and Stages 2 and 3 fine-tune a latent policy and an action decoder per embodiment, exactly as for the embodiment-specific models. The latent action dimension is $d_z = 128$ throughout.

Results. Tables 19 and 20 report per-task probe NMSE and normalized return. Fine-tuned to each embodiment, the unified latent space matches the four embodiment-specific MaskLAM@GT models on average in both metrics (NMSE 0.2082 vs. 0.2061, NR 0.3686 vs. 0.3776), although it sees only a quarter of each embodiment's data, and it remains ahead of every label-free baseline trained per embodiment. The largest per-task gain is on hopper-hop (NMSE 0.170 vs. 0.185, NR 0.324 vs. 0.274), while humanoid-walk and walker-run reach slightly lower returns. One masked LAM can thus serve as a shared pre-trained model from which embodiment-specific policies are obtained with the standard fine-tuning stages.

Table 19: Per-task action-probe NMSE (↓) on DCS (distractor) for a unified latent action space. MaskLAM@GT (unified) is a single LAM pre-trained on all four embodiments with a quarter of the per-embodiment data and probed per task; it matches the embodiment-specific MaskLAM@GT models on average.

Method	cheetah-run	hopper-hop	humanoid-walk	walker-run	Mean
LAPO	$0.3836_{\pm 0.0220}$	$0.6433_{\pm 0.0671}$	$0.6766_{\pm 0.0382}$	$0.3423_{\pm 0.0071}$	$0.5114_{\pm 0.0336}$
LAOM-Labels	$0.1652_{\pm 0.0020}$	$0.1933_{\pm 0.0067}$	$\mathbf{0.3112_{\pm 0.0025}}$	$0.1943_{\pm 0.0026}$	$0.2160_{\pm 0.0034}$
LAOF	$0.3236_{\pm 0.0143}$	$0.4863_{\pm 0.0303}$	$0.5719_{\pm 0.0174}$	$0.6408_{\pm 0.0157}$	$0.5056_{\pm 0.0194}$
LAPO-Slots	$0.6615_{\pm 0.0212}$	$0.6950_{\pm 0.0051}$	$0.7270_{\pm 0.0052}$	$0.5444_{\pm 0.0859}$	$0.6570_{\pm 0.0293}$
MaskLAM@GT	$0.1171_{\pm 0.0007}$	$0.1853_{\pm 0.0256}$	$0.3531_{\pm 0.0059}$	$\mathbf{0.1692_{\pm 0.0049}}$	$\mathbf{0.2061_{\pm 0.0093}}$
MaskLAM@SAM	$\mathbf{0.1148_{\pm 0.0040}}$	$0.2601_{\pm 0.1434}$	$0.3852_{\pm 0.0007}$	$0.1704_{\pm 0.0008}$	$0.2326_{\pm 0.0372}$
MaskLAM@GT (unified)	$0.1246_{\pm 0.0031}$	$\mathbf{0.1702_{\pm 0.0055}}$	$0.3547_{\pm 0.0038}$	$0.1832_{\pm 0.0037}$	$0.2082_{\pm 0.0040}$

Table 20: Per-task normalized return (↑) on DCS (distractor) for a unified latent action space. Latent policies and action decoders fine-tuned per embodiment on top of the unified latent space match the embodiment-specific MaskLAM@GT models on average.

Method	cheetah-run	hopper-hop	humanoid-walk	walker-run	Mean
LAPO	$0.2057_{\pm 0.0211}$	$0.0043_{\pm 0.0005}$	$0.0375_{\pm 0.0115}$	$0.0562_{\pm 0.0038}$	$0.0759_{\pm 0.0092}$
LAOM-Labels	$\mathbf{0.9415_{\pm 0.0018}}$	$\mathbf{0.5425_{\pm 0.0120}}$	$0.0448_{\pm 0.0030}$	$\mathbf{0.7364_{\pm 0.0578}}$	$\mathbf{0.5663_{\pm 0.0186}}$
LAOF	$0.2721_{\pm 0.0529}$	$0.0561_{\pm 0.0189}$	$0.0082_{\pm 0.0022}$	$0.0391_{\pm 0.0040}$	$0.0939_{\pm 0.0195}$
LAPO-Slots	$0.1055_{\pm 0.0141}$	$0.0020_{\pm 0.0007}$	$0.0247_{\pm 0.0052}$	$0.0494_{\pm 0.0113}$	$0.0454_{\pm 0.0078}$
MaskLAM@GT	$0.7588_{\pm 0.0309}$	$0.2736_{\pm 0.0382}$	$\mathbf{0.0758_{\pm 0.0121}}$	$0.4022_{\pm 0.0381}$	$0.3776_{\pm 0.0298}$
MaskLAM@SAM	$0.7384_{\pm 0.0342}$	$0.2164_{\pm 0.1050}$	$0.0645_{\pm 0.0086}$	$0.4130_{\pm 0.0136}$	$0.3581_{\pm 0.0404}$
MaskLAM@GT (unified)	$0.7614_{\pm 0.0703}$	$0.3240_{\pm 0.0160}$	$0.0389_{\pm 0.0137}$	$0.3500_{\pm 0.0910}$	$0.3686_{\pm 0.0478}$

Q Agent Occlusion

Real videos rarely keep the full agent in frame: limbs leave the camera view, foreground objects pass in front of the agent, and contact-rich motion regularly occludes joints behind other parts of the body. A practical latent action model has to keep producing well-aligned latents under this regime. We probe this on the DCS (distractor) by occluding a controlled fraction of the agent at training and evaluation time, and reporting the action-probe NMSE of (7) per task.

Occlusion procedure. For every video sample we synthesize a single random axis-aligned occlusion rectangle anchored to the agent. The construction is as follows:

1. Take the union of the ground-truth foreground mask over the frames of the video sample, $U = \bigvee_t M_t$, and compute its tight axis-aligned bounding box B of area A_B.

2. Sample a target occlusion area $A = \rho\, A_B$, where $\rho \in [0, 1]$ is the occlusion fraction reported on the x-axis of Figure 22.
3. Sample a log-uniform aspect ratio $r \sim \exp(\mathcal{U}(-\frac{1}{2}, \frac{1}{2}))$ and derive height $h = \sqrt{Ar}$ and width $w = A/h$, both clamped to B so the rectangle never leaves the agent footprint.
4. Sample a uniformly random top-left corner inside the remaining slack of B to obtain the final rectangle R.
5. Apply R identically to every frame of the video sample: overwrite the observation pixels inside R with a constant gray fill (corresponding to a 0.0 fill value after normalization) and zero the corresponding entries of the segmentation mask.

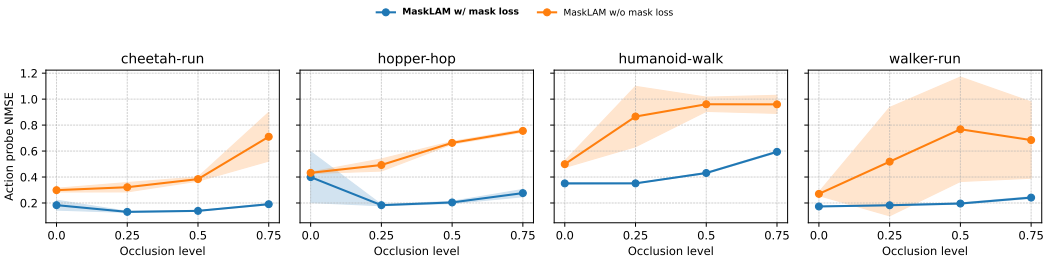


Figure 22: NMSE as a function of agent occlusion fraction on the DCS ($d_z = 128$). The x-axis is the fraction ρ of the agent's bounding-box area occluded by a random gray rectangle during training and evaluation (0% = no occlusion, 75% = three-quarters of agent pixels removed). Blue lines: MaskLAM with masked loss; orange lines: same architecture with the masked loss removed. With the masked loss, NMSE stays nearly flat and tightly concentrated across the full occlusion range on all four tasks; without it, error more than doubles and variance increases sharply, because the growing occlusion patch reduces the action-driven fraction of the unmasked reconstruction target while distractor variance remains unchanged.

The rectangle is resampled per trajectory and per training step, so the model never memorizes a fixed occlusion; held-out samples are occluded with the same procedure. Zeroing the mask inside R mirrors the behavior of any off-the-shelf segmenter, which simply fails to label pixels it cannot see; the IDM and FDM therefore receive exactly the inputs and supervisory signal they would receive in a deployment where occluded body parts disappear from the predicted segmentation.

Action-probe NMSE under occlusion. Figure 22 sweeps ρ from 0% to 75% for both MaskLAM and an ablation that removes the masked loss. With the masked loss the curve stays nearly flat across the full range on every environment, while removing it more than doubles the probe error on cheetah-run and walker-run and degrades the other two environments by a similar margin. The mask-loss curve is also visibly tighter across seeds.

Why masking helps under occlusion. Occluding the agent does not add new exogenous variance, but it shrinks the visible agent footprint and therefore the action-driven component of the FDM target, while distractor variance behind and around the agent stays the same. Under the unmasked loss the action-to-noise ratio of the supervisory target degrades as ρ grows, and the optimization increasingly pressures z_t to encode the distractor variance that now dominates the residual. This is the linearized PCA picture of Zhang et al. [49] applied to a target whose endogenous content has been thinned: as the agent signal in the target shrinks, the principal direction of frame-to-frame change tilts toward the distractor, and so does z_t. Under the masked loss the occluded rectangle is removed from M_{t+1} before the residual is computed, so the target contains only visible agent pixels at every ρ. The action-to-noise ratio of the supervisory signal therefore stays constant regardless of how much of the agent is hidden, and z_t keeps encoding only action-driven variance. Figures 23 and 24 show this at the pixel level. Without the masked loss the FDM continues to reproduce the full scene, occlusion patch, cheetah, and distractor video, pixel-accurately at every ρ, so z_t keeps carrying all of that variance even as the action-relevant share of the target shrinks. With the masked loss the FDM is asked only to reconstruct visible agent pixels and does so cleanly up to 75% occlusion, while everything outside M_{t+1} collapses to the same low-frequency texture seen in Figure 16. Because zeroing the mask inside R is exactly what an off-the-shelf segmenter does on partially visible agents, this experiment also demonstrates MaskLAM's robustness to real-world segmentation behavior.

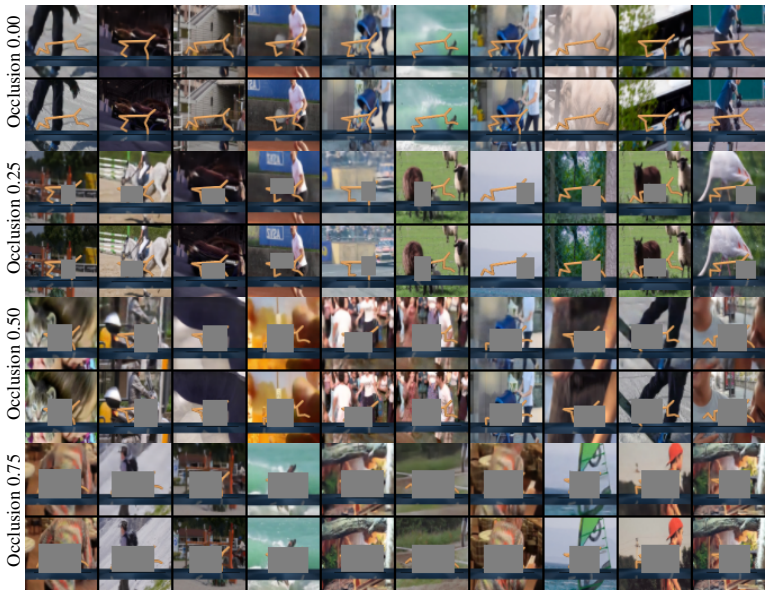


Figure 23: FDM reconstructions of MaskLAM *without* the masked loss objective on cheetah-run (DCS, distractor) under increasing agent occlusion. Rows correspond to occlusion fractions 0%, 25%, 50%, and 75% from top to bottom. Within each row, the top sub-row shows the predicted next observation $\hat{o}_{t+1}$ and the bottom sub-row shows the ground-truth o_{t+1} with the gray occlusion patch overlaid on the cheetah. Without the masked loss, the world model is forced to reconstruct occlusion pixels.



Figure 24: FDM reconstructions of MaskLAM *with* the masked loss objective on cheetah-run (DCS, distractor) under increasing agent occlusion. Layout matches Figure 23: rows correspond to occlusion fractions 0%, 25%, 50%, and 75%, and within each row the top sub-row is the predicted next observation $\hat{o}_{t+1}$ while the bottom sub-row is the ground-truth o_{t+1} with the gray occlusion patch on the cheetah. Restricting reconstruction to the supervisory mask keeps the prediction inside the mask robust to occlusion, recovering the visible cheetah pixels cleanly up to 75% occlusion and mirroring the action-probe NMSE trend in Figure 22.

R MASK QUALITY

In practice, supervisory masks come from an off-the-shelf segmentation model whose boundaries are never pixel-perfect. To quantify how mask quality affects latent action alignment, we perturb ground-truth masks via morphological dilation (expansion) and erosion (shrinkage) with structuring element radii of 0 to 3 pixels. This drops the mask mIoU from 1.0 down to approximately 0.4–0.6 depending on the task and perturbation direction. We retrain MaskLAM@GT from scratch with each perturbed mask set and report the resulting NMSE on the DMW benchmark. We also report baselines as dashed horizontal reference lines to contextualize the absolute error level.

We do not separately perturb MaskLAM@SAM because SAM 2.1 masks already serve as the "imperfect mask" condition: the comparison between MaskLAM@GT (radius 0) and MaskLAM@SAM in Table 1 shows they are within noise, which shows that real segmentation errors from a foundation model are tolerable, even on simulated images that differ from the real-world data these models were trained on. The morphological perturbation experiment isolates a controlled, interpretable axis of degradation beyond what SAM produces.

Dilation vs. erosion. Mask expansion (dilation) includes a growing border of background pixels in the supervisory target. This introduces a small amount of exogenous signal into the loss, but the impact on NMSE is mild. Mask shrinkage (erosion) removes agent boundary pixels, reducing the informative area without adding exogenous noise; here the effect is similarly benign. Across all 10 DMW tasks and both perturbation directions, MaskLAM@GT remains below all label-free baselines at every radius tested, confirming that pixel-perfect segmentation is not required. Mask extraction methods with a systematic boundary bias, for example coarse prompts that over-segment by 1–2 pixels, are therefore tolerable.

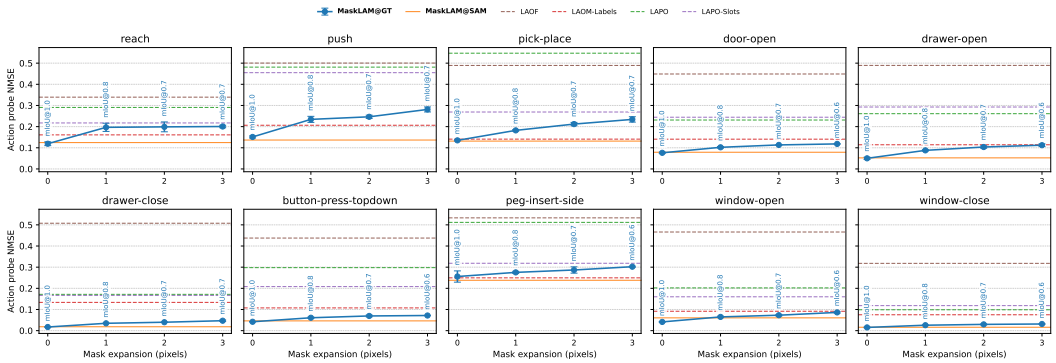


Figure 25: NMSE as a function of morphological dilation radius (0–3 pixels) applied to ground-truth masks before MaskLAM@GT training on the DMW. Each subplot is one task; the x-axis is the dilation radius in pixels (0 = unperturbed ground-truth mask); the y-axis is NMSE. Dashed horizontal lines show baselines. MaskLAM@GT remains below all label-free baselines across the entire perturbation range, indicating that including a thin border of background pixels in the loss does not meaningfully degrade latent action quality.

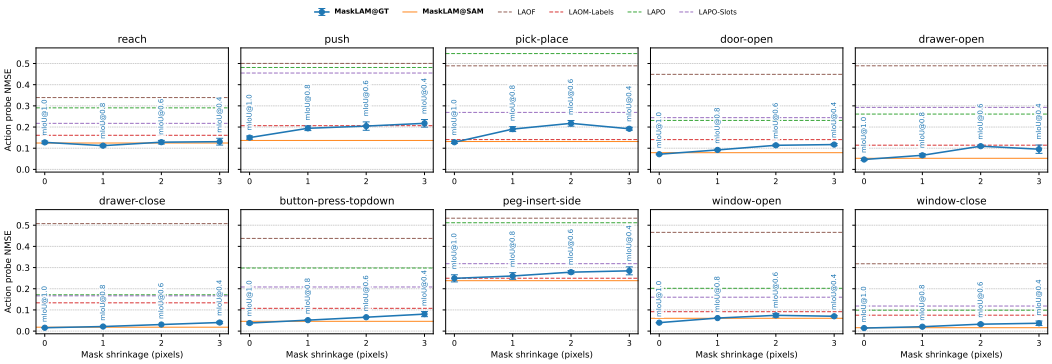


Figure 26: NMSE as a function of morphological erosion radius (0–3 pixels) applied to ground-truth masks before MaskLAM@GT training on the DMW. Layout matches Figure 25. Erosion removes agent boundary pixels, reducing the supervisory area without introducing exogenous signal. MaskLAM@GT degrades gracefully and stays below all label-free baselines even at radius 3, where mIoU drops below 0.5 on several tasks.

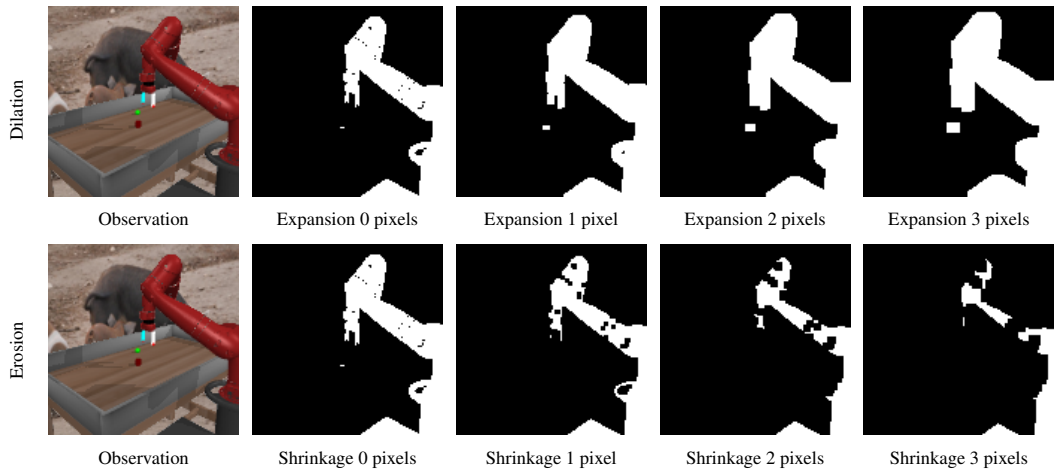


Figure 27: Qualitative visualization of morphological mask perturbations on a representative DMW frame (window-open task, distractor setting). Top row: progressive dilation of the ground-truth mask by 0–3 pixels, which expands the white supervisory region into the background. Bottom row: progressive erosion by 0–3 pixels, which shrinks the supervisory region inward from the boundary. Even at radius 3 the mask retains the coarse agent silhouette, explaining the graceful degradation observed in Figures 25 and 26.

S Extended Ablation Results

Per-environment breakdown of the aggregated ablation in Figure 7a; variant definitions in Section 6.1. The findings of Section 6.1 hold per task on DCS (Figure 30) and DMW (Figure 29), and per benchmark suite (Figure 28), with one exception: the mask channel helps on DCS but not on DMW.

Loss masking is the principal contributor on every task. Removing the masked loss more than doubles NMSE on nearly all DCS and DMW tasks.

The mask channel helps on DCS but slightly hurts on DMW. With loss masking active, removing the mask channel increases NMSE on every DCS task, with the largest effect on cheetah-run and hopper-hop. We attribute the per-task variation to the thin limb geometry of these agents, where an explicit spatial prior helps the model isolate the agent from the dynamic background. On DMW, the channel slightly hurts performance instead, as the larger and more compact arm silhouettes already

provide a sufficient prior. Without loss masking, the channel makes no meaningful difference on any task: it is useful only when the loss already restricts gradients to the masked region.

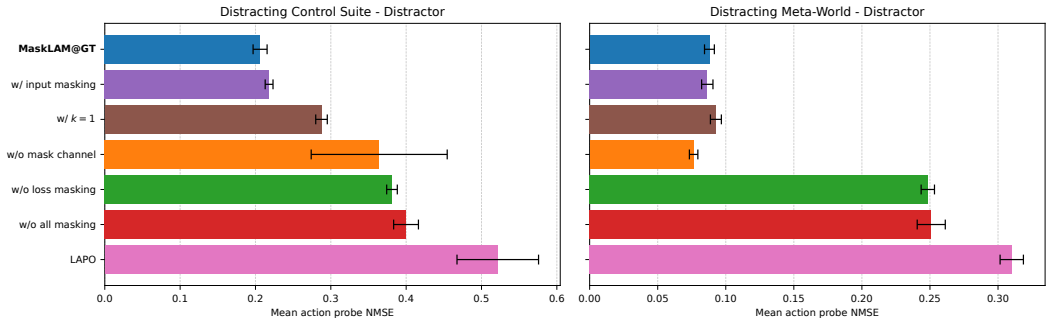


Figure 28: Ablation of MaskLAM components aggregated per benchmark suite (DCS left, DMW right). Loss masking is the principal contributor on both suites; the mask channel helps on DCS but slightly hurts on DMW; multi-step IDM adds a smaller orthogonal gain.

Multi-step IDM contributes a roughly additive gain. Setting $k{=}1$ degrades NMSE by a similar margin whether or not masking is present, consistent with the decorrelation argument of Lamb et al. [24] and confirming that multi-step prediction complements the spatial intervention.

Input masking suffices only for self-actuated agents. On every DCS and DMW task, the NMSE of *w/ input masking* lies within one standard deviation of MaskLAM@GT, consistent with the aggregate result in Section 6.1. In both benchmarks the agents are self-actuated: s^{agent}_{t+1} is determined by s^{agent}_{t} and a_t, so pixels outside M_t carry little additional information about the action, and removing them from the input costs little. Under loss masking, the IDM nevertheless uses this context, attending to ground-contact pixels outside M_{t+1} (Figure 13). The two variants separate once s^{agent}_{t+1} also depends on s^{ctx}_{t}: on Frostbite and Frogger, where the environment influences the agent position, loss masking improves probe accuracy from 0.561 to 0.653 and normalized return from 0.333 to 0.943 (Section T.2).

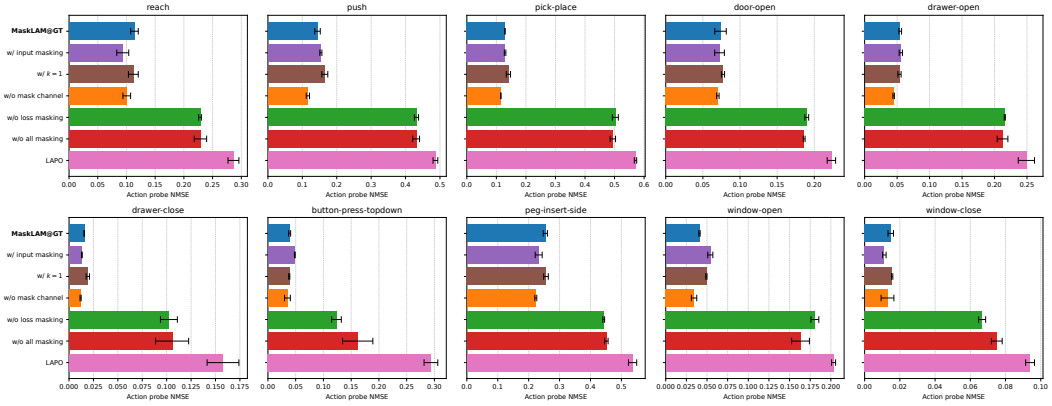


Figure 29: Per-task ablation on DMW (distractor). Loss masking is the principal contributor across all ten tasks; the mask channel slightly hurts performance, as the compact arm silhouettes already provide a sufficient spatial prior; multi-step IDM adds a smaller orthogonal gain.

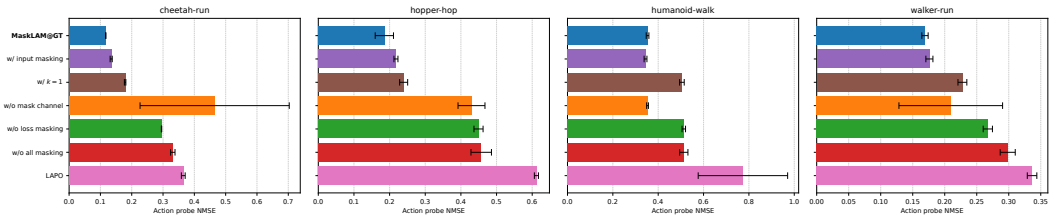


Figure 30: Per-task ablation on DCS (distractor). The hierarchy is consistent across cheetah-run, hopper-hop, humanoid-walk, and walker-run: loss masking is the principal contributor, followed by the mask channel, followed by multi-step IDM. The mask channel's contribution is largest on cheetah-run and hopper-hop, where the thin limb geometry makes an explicit spatial prior most useful.

T Action-Relevant Context Outside the Mask

MaskLAM restricts the prediction target to agent pixels but leaves the full observation visible to the IDM and FDM (Section 4). This section motivates that choice formally (Section T.1), measures its effect on agents moved by their environment (Section T.2), and tests whether manipulated objects should be part of the target (Section T.3).

T.1 Why Loss Masking Preserves Endogenous Context

Loss masking removes reconstruction pressure from all pixels outside the agent mask, including endogenous context, but keeps these pixels visible to the IDM and FDM. We show why this matters on both sides of the LAM. Let $X_t = (s_t^{\text{agent}}, s_{t+1}^{\text{agent}})$ denote the agent transition and $C_t = (s_t^{\text{ctx}}, s_{t+1}^{\text{ctx}})$ the endogenous context, which lies outside M_t by the spatial separability assumption (2).

Inverse dynamics benefits from context. This paragraph proves Proposition 1. Context is informative about the action beyond the agent transition exactly when $I(a_t; C_t \mid X_t) > 0$, or equivalently $p(a_t \mid X_t, C_t) \neq p(a_t \mid X_t)$. For a Bayes-optimal predictor of a_t, the gain from observing C_t is

$$\mathcal{R}_{\text{agent}} - \mathcal{R}_{\text{full}} = \begin{cases} H(a_t \mid X_t) - H(a_t \mid X_t, C_t) = I(a_t; C_t \mid X_t) & \text{(log loss)}, \\ \mathbb{E}\big[\mathrm{Var}\big(\mathbb{E}[a_t \mid X_t, C_t] \mid X_t\big)\big] & \text{(squared error)}, \end{cases} \tag{10}$$

where the second case follows from the law of total variance. Both gains are non-negative. The first is positive if and only if $I(a_t; C_t \mid X_t) > 0$, and the second if and only if $\mathbb{E}[a_t \mid X_t, C_t] \neq \mathbb{E}[a_t \mid X_t]$ with positive probability. This holds for supervised inverse dynamics and motivates why the IDM should see the full observation, but it does not yet explain why hiding context harms latent action learning.

Hiding context from the FDM contaminates the latent action. The masked loss (3) reconstructs the agent's next state $Y_t = s_{t+1}^{\text{agent}}$. Assume $Y_t = f(s_t^{\text{agent}}, a_t, c_t)$, where c_t is the part of the context visible in the FDM input, for example the velocity of a platform carrying the agent, and that c_t affects Y_t given the action, i.e., $\mathbb{E}\big[\mathrm{Var}(Y_t \mid s_t^{\text{agent}}, a_t)\big] > 0$. With loss masking, the FDM observes $(s_t^{\text{agent}}, c_t)$, so the latent $z_t = a_t$ already attains zero reconstruction error, and the objective exerts no pressure to encode c_t in z_t. With input masking, the FDM observes only s_t^{agent}. Consider any latent that carries no information about the context beyond the action, $z_t \perp c_t \mid (s_t^{\text{agent}}, a_t)$. Then conditioning on z_t adds no information about Y_t beyond $(s_t^{\text{agent}}, a_t)$, and the reconstruction error of the Bayes-optimal FDM is bounded by

$$\mathbb{E}\big[\mathrm{Var}(Y_t \mid s_t^{\text{agent}}, z_t)\big] \;\geq\; \mathbb{E}\big[\mathrm{Var}(Y_t \mid s_t^{\text{agent}}, a_t)\big] \;>\; 0. \tag{11}$$

Any latent that reduces the error below this bound must therefore violate the independence, so $I(z_t; c_t \mid s_t^{\text{agent}}, a_t) > 0$: under input masking, the objective drives the latent action to encode context instead of, or in addition to, the action. Conversely, masking only the IDM input avoids this pressure but restricts z_t to a function of X_t, so the latent cannot separate the action from context-driven motion and loses the information in (10). Only masking the target keeps both the IDM and the FDM free to use the context. This is the mechanism behind the Atari results (Section T.2), where

c_t is the motion of the floe, log, or turtle carrying the agent. It also explains why agent-only targets suffice in manipulation (Section T.3): object states belong to the context, which remains visible under loss masking and therefore does not need to enter the target.

T.2 Agents Moved by Their Environment: Frostbite and Frogger

Motivation. In DCS and DMW the agents are mostly self-actuated: the next agent state is determined by the current agent state and the action, and input masking performs on par with loss masking (Section 6.1). We therefore select two Atari games [3] in which the environment exerts forces on the agent. In Frostbite the agent jumps across ice floes that drift laterally; while it stands on a floe, its position changes without any action. In Frogger the agent crosses a road and then rides logs and turtles across a river, where its displacement is again partly imposed by the platform it occupies. In the notation of Section T.1, the platform state is part of C_t and $I(a_t; C_t \mid X_t) > 0$: the same agent displacement can result from different actions depending on how the platform moves.

Setup. Data collection, observations, and ground-truth agent masks are described in Section B.3; floes, logs, turtles, and cars lie outside the mask. We compare MaskLAM@GT with the *w/ input masking* variant of Section 6.1, which zeroes all pixels outside M_t before the IDM and FDM. Because the action spaces are discrete, we report probe accuracy instead of NMSE (Section F), together with normalized return.

Results. Table 21 shows that loss masking recovers the executed action more reliably ($+12.0$ points on Frostbite, $+6.4$ on Frogger), and the gap widens downstream: loss masking reaches near-expert return in both games, whereas input masking collapses on Frogger (0.038) and is unstable across seeds on Frostbite (± 0.48). The difference arises in the FDM. Input masking zeroes the floe, log, or turtle in the inputs of both IDM and FDM, so the FDM cannot predict the platform-induced drift of the agent from what it sees. The only way to reduce the reconstruction error inside M_{t+1} is then to encode the drift in z_t, which mixes platform motion into the latent action and degrades the decoded policy. Loss masking removes the same pixels only from the target: the FDM predicts the drift from the visible platform, so z_t only needs to encode the action. The Eigen-CAM maps in Figure 31 are consistent with this account: saliency covers the agent and the platforms that move it, outside M_{t+1}. In Frogger, saliency also falls on the traffic in the road section. We attribute this to sprite flicker in the Atari rendering: the frog and the cars in its lane flicker in sync, so the flickering lane reveals the frog's vertical position and thereby the executed move. This cue is a rendering artifact and not a physical interaction, but like the platforms it lies outside M_{t+1} and is used by the IDM without entering the supervisory target.

Table 21: Loss masking versus input masking on Atari games in which the environment moves the agent (mean $\pm$ std over 3 seeds). Loss masking achieves higher probe accuracy and near-expert return; input masking collapses on Frogger and is unstable on Frostbite.

	Probe accuracy ↑			Normalized return ↑		
Method	**Frostbite**	**Frogger**	**Mean**	**Frostbite**	**Frogger**	**Mean**
MaskLAM@GT (loss masking)	$\mathbf{0.6902}_{\pm 0.0111}$	$\mathbf{0.6148}_{\pm 0.0005}$	$\mathbf{0.6525}_{\pm 0.0058}$	$\mathbf{0.9070}_{\pm 0.0023}$	$\mathbf{0.9784}_{\pm 0.0011}$	$\mathbf{0.9427}_{\pm 0.0017}$
w/ input masking	$0.5706_{\pm 0.0067}$	$0.5505_{\pm 0.0021}$	$0.5605_{\pm 0.0044}$	$0.6274_{\pm 0.4786}$	$0.0384_{\pm 0.0110}$	$0.3329_{\pm 0.2448}$
Chance	0.056 ($\lvert\mathcal{A}\rvert$=18)	0.200 ($\lvert\mathcal{A}\rvert$=5)	—	—	—	—

T.3 Agent-Only versus Agent-and-Object Masks

MaskLAM restricts the supervisory target to agent pixels, so changes of objects the agent interacts with, such as a door swinging open or a peg sliding into a hole, lie outside the target. To test whether this discards action-relevant information, we select the most contact-rich DMW tasks, where such object changes are most pronounced, and compare agent-only masks with masks that additionally cover the manipulated object.

Setup. On four contact-rich DMW tasks (door-open, peg-insert-side, pick-place, push) we train MaskLAM@GT with M_t covering the agent and the manipulated object, using simulator object masks, in both the distractor-free and the distractor setting. Everything else follows MaskLAM@GT. If the agent-only target discarded relevant signal, the object-inclusive variant should improve substantially.

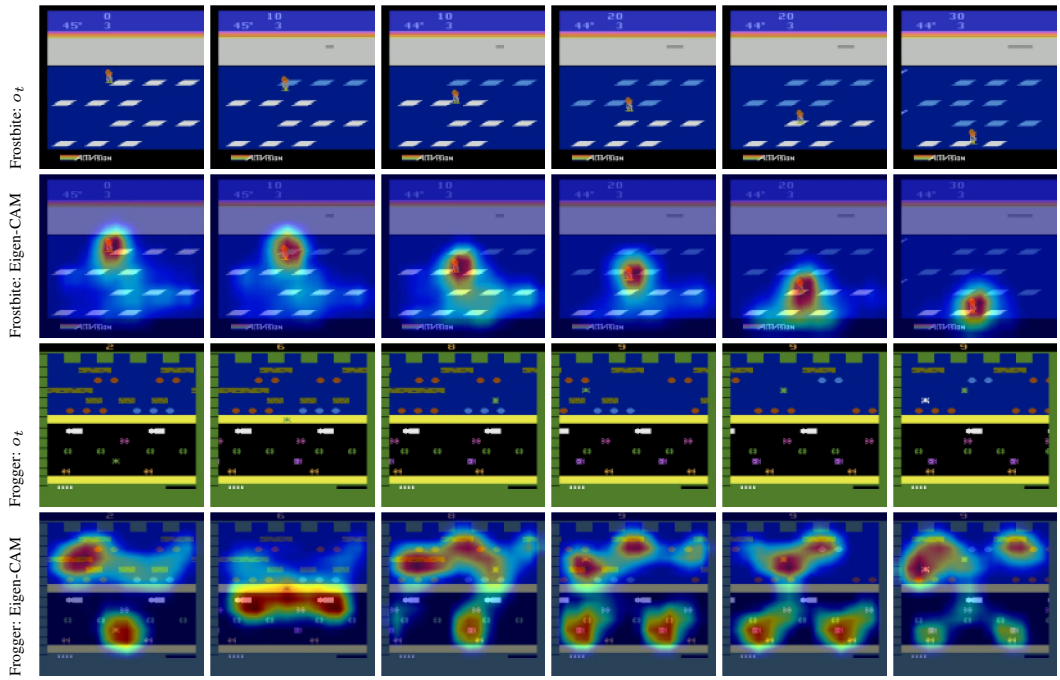


Figure 31: Eigen-CAM saliency of the MaskLAM IDM on Frostbite and Frogger. For each game, the top row shows the observation o_t and the bottom row the IDM saliency (warmer colors indicate higher influence on z_t). In Frostbite, saliency is centered on the agent and extends onto the ice floe it stands on. In Frogger, saliency falls on the logs and turtles in the river section, which lie outside the agent mask, and additionally on the traffic in the road section, whose sprites flicker in sync with the frog (Section T.2).

Results. Table 22 shows that the two targets agree on average: mean NMSE changes from 0.1462 to 0.1437 without distractors and from 0.1502 to 0.1484 with distractors, and no task improves consistently across both settings. Object motion caused by the agent is largely determined by the agent's own motion, and it remains visible to the IDM as input. Agent-only masks therefore suffice; when an action's effect is visible only on an object, the mask can be extended to include it.

Table 22: Action-probe NMSE (↓) with agent-only versus agent-and-object target masks on contact-rich DMW tasks (mean ± std over 3 seeds). Including the manipulated object in the target leaves NMSE unchanged on average in both settings.

Setting	Target mask	door-open	peg-insert-side	pick-place	push	Mean
Distractor-free	Agent (MaskLAM@GT)	$0.0763_{\pm 0.0017}$	$0.2417_{\pm 0.0079}$	$0.1290_{\pm 0.0093}$	$0.1376_{\pm 0.0047}$	$0.1462_{\pm 0.0059}$
	Agent + object	$0.0685_{\pm 0.0031}$	$0.2432_{\pm 0.0075}$	$0.1299_{\pm 0.0091}$	$0.1332_{\pm 0.0075}$	$0.1437_{\pm 0.0068}$
Distractor	Agent (MaskLAM@GT)	$0.0738_{\pm 0.0077}$	$0.2538_{\pm 0.0068}$	$0.1291_{\pm 0.0005}$	$0.1441_{\pm 0.0080}$	$0.1502_{\pm 0.0057}$
	Agent + object	$0.0771_{\pm 0.0049}$	$0.2454_{\pm 0.0048}$	$0.1233_{\pm 0.0045}$	$0.1480_{\pm 0.0053}$	$0.1484_{\pm 0.0049}$

U Failure Modes of LAOF and LAPO-Slots

On our distractor benchmarks, LAOF and LAPO-Slots do not reproduce the gains reported in their original papers: both stay close to or behind LAPO on DCS, LAOF falls behind LAPO on DMW, and only LAPO-Slots improves over LAPO on DMW (Table 1). To explain this gap, we trace each method back to its mask-based prior and identify a distinct failure mode in each case. Both could in principle be alleviated by replacing the underlying component, the optical flow model for LAOF and the object-centric encoder for LAPO-Slots, with a stronger alternative; the conclusions here concern the priors as currently instantiated, not the methods in their idealized form.

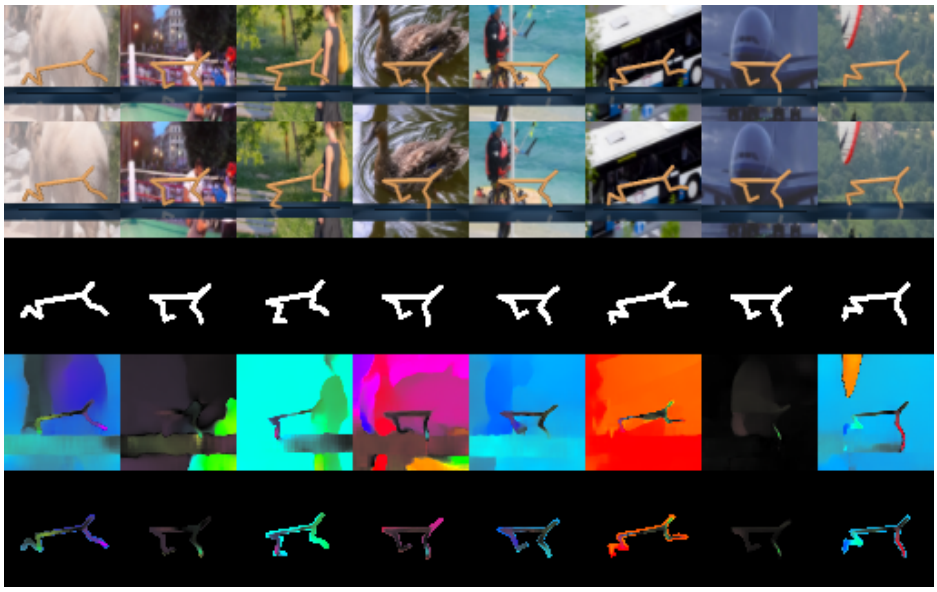


Figure 32: LAOF flow target construction on DCS cheetah-run (distractor). Eight transitions sampled from the dataset. Rows top-to-bottom: current observation o_t; next observation o_{t+1}; supervisory mask M_{t+1}; raw RAFT optical flow between o_t and o_{t+1}; masked flow $M_{t+1} \odot \text{RAFT}(o_t, o_{t+1})$, the target for LAOF's auxiliary flow decoder loss. Distractor motion in the dynamic background produces large flow magnitudes that bleed into the cheetah mask along its boundary, so the masked target inside the mask is dominated by exogenous motion instead of agent motion. The flow decoder is consequently asked to predict distractor flow conditioned on z_t, which pushes z_t to encode it.

LAOF: distractor optical flow bleeds into the agent mask. LAOF supervises an auxiliary flow decoder on RAFT optical flow restricted to the agent mask, so that only agent-driven motion shapes the latent action z_t. Figure 32 shows the construction on DCS cheetah-run with distractors. Distractor regions adjacent to the cheetah produce large RAFT flow magnitudes that, after multiplying by M_{t+1}, contaminate the target inside the mask, so the bottom-row flow the decoder is asked to predict is mostly exogenous motion. The decoder must therefore explain that motion conditioned on z_t, which forces z_t to carry distractor information just as in unmasked LAPO. Figure 33 shows the same construction on DMW push with distractors. The bleed covers a smaller area, but not because the mask is sharper: at DMW's higher rendering resolution the agent simply occupies more pixels, so the contaminated boundary region is a smaller fraction of the masked area. The distractor flow magnitude remains high relative to the slow robot arm, so wherever the bleed does occur it still dominates the local target.

LAPO-Slots: agent identity is not stable across slot indices. The full LAPO-Slots pipeline is described in Section D; here we focus on the three stages relevant to the failure mode. Stage 1 (VideoSAUR) decomposes each frame into K object-centric slots; Stage 2 selects the single most action-predictive slot index per task by linearly probing each slot's features against ground-truth actions; Stage 3 trains the LAM on the features of that one selected slot. The pipeline therefore depends on the agent occupying the same slot index across all samples in the dataset, since the Stage 2 selection is a single fixed index applied uniformly at Stage 3 training time: whenever VideoSAUR routes the agent to a different slot on a sample, the LAM input on that sample contains no agent. We use $K{=}4$ slots on DCS and $K{=}3$ slots on DMW, following the original protocol [23].

Figure 34 shows a representative *stable* run on DCS cheetah-run (distractor): slot 2 attends to the cheetah across all eight sampled transitions while the remaining slots absorb background structure, so a Stage 2 selection of slot 2 retains the agent on every sample. Figure 35 shows the failure mode on the same task: cheetah identity is not bound to a single slot index, but is split across slot 3 and slot 4 across the sampled batch, so no fixed slot index covers the agent on every sample. Figures 36 and 37 show the same pattern on DMW push (distractor): a stable run where slot 2 carries the robot

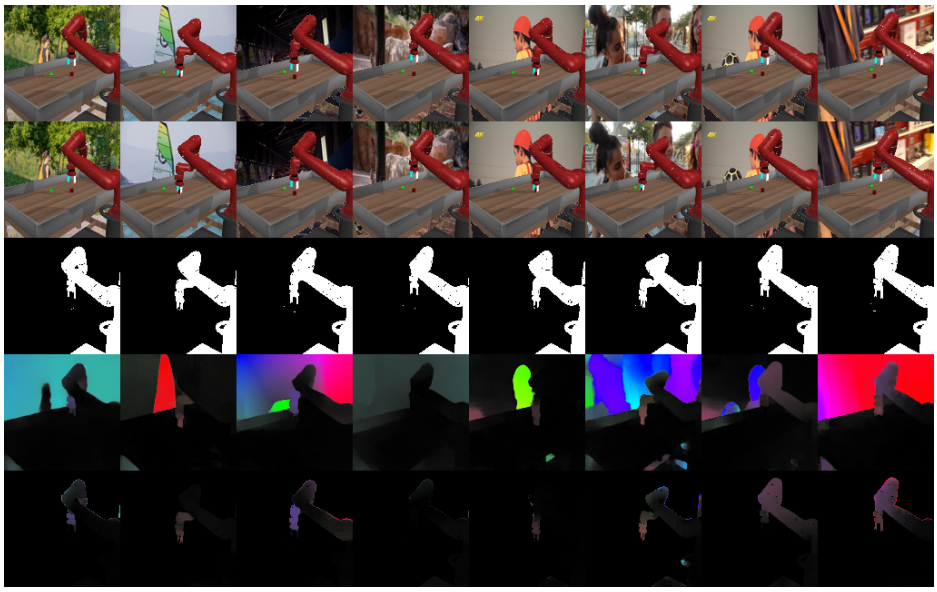


Figure 33: LAOF flow target construction on DMW push (distractor). Layout matches Figure 32. The contaminated boundary region covers a smaller fraction of the masked area than on DCS, not because the mask is sharper but because the agent occupies more pixels at DMW's higher rendering resolution. The distractor flow magnitude is still high relative to the slow robot arm, so where bleed does occur it still dominates the local target.

arm on every sample, and a failed run where arm identity is distributed across slots 1, 2, and 3 across the sampled batch. Because Stage 3 commits to a single slot index, this instability translates directly into LAM input dropout on the samples where VideoSAUR routes the agent elsewhere.

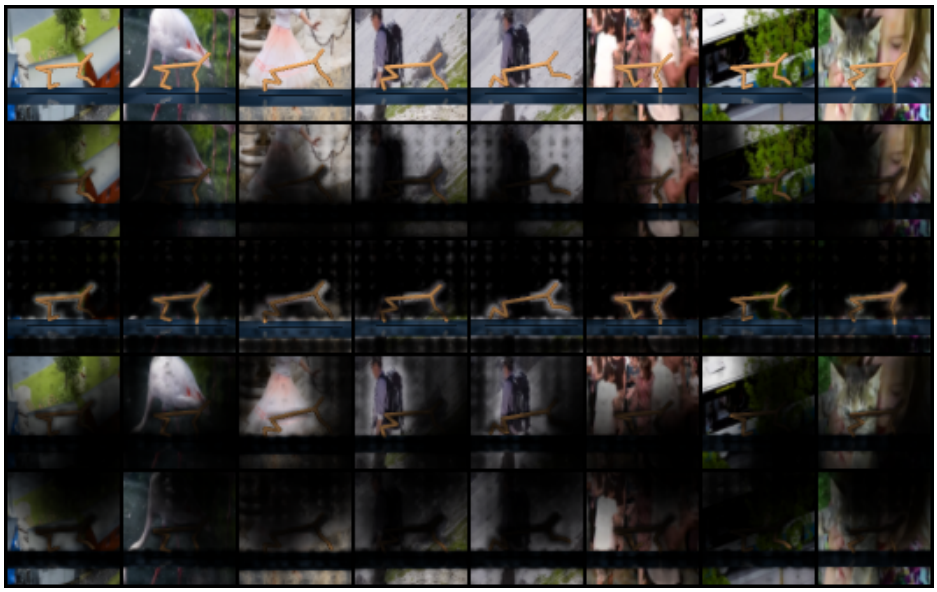


Figure 34: LAPO-Slots VideoSAUR slot decomposition on DCS cheetah-run (distractor), stable run. Eight transitions sampled from the dataset. Top row: input observation o_t. Rows 2–5: per-slot masked observations for slots 1–4. Slot 2 (row 3) consistently attends to the cheetah on every sampled frame; the remaining slots absorb distractor background structure. A Stage 2 selection of slot 2 therefore retains the agent on every dataset sample and the downstream LAM input is well defined.



Figure 35: LAPO-Slots VideoSAUR slot decomposition on DCS cheetah-run (distractor), failed run. Layout matches Figure 34. Cheetah identity is split between slots 3 and 4 (rows 4 and 5) across the sampled batch, so no single slot index covers the agent on every sample. Because Stage 3 trains the LAM on the features of one fixed slot, the agent is dropped from the LAM input on the samples where VideoSAUR assigns it to the other slot.

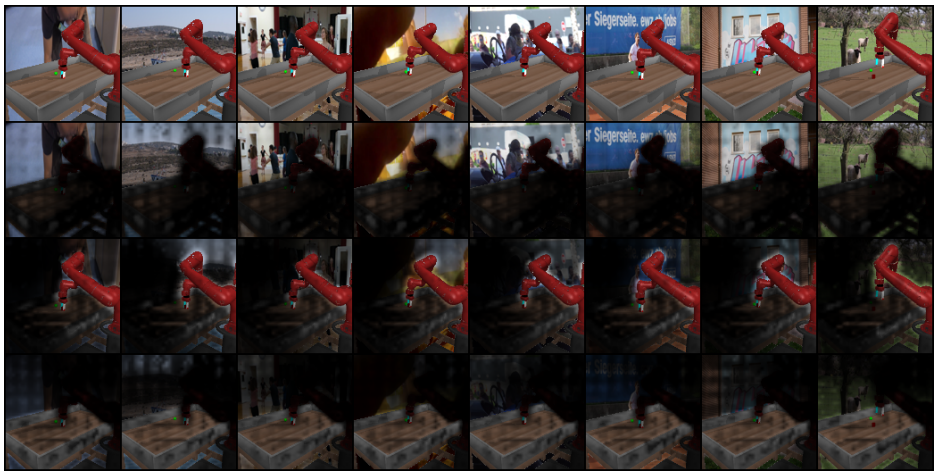


Figure 36: LAPO-Slots VideoSAUR slot decomposition on DMW push (distractor), stable run. Eight transitions sampled from the dataset. Top row: input observation o_t. Rows 2–4: per-slot masked observations for slots 1–3. Slot 2 (row 3) consistently attends to the robot arm on every sampled frame; slots 1 and 3 absorb table and distractor regions.



Figure 37: LAPO-Slots VideoSAUR slot decomposition on DMW push (distractor), failed run. Layout matches Figure 36. Robot-arm identity is distributed across slots 1, 2, and 3 across the sampled batch, so any fixed Stage 2 slot selection loses the arm on a subset of samples and the downstream LAM is trained on intermittently empty input.