LATENT ACTION POLICIES FROM SUBOPTIMAL DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent Action Models (LAMs) learn control representations from action-free video, but existing pipelines typically train latent policies with behavioral cloning, without an explicit policy improvement objective. Although offline RL offers a natural approach to policy improvement, standard methods do not transfer directly to learned latent action spaces. We find that the structure of learned latent actions makes naive offline RL unreliable: directly maximizing value over latent actions or fitting a unimodal policy can produce latent actions that cannot be realized by physical execution, while the latent representation can merge physical controls that receive different rewards. To understand when latent-space optimization remains valid, we derive conditions under which policy improvement transfers to decoded execution and under which latent and physical optima align. Building on these insights, we propose LAPIS for goal-conditioned latent policy improvement, grounding value-based policy improvement in observed state–latent transitions and using a conditional flow policy to capture their multimodal structure. Across visual manipulation and navigation tasks, LAPIS consistently improves over latent-space baselines, while retaining strong performance on cube tasks even with only 1% action labels.

1 INTRODUCTION

Latent Action Models (LAMs) have emerged as a promising paradigm for learning control representations from action-free video by inferring latent actions that explain observed state transitions (Ye et al., 2025; Chen et al., 2025; 2024). Because latent-action learning captures transition and outcome structure rather than task optimality, it can naturally exploit large collections of diverse and suboptimal video, including failed trajectories that would be poor demonstrations but remain informative for learning how the environment changes. Yet existing LAM pipelines typically train the resulting latent policies with behavioral cloning (BC) (Schmidt & Jiang, 2024; Ye et al., 2025; Chen et al., 2024; Bruce et al., 2024), without explicitly optimizing latent policies for improved task return. This creates a gap between the broad data that LAMs can learn from and the policies that current pipelines can extract from it.

Offline reinforcement learning (RL) offers a natural way to improve beyond the observed behavior (Levine et al., 2020; Kostrikov et al., 2022; Fujimoto et al., 2019; Ki et al., 2025; Lu et al., 2025), but applying standard offline RL directly in a learned latent action space is not straightforward. We identify three properties that distinguish learned latent actions from physical actions. The first two concern *realizability*—whether the transition outcome represented by a latent action can be physically produced from the current state: (i) *state-conditioned realizability*, whereby a latent action realizable at one state can be unrealizable at another; and (ii) *non-convex realizable set*, whereby interpolating between realizable latent actions can yield unrealizable ones. The third, (iii) *outcome-based abstraction*, arises because distinct physical controls that reach the same outcome can map to the same latent action.

These properties expose two distinct failure modes of naive offline RL in latent action spaces. The first is an optimization–execution mismatch. Because the realizable latent set is state-dependent and potentially non-convex, standard offline RL can move outside this set by directly maximizing latent-action value or through unimodal policy fitting, producing latent actions that cannot be faithfully realized by physical execution. This raises the question: *when does latent-space policy improvement transfer to decoded physical execution?* The second is an objective mismatch. Because outcome-based abstraction can merge physical controls with different rewards, the optimal latent policy need

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

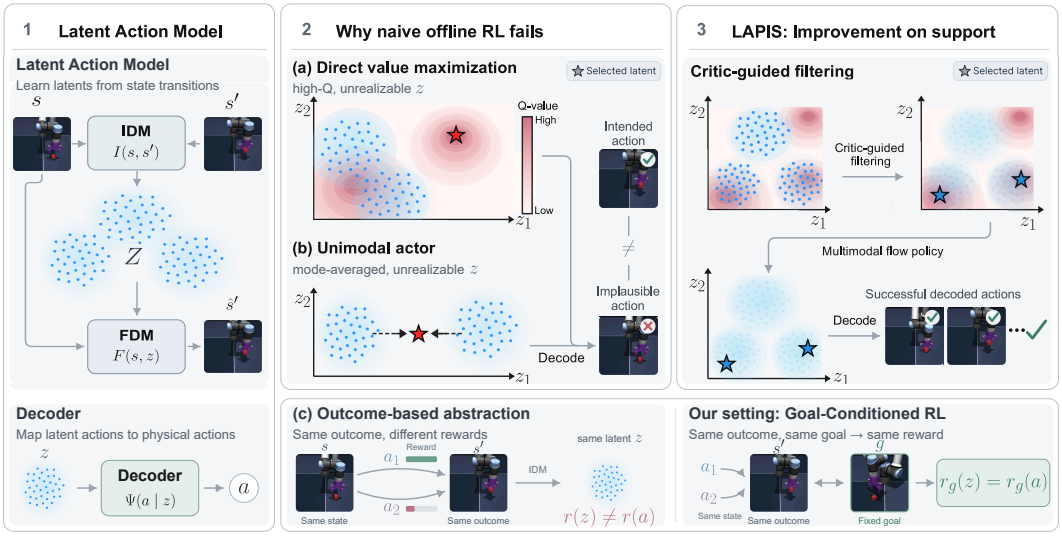

Figure 1: **Overview of LAPIS.** Left: a Latent Action Model learns latent actions from action-free trajectories and a decoder from limited action labels. Middle: naive offline RL can produce unrealizable latent actions through value maximization or unimodal policy fitting, while outcome-based abstraction can merge reward-distinct controls. Right: LAPIS performs support-aware policy improvement with critic-guided filtering and a multimodal flow policy, instantiated with an outcome-compatible goal-conditioned objective.

not correspond to the optimal physical policy. This raises a separate question: *when do the latent and physical policy optima align?*

To answer these questions, we analyze paired physical- and latent-action batch MDPs constructed from the same observed transitions. For the first question, we show that batch-constrained latent policy improvement transfers to physical execution under deterministic dynamics and supported outcome identifiability. For the second, we show that latent and physical optima align under an outcome-determined objective within the decoder-covered, batch-constrained physical policy class. These results characterize when latent-space policy improvement remains valid and provide the principles behind our method.

Building on these insights, we introduce LAPIS (Latent Action Policy Improvement on Support), a latent policy improvement framework motivated by these failure modes. Rather than directly optimizing a latent actor toward arbitrary high-value actions, LAPIS grounds value-based policy improvement in observed state–latent transitions and uses a conditional flow policy to capture their potentially multimodal structure. LAPIS is instantiated with a goal-conditioned objective based on future states, which is naturally compatible with the outcome semantics captured by latent actions. At inference, classifier-free guidance provides a simple way to adjust the emphasis between behavior and high-value policy branches. While standard offline RL methods struggle when applied directly in the learned latent space, LAPIS achieves strong performance across OGBench visual manipulation and navigation tasks under both full and scarce action supervision, outperforming prior semi-supervised offline RL baselines with only 5% or 1% action labels through action-free LAM pretraining. Further experiments examine the contributions of these design choices.

2 Related Work

Latent action models from action-free video A growing body of work learns controllable representations from action-free video by inferring latent actions that explain observed transitions (Edwards et al., 2019; Rybkin et al., 2019; Baker et al., 2022; Bruce et al., 2024; Google DeepMind, 2024). Recent approaches learn such representations using discrete or continuous latent codes and extend latent action modeling to richer video and vision-language-action settings (Schmidt & Jiang, 2024; Ye et al., 2025; Chen et al., 2024; Gao et al., 2025; Zhang et al., 2025; Chen et al., 2025). Their

control pipelines predominantly train latent policies by behavioral cloning and use a small labeled set to decode latent actions into physical controls. LAPO (Schmidt & Jiang, 2024) additionally incorporates reinforcement learning by applying online PPO to fine-tune the physical action policy after latent-action pretraining. In contrast, we study policy improvement directly in the learned latent action space using offline data, with scarce action labels used for physical action decoding.

Action-free and semi-supervised reinforcement learning. A related line of work studies learning control from trajectories with missing or limited action annotations, using inferred actions, latent representations, or other forms of auxiliary supervision to exploit action-free experience (Torabi et al., 2018; Baker et al., 2022; Ghosh et al., 2023; Park et al., 2023; Zheng et al., 2023). Most closely related is LAQ (Chang et al., 2022), which performs Q-learning over latent actions from state-only data but relies on a hand-specified controller for execution. In contrast, we learn a latent action space from action-free data together with a physical-action decoder from scarce action labels, and study offline policy improvement directly in this learned latent action space. This setting makes the relationship between latent-space optimization and decoded physical execution central, motivating our analysis of when latent policy improvement transfers to physical control. Further related work is discussed in Appendix A.

3 PRELIMINARIES

Goal-conditioned reinforcement learning We model the environment as an MDP $\mathcal{M} = (S, A, p, r, \gamma)$, with dynamics $p(s' \mid s, a)$, bounded reward r, and initial-state distribution μ_0. We write $(s, a, s') = (s_t, a_t, s_{t+1})$ for a transition. A policy $\pi(a \mid s)$ induces the usual state-action and state value functions $Q^\pi_\mathcal{M}$ and $V^\pi_\mathcal{M}$, with expected return $J_\mathcal{M}(\pi) = \mathbb{E}_{s_0\sim\mu_0}[V^\pi_\mathcal{M}(s_0)]$. We refer to $\mathcal{M}$ as the ground MDP to distinguish it from the batch MDPs introduced in Section 5. In the goal-conditioned setting, the policy additionally conditions on a goal g, $\pi(a \mid s, g)$. Contrastive RL (Eysenbach et al., 2022) casts goal-conditioned control as density-ratio estimation, bypassing explicit modeling of transition dynamics or reward functions. The goal-reaching reward is $(1-\gamma)\, p(s'{=}g \mid s, a)$, and the corresponding goal-conditioned value equals the discounted future-state occupancy:

$$Q^\pi_g(s, a) = p^\pi(s_{t+}{=}g \mid s, a) = (1-\gamma)\sum_{k=1}^{\infty} \gamma^{k-1} p^\pi(s_{t+k}{=}g \mid s_t = s, a_t = a), \tag{1}$$

where the right-hand side denotes the discounted occupancy of future states visited from (s, a) under π. To illustrate the contrastive principle, consider a critic parameterized as a bilinear score $f_\phi(s, a, g) = \phi(s, a)^\top \psi(g)/\sqrt{d}$ and trained by noise-contrastive binary classification, with a positive goal g^+ sampled from the future-state occupancy and a negative goal $g^- \sim p_B$ sampled from the dataset state distribution,

$$\max_{\phi,\psi}\ \mathbb{E}_{(s,a),\, g^+,\, g^-}\big[\log \sigma(f_\phi(s, a, g^+)) + \log(1 - \sigma(f_\phi(s, a, g^-)))\big]. \tag{2}$$

For the ideal binary classification problem, the optimal score is the log density ratio, $f^* = \log Q^\pi_g - \log p_B(g)$ wherever the ratio is defined. LAPIS uses a TD-InfoNCE estimator, detailed in Appendix E.3. In the offline setting, directly maximizing this critic can drive the actor outside the support of the fixed behavior data, so offline CRL regularizes policy improvement with behavioral cloning,

$$\mathcal{L}_{\mathrm{CRL}}(\pi) = \mathbb{E}_{(s,a,g)\sim\mathcal{D}}\big[-f_\phi(s, \pi(s, g), g) + \alpha\, \|\pi(s, g) - a\|_2^2\big],$$

improving the actor toward high-value actions while keeping it close to the observed behavior.

Latent action models A Latent Action Model (LAM) learns a latent action space Z from an action-free dataset $\mathcal{B} = \{(s_i, s'_i)\}_{i=1}^N$ by training an inverse dynamics model I and a forward dynamics model F, with $z = I(s, s')$ and $\hat{s}' = F(s, z)$ with a bottlenecked latent representation,

$$\min_{I,F}\ \mathbb{E}_{(s,s')\sim\mathcal{B}}\big[\|s' - F(s, I(s, s'))\|_2^2\big]. \tag{3}$$

Relabeling each transition by its latent action $z_i = I(s_i, s'_i)$ yields the latent dataset $\mathcal{B}^z = \{(s_i, z_i, s'_i)\}_{i=1}^N$. A separate small labeled dataset $\mathcal{B}^{\mathrm{dec}} = \{(s_i, a_i, s'_i)\}_{i=1}^{N_{\mathrm{dec}}}$ with $N_{\mathrm{dec}} \ll N$ supplies the action supervision for the decoder.

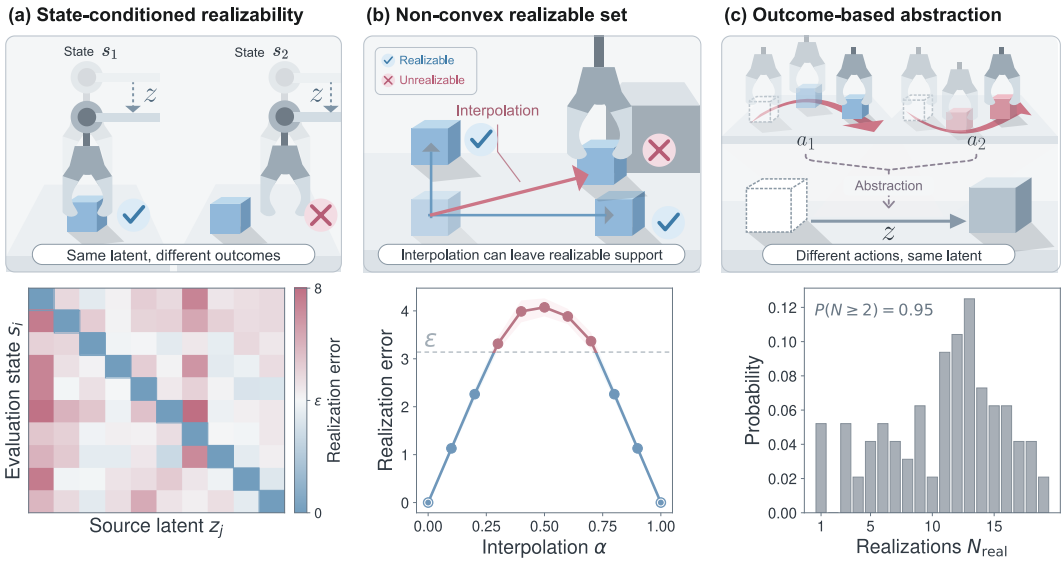

Figure 2: **Structural properties of learned latent action spaces.** *Top row:* concept schematics for **(a)** *state-conditioned realizability* (the same latent can be realizable at s_1 but not s_2), **(b)** the *non-convex realizable set* (interpolating two realizable latents can pass through unrealizable ones), and **(c)** *outcome-based abstraction* (distinct physical controls reaching the same successor share one latent). *Bottom row:* empirical diagnostics for the experimental LAM on `visual-cube-single-play-v0`; the diagnostic protocol is in Appendix B.

The deployed system composes a *latent policy* with a *decoder*. The latent policy $\pi_z(z \mid s)$ selects a latent transition at the current state, and the state-marginal decoder $\Psi(a \mid z)$ maps the chosen latent to a physical action, trained on $\mathcal{B}^{\text{dec}}$ by $\max_\Psi \mathbb{E}_{\mathcal{B}^{\text{dec}}}[\log \Psi(a \mid z)]$ and frozen thereafter. Deployment samples $z \sim \pi_z(\cdot \mid s)$ then $a \sim \Psi(\cdot \mid z)$, giving the deployed policy $\bar{\pi} = \Psi \circ \pi_z$. Section 5 introduces an ideal state-conditioned batch decoder $\Psi^*(a \mid s, z)$ only for the transfer analysis.

4 Latent Action Structure and Failures

A latent policy selects a latent action representing a desired transition outcome, which must ultimately be realized through physical control at deployment. Section 4.1 characterizes the realizability and abstraction structure of learned latent actions. Section 4.2 then shows how realizability can break the connection between latent-space optimization and physical execution, while outcome abstraction can misalign latent and physical objectives.

4.1 Structural properties of learned latent action spaces

To formalize whether a target latent can be physically realized from the current state, we define its realization error as the infimum of its distance to the latents of reachable outcomes.

Definition 1 (Latent realizability)**.** *Let* $\mathcal{R}(s) := \bigcup_{a \in A} \operatorname{supp} p(\cdot \mid s, a)$ *be the set of successor states reachable from* s*. The* latent realization error *of a target latent* z *at state* s *is*

$$\delta(s, z) = \inf_{s' \in \mathcal{R}(s)} \left\| I(s, s') - z \right\|_2,$$

and z *is* ϵ-realizable *at* s *if* $\delta(s, z) \leq \epsilon$*, defining the realizable set* $\mathcal{Z}^\epsilon_{\text{real}}(s) = \{z : \delta(s, z) \leq \epsilon\}$.

The observed set $\mathcal{B}^z$ contains only a subset of realizable state–latent pairs; absence from $\mathcal{B}^z$ does not imply unrealizability. Figure 2 illustrates the properties discussed below.

State-conditioned realizability Whether a latent is realizable depends on the current state: the same z can be realizable from s_1 but not s_2, so realizability is a property of the pair (s, z), not of z alone ($\mathcal{Z}^\epsilon_{\text{real}}(s_1) \neq \mathcal{Z}^\epsilon_{\text{real}}(s_2)$). In Figure 2(a), we take latents with known realizations at their

source states and search for physical realizations of the same latents from other states. Latents have near-zero realization error at their source states, while 73% of transfers to other states exceed the diagnostic threshold under finite search.

Non-convex realizable set Even at a fixed state, the space between two realizable latents need not be realizable. For $z_1, z_2 \in \mathcal{Z}^{\epsilon}_{\text{real}}(s)$, an interpolation $z_\alpha = (1-\alpha)z_1 + \alpha z_2$ need not be realizable, so $\mathcal{Z}^{\epsilon}_{\text{real}}(s)$ need not be convex. In Figure 2(b), we select pairs of realizable endpoint latents at the same state and search for physical realizations along the interpolation between them. The endpoints admit known physical realizations, while interior errors exceed the diagnostic threshold.

Outcome-based abstraction The encoder $z = I(s, s')$ labels the transition outcome rather than the physical command, so distinct actions reaching the same successor from s share a latent. In Figure 2(c), we search for substantially different action chunks that reach each target outcome within a fixed tolerance; 95% of sampled targets admit at least two such realizations.

4.2 Why the properties break naive offline RL

These properties become consequential when offline RL is applied in the learned latent space, giving rise to two distinct failure modes: an optimization–execution mismatch and an objective mismatch.

Optimization–execution mismatch *State-conditioned realizability* and a *non-convex realizable set* imply that physically realizable latent actions depend on the current state, while interpolations between realizable latents can be unrealizable. As a result, directly optimizing the actor to maximize learned value can move it toward unrealizable latents despite behavior regularization, while a unimodal actor can place probability between separated realizable modes and generate unrealizable latents. Thus, even when the critic favors a high-value latent action z, if z is unrealizable, decoding and physical execution can reach a different successor than the outcome represented by z. Consequently, latent-space policy improvement need not translate to improvement after decoding,

$$\mathcal{J}_{\text{latent}}(\pi'_z) \geq \mathcal{J}_{\text{latent}}(\pi_z) \quad \not\Longrightarrow \quad J_{\mathcal{M}}(\Psi \circ \pi'_z) \geq J_{\mathcal{M}}(\Psi \circ \pi_z).$$

Here $\mathcal{J}_{\text{latent}}$ denotes the learned latent-space optimization objective.

Objective mismatch *Outcome-based abstraction* can erase reward-relevant distinctions between physical controls. When distinct controls $a_1 \neq a_2$ reach the same successor s', they share the latent action $z = I(s, s')$, even if an action-dependent reward assigns $r(s, a_1, s') \neq r(s, a_2, s')$. Because the latent policy can select only z, it cannot distinguish between these reward-distinct physical controls. Consequently, a policy that is optimal in the latent-action MDP need not decode to an optimal physical-action policy,

$$\pi^*_z \in \arg\max_{\pi_z \in \Pi_{\mathcal{B}^z}} J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) \quad \not\Longrightarrow \quad (\Psi^* \circ \pi^*_z) \in \arg\max_{\pi \in \Pi_{\mathcal{B}^a}} J_{\mathcal{M}}(\pi).$$

Here, $\Pi_{\mathcal{B}^a}$ and $\Pi_{\mathcal{B}^z}$ denote the physical and latent policy classes restricted to observed state–action and state–latent pairs, respectively, and $\Psi^\star$ is the batch-exact decoder defined in Section 5.

These failure modes raise two questions: when does latent-space policy improvement transfer to physical execution, and when do latent and physical policy optima align? Section 5 establishes conditions for both, which motivate the design of LAPIS in Section 6.

5 When Does Latent Policy Improvement Work?

To answer these questions, we construct paired batch MDPs from the same observed transitions under physical and latent action representations,

$$\mathcal{M}_{\mathcal{B}^a} = (S \cup \{s_{\text{abs}}\}, A, p_{\mathcal{B}^a}, r_{\mathcal{B}^a}, \gamma), \qquad \mathcal{M}_{\mathcal{B}^z} = (S \cup \{s_{\text{abs}}\}, Z, p_{\mathcal{B}^z}, r_{\mathcal{B}^z}, \gamma).$$

The two MDPs are constructed from $\mathcal{B}^a = \{(s_i, a_i, s'_i)\}_{i=1}^N$ and its latent relabeling $\mathcal{B}^z = \{(s_i, z_i, s'_i)\}_{i=1}^N$, with $z_i = I(s_i, s'_i)$, while unobserved pairs transition to the zero-reward absorbing state s_{abs}. Because realizability is state-conditioned, support in $\mathcal{M}_{\mathcal{B}^z}$ is over observed pairs (s, z), not z globally. We further assume *supported outcome identifiability*: each supported (s, z) corresponds

to a unique successor $F_{\mathcal{B}}(s, z)$, so that a supported latent action has a well-defined transition outcome at each state. This supported successor map is distinct from the learned forward predictor F. For the analysis, we deploy a latent policy through the *batch-exact state-conditioned decoder* $\Psi^*(a \mid s, z)$, which selects physical actions observed at s to produce the successor represented by z in the batch. The support domain, initial-state and terminal conventions, batch rewards, and decoder construction are specified in Appendix C.1. All exact results below use this common construction. Appendix A.4 discusses how our analysis extends classical batch-constrained RL and MDP abstraction to learned latent actions.

5.1 When does latent policy improvement transfer?

We first ask when an improvement measured in the latent batch MDP is preserved by decoded physical execution. To avoid selecting unrealizable latent actions, we restrict the latent policy to state–latent transitions observed in the batch.

Definition 2 (Batch-constrained policy)**.** *A policy is batch-constrained if its trajectory uses only observed state–action pairs in $\mathcal{B}^a$. A latent batch-constrained policy is defined analogously over $\mathcal{B}^z$.*

Every observed state–latent pair $(s, z) \in \mathcal{B}^z$ has a physical realization by construction, and the batch-exact decoder Ψ^* selects physical actions observed at s that realize this outcome. A batch-constrained latent policy selects only latent actions that are physically realizable at the current state.

Support alone, however, does not ensure exact transfer. Under stochastic dynamics, the same decoded action can reach a different successor from the one used to evaluate z in the latent batch MDP, breaking the correspondence between latent evaluation and physical execution. We therefore assume deterministic ground dynamics, which leads to Theorem 1.

Assumption 1 (Deterministic ground dynamics)**.** *The successor is a deterministic function of the current state and physical action, $s' = T(s, a)$.*

Theorem 1 (On-support policy improvement transfer)**.** *Under Assumption 1, let π_z be a batch-constrained latent policy. Then the latent return equals the decoded physical return, $J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) = J_{\mathcal{M}}(\Psi^* \circ \pi_z)$, and consequently, for any two such latent policies π_z and π'_z,*

$$J_{\mathcal{M}_{\mathcal{B}^z}}(\pi'_z) \geq J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) \quad \Longrightarrow \quad J_{\mathcal{M}}(\Psi^* \circ \pi'_z) \geq J_{\mathcal{M}}(\Psi^* \circ \pi_z).$$

Proof. See Appendix C.2. □

Thus, return improvement between batch-constrained latent policies transfers to their batch-exact decoded policies. Appendix D.3 extends this exact result to stochastic dynamics, bounding the transfer gap by the mismatch between the latent batch transition and the transition induced by decoded execution.

5.2 When do latent and physical optima align?

Even when latent policy improvement transfers to decoded execution as in Theorem 1, this alone does not guarantee alignment between the optimal latent and physical-action policies. Under a generic action-dependent reward, *outcome-based abstraction* can collapse reward-distinct physical controls into the same latent action, removing potentially necessary distinctions for physical optimality.

Assumption 2 (Outcome-determined reward)**.** *The reward depends on a transition only through the current and next states, $r(s, a, s') = r(s, s')$.*

Under Assumption 2, physical actions reaching the same successor from the same state receive the same immediate reward, removing the reward ambiguity introduced by outcome-based abstraction. We therefore obtain the following alignment result.

Theorem 2 (Optimal-policy alignment)**.** *Under Assumptions 1–2, for the same initial distribution μ_0,*

$$\pi_z^* \in \arg\max_{\pi_z \in \Pi_{\mathcal{B}^z}} J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) \quad \Longrightarrow \quad (\Psi^* \circ \pi_z^*) \in \arg\max_{\pi \in \Pi_{\mathcal{B}^a}} J_{\mathcal{M}}(\pi).$$

Proof. See Appendix C.3. □

Algorithm 1 LAPIS training and deployment

Require: transition batch $\mathcal{B}$, labeled set $\mathcal{B}_{\text{dec}}$, guidance scale w
1: Train IDM $I(s, s')$, FDM $F(s, z)$, and decoder $\Psi(a \mid z)$; form $\mathcal{B}^z$
2: **repeat**
3: Score with $f_\phi(s, z, g)$ and gate with $G(s, z, g)$ in Eq. (4)
4: Fit $v_\theta(x_t, t; s, g, o)$ by minimizing $\mathcal{L}_{\text{actor}}$ in Eq. (5)
5: **until** convergence
6: **Deploy:** generate z with v_w^{CFG} and execute $a \sim \Psi(\cdot \mid z)$

6 LAPIS: Policy Improvement on Support

Section 5 motivates batch-constrained latent policy improvement over supported state–latent transitions. LAPIS follows this principle by using critic-guided filtering to select high-value transitions from the observed state–latent pairs and a conditional flow policy to capture their potentially multimodal distribution. We instantiate LAPIS in goal-conditioned RL, using future states as goals to obtain a reward-label-free objective compatible with outcome-based latent actions. Appendix D relates the practical algorithm to the supported-improvement analysis and explains how the learned components instantiate its principles.

Critic-guided transition filtering LAPIS fits a goal-conditioned contrastive critic $f_\phi(s, z, g)$ for the behavior latent policy π_β^z and uses it to score only the state–latent transitions observed in $\mathcal{B}^z$. We retain high-value transitions using a learned state–goal threshold,

$$G(s, z, g) := \mathbf{1}\big[f_\phi(s, z, g) > \hat{V}(s, g)\big], \tag{4}$$

where $\hat{V}(s, g)$ is learned by expectile regression in the critic's score space (Appendix E.3). We show in Appendix C.4 that, for the optimal density-ratio score $f^\star$, critic-guided filtering corresponds to a supported reweighting that preserves support and improves return.

Multimodal policy extraction To capture the potentially multimodal distribution of gated high-value transitions, LAPIS uses conditional flow matching. For $x_0 \sim \mathcal{N}(0, I)$ and target latent z, we use the linear path $x_t = (1-t)x_0 + tz$ with target velocity $z - x_0$. The velocity model $v_\theta(x_t, t; s, g, o)$ is trained with two branches: $o{=}0$ models all observed transitions, while $o{=}1$ models only the high-value transitions selected by $G(s, z, g)$,

$$\mathcal{L}_{\text{actor}} = \mathbb{E}\big[G(s, z, g)\, \|v_\theta(x_t, t; s, g, 1) - (z - x_0)\|_2^2 + \lambda\, \|v_\theta(x_t, t; s, g, 0) - (z - x_0)\|_2^2\big]. \tag{5}$$

At inference, classifier-free guidance combines the behavior and high-value branches,

$$v_w^{\text{CFG}} = (1 - w)\, v_\theta(\cdot; 0) + w\, v_\theta(\cdot; 1),$$

where $w{=}0$ recovers the behavior branch, $w{=}1$ the high-value branch, and $w{>}1$ further emphasizes the high-value branch.

7 Experiments

Our experiments address five questions: **(Q1)** How does LAPIS perform in learned latent action spaces, and does standard goal-conditioned offline RL remain effective? **(Q2)** How does LAPIS compare with goal-conditioned baselines adapted to limited action supervision? **(Q3)** How important is it to keep latent policy improvement on supported state–latent transitions? **(Q4)** Which components of LAPIS drive its gains? **(Q5)** How does decoder conditioning affect performance across action-label budgets?

Benchmarks We evaluate on the OGBench visual suite (Park et al., 2025), covering `cube` and `scene` manipulation as well as `antmaze` navigation, and additionally on visual D4RL Franka Kitchen (Fu et al., 2020; Gupta et al., 2020) for multi-subtask manipulation. All tasks are evaluated in a goal-conditioned offline RL setting with 64×64 RGB observations; example observations from OGBench and Kitchen are shown in Figures 5 and 6, respectively. We use $x\%$ *labels* to denote a setting where $x\%$ of the trajectories have action annotations while the remaining $100{-}x\%$ are available as action-free data, and evaluate at 100%, 5%, and 1% labels.

Table 1: Success rate (%) on the OGBench visual suite. Gray columns report real-action policies trained with 100% action labels, and the two 5%-label groups report results without and with LAM.

	100% labels (oracle)					5% labels (w/o LAM)				5% labels (w/ LAM)			
Task	GCBC	F-GCBC	GCIQL	CRL	LAPIS	F-GCBC	GCIQL	CRL	LAPIS	F-GCBC	GCIQL	CRL	LAPIS
v-antmaze-medium-navigate	11 ±2	8 ±1	11 ±1	**95** ±2	48 ±3	3 ±0	4 ±1	**8** ±1	2 ±1	8 ±1	7 ±2	11 ±2	**49** ±3
v-antmaze-large-navigate	4 ±0	1 ±0	4 ±1	**85** ±3	14 ±3	**1** ±0	0 ±0	0 ±1	0 ±0	1 ±0	2 ±1	3 ±1	**10** ±2
v-antmaze-giant-navigate	0 ±0	0 ±0	0 ±0	**47** ±1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
v-antmaze-teleport-navigate	5 ±1	7 ±1	6 ±1	**51** ±1	23 ±1	**2** ±1	1 ±1	1 ±1	**2** ±1	6 ±1	4 ±1	6 ±1	**21** ±3
v-cube-single-play	5 ±1	11 ±2	30 ±5	18 ±10	**45** ±4	2 ±1	**7** ±2	1 ±1	4 ±1	4 ±1	5 ±2	4 ±2	**62** ±5
v-cube-double-play	1 ±1	5 ±3	1 ±1	1 ±0	**30** ±6	0 ±0	0 ±0	0 ±0	0 ±0	4 ±1	1 ±1	1 ±0	**31** ±6
v-cube-triple-play	15 ±2	19 ±0	15 ±1	13 ±3	**22** ±1	0 ±0	0 ±0	0 ±0	**1** ±1	16 ±1	12 ±6	11 ±4	**20** ±0
v-cube-quadruple-play	8 ±1	11 ±3	7 ±1	4 ±1	**16** ±1	0 ±0	0 ±0	0 ±0	0 ±0	10 ±3	3 ±1	4 ±2	**14** ±3
v-scene-play	12 ±2	20 ±3	12 ±2	13 ±3	**45** ±1	2 ±1	1 ±1	3 ±1	**4** ±0	9 ±2	6 ±1	6 ±4	**29** ±4

Baselines and evaluation In Tables 1–3, magenta method names indicate the use of a LAM. We compare LAPIS against GCBC, flow-based GCBC (F-GCBC), goal-conditioned IQL (GCIQL; Kostrikov et al., 2022), and contrastive RL (CRL; Eysenbach et al., 2022) in Table 1. Table 3 extends the comparison to goal-conditioned baselines adapted to limited action supervision, adding SS-ORL (Zheng et al., 2023), VPT-RL (Baker et al., 2022), and HIQL (Park et al., 2023). For OGBench, we report success rates as mean ± std over four seeds, evaluating 20 episodes on each of five tasks per environment. All latent action models use a temporal gap of $H = 3$, with $z_t = I(s_t, s_{t+H})$. Detailed implementation settings for each algorithm are provided in Appendix E.

7.1 Goal-conditioned offline RL in learned latent action spaces (Q1)

Table 1 evaluates whether goal-conditioned policy improvement remains effective when physical actions are replaced by learned latent actions under scarce action supervision. We include a 100%-label oracle and 5%-label training without LAM pretraining as reference points. With LAM pretraining, LAPIS substantially improves over 5%-label training without LAM and matches or exceeds its fully supervised LAPIS counterpart on several tasks. Standard latent-space baselines remain weaker: CRL directly maximizes critic value with a BC-regularized actor, which can move away from observed latent modes toward unrealizable high-value regions, while GCIQL fits advantage-weighted latent actions, which can average separated high-value modes under a unimodal actor. LAPIS instead filters observed state–latent transitions and models their multimodal distribution with a flow policy.

Beyond OGBench: visual Franka Kitchen We further evaluate on visual D4RL Franka Kitchen, a multi-subtask manipulation benchmark. Table 2 reports the percentage of required subtasks completed on the `partial` and `mixed` target-task sets; details are provided in Appendix E.1. LAPIS also consistently outperforms GCIQL and CRL on both target-task sets with 100% and 5% action labels.

Table 2: Normalized score (%) on visual D4RL Kitchen with 100% and 5% action labels.

	100% labels (oracle)			5% labels (w/ LAM)		
Task	GCIQL	CRL	LAPIS	GCIQL	CRL	LAPIS
partial	21 ±9	28 ±4	**60** ±4	19 ±9	26 ±12	**44** ±6
mixed	22 ±3	31 ±9	**53** ±7	21 ±9	25 ±12	**41** ±5

7.2 Comparison under limited action supervision (Q2)

Table 3: Success rate (%) on OGBench `cube` tasks with 5% and 1% action labels. We compare LAPIS with goal-conditioned methods adapted to learning under limited physical-action supervision.

	5% labels					1% labels				
Task	SS-ORL	VPT-RL	GCIQL	HIQL	LAPIS	SS-ORL	VPT-RL	GCIQL	HIQL	LAPIS
v-cube-single-play	55 ±3	48 ±6	0 ±0	24 ±8	**62** ±5	0 ±0	0 ±0	0 ±0	2 ±4	**50** ±6
v-cube-double-play	11 ±6	17 ±13	0 ±0	5 ±2	**31** ±6	0 ±0	0 ±0	0 ±0	0 ±0	**20** ±8
v-cube-triple-play	9 ±5	15 ±9	0 ±0	13 ±4	**20** ±0	0 ±0	0 ±0	0 ±0	1 ±1	**17** ±1
v-cube-quadruple-play	0 ±0	2 ±3	0 ±0	6 ±1	**14** ±3	0 ±0	0 ±0	0 ±0	0 ±0	**10** ±3

We next compare LAPIS with goal-conditioned baselines adapted to learning under limited physical-action supervision. These adaptations use hindsight goal relabeling, with implementation details provided in Appendix E.4. On the four `cube` tasks, LAPIS achieves the strongest performance at both 5% and 1% action labels (Table 3). The gap becomes particularly pronounced at 1%, where the competing methods largely collapse while LAPIS retains non-trivial performance across all four tasks. These results show that LAPIS can effectively exploit action-free trajectories even when physical action supervision is extremely limited.

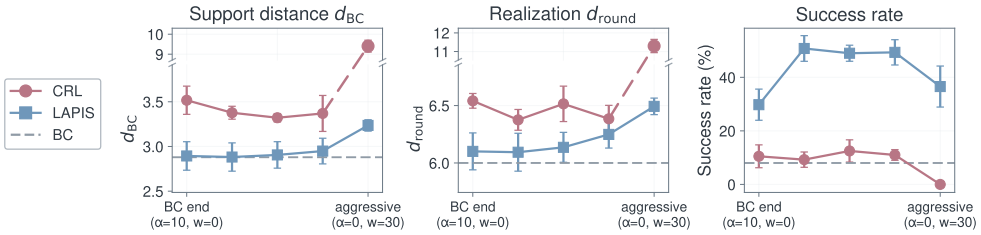

Figure 3: Support–success diagnostic (`antmaze`). We sweep CRL's BC coefficient α and LAPIS guidance w, measuring support distance d_{BC}, realization round-trip error d_{round}, and task success.

7.3 Effect of supported policy improvement (Q3)

We examine whether successful latent policy improvement remains close to the behavior-supported latent region. Starting from behavior-like policies, we strengthen critic-driven improvement by decreasing CRL's BC coefficient α and increasing LAPIS guidance w. Figure 3 measures d_{BC}, the distance from a policy latent to the nearest latent sampled from the BC policy at the same state–goal pair, and d_{round}, the round-trip latent error obtained by decoding a selected latent into physical control, executing it in the environment, and measuring the distance between the original latent and the latent re-encoded from the resulting transition; lower values indicate more faithful physical realization. As CRL becomes more aggressive, both distances increase and task success collapses, whereas LAPIS remains closer to the behavior-supported region.

7.4 Which components drive LAPIS? (Q4)

LAPIS combines critic-guided filtering with a multimodal flow actor. To isolate their contributions, we perform a 2×2 ablation crossing the policy-improvement mechanism {max-Q, gate} and actor parameterization {deterministic, flow}, while keeping the LAM, decoder, critic, and training budget fixed (Table 4). Critic-guided filtering consistently improves success with either actor parameterization, while the flow actor provides a further gain when combined with the gate.

Table 4: Improvement $\times$ actor on representative tasks under 5% labels.

Improvement	Actor	`cube`	`scene`	`antmaze`
CRL (max-Q)	det	4 ± 2	6 ± 4	11 ± 2
CRL (max-Q)	flow	5 ± 3	7 ± 4	5 ± 2
LAPIS (gate)	det	36 ± 6	16 ± 4	43 ± 4
LAPIS (gate)	flow	**62** ± 5	**29** ± 4	**49** ± 3

7.5 Effect of decoder conditioning (Q5)

We compare state-marginal $\Psi(a \mid z)$ and state-conditioned $\Psi(a \mid s, z)$ decoders across label budgets, keeping the 5%-label LAM fixed and varying only decoder supervision (Table 5). With 5% and 1% action labels, the marginal decoder achieves higher success rates across all three domains. With full action supervision, the state-conditioned decoder performs better in every domain. These results support our choice of a state-marginal decoder for LAPIS under scarce action supervision, while highlighting the benefit of state conditioning when action labels are abundant.

Table 5: Task success (%) across decoder types and action-label budgets.

Env	Decoder	100% labels	5% labels	1% labels
`cube`	$\Psi(a \mid z)$	54 ± 2	**62** ± 5	**50** ± 6
	$\Psi(a \mid s, z)$	**74** ± 4	55 ± 6	46 ± 3
`scene`	$\Psi(a \mid z)$	25 ± 4	**29** ± 4	**24** ± 5
	$\Psi(a \mid s, z)$	**38** ± 6	20 ± 5	16 ± 3
`antmaze`	$\Psi(a \mid z)$	48 ± 6	**49** ± 3	**44** ± 5
	$\Psi(a \mid s, z)$	**52** ± 4	27 ± 2	38 ± 4

8 Conclusion

We study offline policy improvement in latent action spaces learned from action-free data. We show that state-conditioned realizability, non-convex realizable sets, and outcome-based abstraction can make standard offline RL unreliable, and characterize when latent improvement transfers to physical execution. In goal-conditioned RL, LAPIS performs support-aware policy improvement through critic-guided filtering over observed state–latent transitions and a multimodal flow policy. Across visual manipulation and navigation tasks, LAPIS consistently outperforms latent-space baselines and recovers much of the performance of fully supervised counterparts with only 5% action labels.

Reproducibility statement

We document the experimental protocol and implementation in the appendix. Appendix E provides the datasets, label-budget construction, model and policy architectures, hyperparameters, per-domain configurations, and evaluation protocols. Complete statements and proofs of the main theoretical results are provided in Appendix C, with the approximate-transfer extension in Appendix D.3 and the correspondence between the analysis and practical LAPIS discussed throughout Appendix D. The empirical realizability diagnostics underlying Figure 2 are documented in Appendix B. The OGBench datasets used in our main experiments are publicly available.

Ethics statement

This work studies offline reinforcement learning on publicly available simulated robotics and navigation benchmarks. It does not involve human subjects, personal or sensitive data, or the collection or release of new datasets. As with autonomous-control research more broadly, downstream deployment may involve general dual-use considerations, but the experiments and claims in this work are limited to simulated environments.

AI use statement

We used general-purpose generative AI assistants for code inspection and debugging, manuscript and figure preparation, and checks of experimental and theoretical reasoning. All AI-assisted suggestions were reviewed and verified by the authors, who take full responsibility for the final manuscript.

References

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Théo Cachet, Christopher R. Dance, and Olivier Sigaud. Bridging environments and language with rendering functions and vision-language models. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5135–5188, 2024.

Matthew Chang, Arjun Gupta, and Saurabh Gupta. Learning value functions from undirected state-only experience. In *International Conference on Learning Representations (ICLR)*, 2022.

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

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, Jianyu Chen, and Jiang Bian. villa-X: Enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025.

Ashley D Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning (ICML)*, 2019.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *International Conference on Machine Learning (ICML)*, 2025.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning (ICML)*, 2023.

Google DeepMind. Genie 2: A large-scale foundation world model. Technical report, Google DeepMind, 2024.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1025–1037. PMLR, 2020.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Donghyeon Ki, JunHyeok Oh, Seong-Woong Shim, and Byung-Jun Lee. Prior-guided diffusion planning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*, 2006.

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

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *International Conference on Learning Representations (ICLR)*, 2025.

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

Vivek Myers, Andre He, Kuan Fang, Homer Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. Goal representations for instruction following: A semi-supervised language interface to control. In *Conference on Robot Learning (CoRL)*, 2023.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-markov decision processes. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 1011–1016, 2003.

Oleh Rybkin, Karl Pertsch, Konstantinos G Derpanis, Kostas Daniilidis, and Andrew Jaegle. Learning what you can do before doing anything. In *International Conference on Learning Representations (ICLR)*, 2019.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations (ICLR)*, 2024.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations (ICLR)*, 2025.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations (ICLR)*, 2024.

Qinqing Zheng, Mikael Henaff, Brandon Amos, and Aditya Grover. Semi-supervised offline reinforcement learning with action-free trajectories. In *International Conference on Machine Learning (ICML)*, 2023.

Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: Latent action space for offline reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

A Extended Related Work

A.1 Offline RL in Learned Latent or Skill Spaces

Prior work has used learned latent or skill representations to constrain policy optimization to behavior-like controls. PLAS (Zhou et al., 2020) learns a continuous latent action space with a conditional VAE and performs offline policy optimization through a behavior-trained decoder. Skill-based methods such as OPAL (Ajay et al., 2021), SPiRL (Pertsch et al., 2020), and latent planning approaches (Lynch et al., 2020) instead learn temporally extended action representations and optimize high-level policies over them. These approaches typically rely on action-labeled experience to learn the latent control representation or its low-level execution model. In contrast, we study offline policy improvement in a latent action space learned primarily from action-free trajectories, using scarce action labels only to ground latent decisions in physical control.

A.2 Action-Free and Semi-Supervised Reinforcement Learning

Learning control from trajectories with missing actions has been studied through imitation from observation (Torabi et al., 2018), action inference from video (Baker et al., 2022), latent-intention learning (Ghosh et al., 2023), hierarchical goal-conditioned learning (Park et al., 2023), and semi-supervised offline RL (Zheng et al., 2023). These approaches exploit action-free experience through inferred actions, state or goal abstractions, or value learning without complete action supervision. Our setting differs in that policy improvement is performed directly in a learned latent action space and physical actions are needed only for decoding. This makes the relationship between latent-space optimization and physical execution a central part of the problem.

A.3 Offline Goal-Conditioned and Contrastive RL

Offline goal-conditioned RL learns multi-goal policies from fixed datasets without additional environment interaction. Contrastive RL (Eysenbach et al., 2022) formulates goal-conditioned control through density-ratio estimation of future-state occupancy, while subsequent work develops related contrastive objectives for offline control (Zheng et al., 2024). LAPIS adopts this goal-conditioned formulation in the learned latent-action space, using future states as goals and a contrastive critic to estimate which observed state–latent transitions are useful for reaching them.

Recent work has also explored expressive generative policies and guidance-based policy extraction (Lipman et al., 2023; Ho & Salimans, 2022; Frans et al., 2025). Most closely related, CFGRL (Frans et al., 2025) derives classifier-free guidance as a controllable policy-improvement operator; in its value-free goal-conditioned formulation, guidance combines unconditional and goal-conditioned generative policy branches without an explicit value function. LAPIS shares guided flow-based extraction, but differs in where policy improvement is introduced. LAPIS first estimates goal-conditioned future-state occupancy with a TD-InfoNCE critic and filters only observed state–latent transitions to construct a high-value distribution whose support remains within the observed conditional latent support. The flow policy then models this support-aware target, and classifier-free guidance is applied only at inference.

This distinction is particularly important in learned latent-action spaces. Unlike physical actions, a latent action is a learned transition label whose usefulness depends on state-conditioned realizability and on whether decoding faithfully produces the represented outcome. Thus, improving a generative policy over latents does not by itself imply improved physical execution. Our transfer analysis characterizes when latent-space improvement survives decoding, while the direct comparison in Appendix E.5 shows that increasing goal guidance can improve physical-action CFGRL but does not provide consistent improvement when the same construction is applied directly in the learned latent space.

A.4 Relation to Batch-Constrained RL and MDP Abstraction

Our theoretical analysis is related to classical results in batch-constrained reinforcement learning and MDP abstraction, but addresses a different cross-representation question. Batch-constrained RL studies policy evaluation and improvement when the policy is restricted to actions represented in a

fixed offline dataset, thereby avoiding extrapolation to unsupported physical actions (Fujimoto et al., 2019). In our setting, however, the policy acts in a *learned* action space: observed physical transitions are relabeled by an inverse model as latent actions, and each latent decision must subsequently be decoded back to physical control. Consequently, restricting the policy to observed latent actions is not sufficient by itself to establish correspondence with physical execution. The latent support is state-dependent, a supported pair (s, z) must identify the transition outcome represented by z, and the decoder must realize that outcome through physical actions.

Theorem 1 characterizes this additional structure. We construct physical- and latent-action batch MDPs from the same observed transitions and introduce supported outcome identifiability together with a batch-exact decoder that links a supported latent decision to its physical realizations. Under deterministic dynamics, this construction yields exact equality between the return of a batch-constrained latent policy and that of its decoded physical policy. Thus, the result extends the batch-constrained perspective from policy optimization within a fixed action space to policy improvement through a learned action representation whose semantics and physical execution must themselves be related.

Our optimality analysis is also connected to MDP homomorphism and abstraction theory, which studies when an abstract decision process preserves the value structure of the underlying MDP (Ravindran & Barto, 2003; Li et al., 2006). The latent actions considered here induce an outcome-based action abstraction learned from observed transitions: distinct physical controls can map to the same latent action whenever they produce the same represented outcome. This abstraction can preserve execution while discarding distinctions that matter for reward. Theorem 2 identifies the corresponding condition for optimality alignment: when reward is determined by the transition outcome, an optimal latent policy decodes to an optimal policy within the decoder-covered, batch-constrained physical policy class. The counterexample in Appendix C.3 shows that this condition is substantive: latent-to-physical return transfer can hold exactly even when latent and physical optima do not align.

Taken together, these results connect batch-constrained and abstraction-based reasoning to learned latent actions, where the representation is inferred from action-free transitions rather than given a priori. The additional questions are therefore not only whether a policy remains within dataset support or whether an abstraction preserves an MDP, but whether a learned latent label identifies a physically executable outcome and whether the distinctions removed by that representation are compatible with the downstream objective.

B EMPIRICAL REALIZABILITY DIAGNOSTICS

The bottom-row diagnostics in Figure 2 evaluate the three structural properties discussed in Section 4.1 on `visual-cube-single-play-v0`. All latent-space distances are computed in the standardized, quantized latent coordinates used throughout the experiments.

State-conditioned realizability. For Figure 2(a), we select latents with known physical realizations at their source states and evaluate whether the same latents can be realized from other query states. For each state–latent pair, realization error is approximated by finite search over physical action chunks. The candidate set combines known realizing chunks, dataset chunks retrieved from nearby states and re-executed at the query state, and multi-start cross-entropy-method (CEM) refinements within the environment's action bounds. Rows correspond to query states and columns to source latents.

Non-convex realizable set. For Figure 2(b), we select pairs of separated latents that are both realizable from the same state and linearly interpolate between them. Realization error is evaluated on a uniform interpolation grid. The action chunks realizing both endpoints are included in the candidate set at every interpolation point, together with the same dataset-based and CEM search used above. Confidence bands are computed across latent pairs.

Diagnostic threshold. Panels (a–b) use a common threshold calibrated from two finite-search error samples. Let D_{self} contain errors for known-realizable source latents searched at their source states without supplying their original realizing chunks, and let D_{cross} contain errors for those latents searched from different states. Using 16 anchor states and CEM population 256 with 5 iterations, we set

$$\epsilon = \frac{Q_{0.9}(D_{\text{self}}) + Q_{0.1}(D_{\text{cross}})}{2} \approx 3.14,$$

where Q_p denotes the empirical p-quantile. We use the rounded value 3.14 in both panels.

Outcome-based abstraction. For Figure 2(c), we test whether substantially different physical action chunks can realize the same target outcome. For each target, targeted CEM search minimizes outcome distance, with restarts repelled from previously discovered chunks to encourage distinct realizations. Outcome distance is Euclidean in simulator generalized-position coordinates standardized using a candidate rollout pool at each anchor state, while action distance is Euclidean over flattened chunks. A candidate is retained when its outcome distance is at most τ_{out} and its distance from every previously retained chunk is at least τ_a. We use 12 anchor states with 8 target outcomes each and 20 CEM restarts per target, all run without early stopping. At each anchor, τ_{out} is set to the 1st percentile of pairwise outcome distances and τ_a to the 40th percentile of pairwise action-chunk distances. Each target's observed source chunk is included as the first realization.

Search-budget robustness. Because realization error is estimated through finite action-space search, we additionally repeat the search with a substantially larger CEM budget. The default search uses population 256 and 5 iterations, while the enlarged search uses population 1024 and 12 iterations. We also evaluate a known-realizable floor by searching for source latents without providing their original realizing action chunks. This comparison tests whether the conclusions of panels (a–b) persist under substantially stronger search, without interpreting failure to find a realization as proof of global unrealizability. Results are summarized in Table 6.

Table 6: Realization error under different search budgets (lower is better; $\epsilon = 3.14$). Interpolation denotes interior points in Figure 2(b), and state swap denotes off-diagonal pairs in Figure 2(a).

Search	Population × iterations	Interpolation	State swap
Default	256×5	4.02	4.12
Enlarged	1024×12	3.69	3.51

C PROOFS AND THEORETICAL DERIVATIONS

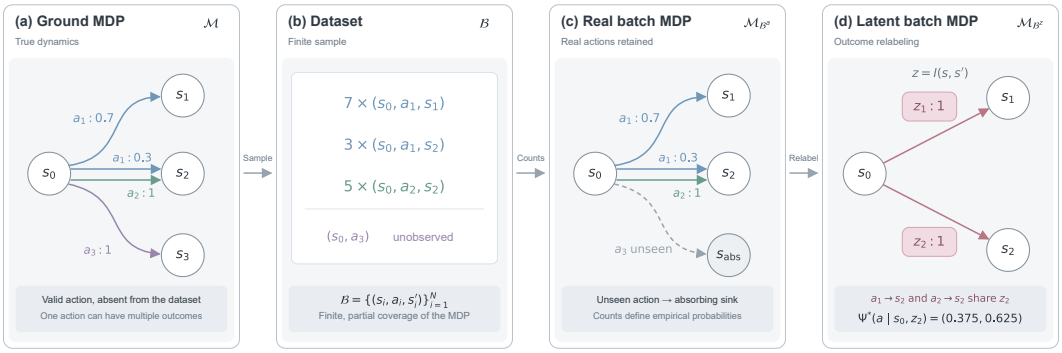

Figure 4: **Paired real-action and latent-action batch MDPs. (a)** A real action in the ground MDP can induce multiple outcomes. **(b,c)** The finite dataset defines an empirical real-action batch MDP, with unseen actions routed to an absorbing sink. **(d)** Relabeling transitions by $z = I(s, s')$ separates distinct outcomes while merging different real actions that reach the same successor.

We first formalize the paired physical and latent batch MDPs, then establish policy improvement transfer, optimal-policy alignment, and the supported reweighting guarantee underlying LAPIS. Appendix C.1 fixes notation and gives the exact construction of the paired physical and latent batch MDPs together with the batch-exact decoder Ψ^*. Appendix C.2 proves the on-support transfer result (Theorem 1). Appendix C.3 defines the decoder-covered policy class and proves optimal-policy alignment (Theorem 2). Appendix C.4 establishes the supported reweighting guarantee underlying LAPIS's policy improvement principle. The correspondence between this analysis and practical LAPIS, together with the associated diagnostics, is discussed in Appendix D. We discuss the relationship of these results to classical batch-constrained RL and MDP abstraction theory in Appendix A.4.

C.1 BATCH-MDP DEFINITIONS AND DECODER

Throughout the analysis, we assume bounded rewards, $R_{\max} := \sup_{s,a,s'} |r(s, a, s')| < \infty$, and a discount factor $0 \le \gamma < 1$. For a policy π in an MDP $\mathcal{N}$, we write $J_{\mathcal{N}}(\pi) = \mathbb{E}_{s_0 \sim \mu_0}[V^{\pi}_{\mathcal{N}}(s_0)]$, using the same initial-state distribution μ_0 for all comparisons. For the analysis, $\mathcal{B}^a$ pairs each observed transition with the physical action that generated it, whether or not that action label is available to the learning algorithm. The batch-MDP rewards are defined for the analysis; the learning algorithm does not require reward labels.

Support domain and termination. Let $S_{\mathcal{B}}$ be the set of states and successors observed in the batch, and let $N(s, a, s')$ denote the number of occurrences of (s, a, s') in $\mathcal{B}^a$. The observed physical and latent actions at state s are

$$A_{\mathcal{B}}(s) := \left\{ a : \sum_{s'} N(s, a, s') > 0 \right\}, \qquad Z_{\mathcal{B}}(s) := \left\{ I(s, s') : \sum_{a} N(s, a, s') > 0 \right\}.$$

We assume $\operatorname{supp} \mu_0 \subseteq S_{\mathcal{B}}$ and consider a nonempty class of policies that remain on observed pairs until true termination. True terminal states have zero-reward absorbing self-loops in both the physical and latent processes. A trajectory cutoff is not treated as termination: a policy that reaches a nonterminal state with no observed continuation is excluded from the comparison class. All exact comparisons concern states reachable from μ_0 under the policies being compared.

Both batch MDPs are defined on $S \cup \{s_{\text{abs}}\}$, with unobserved pairs routed to the zero-reward absorbing state s_{abs}. Starting from $S_{\mathcal{B}}$, their transitions remain within the finite set $S_{\mathcal{B}} \cup \{s_{\text{abs}}\}$, even when the ground state space is continuous. For a fixed policy, the Bellman operator is a γ-contraction on bounded functions and has a unique fixed point. Sums in the exact analysis are over finite empirical support; in the continuous-kernel extension of Appendix D.3, they are interpreted as integrals.

Definition 3 (Real action batch MDP)**.** *The physical-action batch MDP* $\mathcal{M}_{\mathcal{B}^a} = (S \cup \{s_{\text{abs}}\}, A, p_{\mathcal{B}^a}, r_{\mathcal{B}^a}, \gamma)$ *uses the empirical transition probabilities*

$$p_{\mathcal{B}^a}(s' \mid s, a) = \frac{N(s, a, s')}{\sum_{\tilde{s}} N(s, a, \tilde{s})} \quad \text{for } a \in A_{\mathcal{B}}(s).$$

On observed transitions, $r_{\mathcal{B}^a}(s, a, s') = r(s, a, s')$*. For unobserved pairs,* $p_{\mathcal{B}^a}(s_{\text{abs}} \mid s, a) = 1$*. The sink is absorbing, and all transitions into or within it receive zero reward. Consequently,* $\|V^{\pi}_{\mathcal{M}_{\mathcal{B}^a}}\|_\infty \le R_{\max}/(1-\gamma)$ *for every policy* π*.*

Definition 4 (Supported outcome identifiability)**.** *At a fixed state* s*, any two observed successors* s'_1, s'_2 *satisfy*

$$I(s, s'_1) = I(s, s'_2) \quad \Longrightarrow \quad s'_1 = s'_2.$$

Thus, each $z \in Z_{\mathcal{B}}(s)$ *identifies a unique observed successor, denoted* $F_{\mathcal{B}}(s, z)$*.*

This condition applies only to observed successors at the same state. It allows distinct physical actions to share a latent when they reach the same successor, but excludes different observed successors from sharing a latent. The map $F_{\mathcal{B}}$ is an exact lookup on the supported transition graph, not the learned forward predictor F in Equation (3). Its definition does not require exact reconstruction by F, and the analysis evaluates $F_{\mathcal{B}}$ only at supported pairs.

Definition 5 (Latent action batch MDP)**.** *Relabel each observed transition with* $z_i = I(s_i, s'_i)$ *to obtain* $\mathcal{B}^z = \{(s_i, z_i, s'_i)\}_{i=1}^N$*, and let* $N^z(s, z, s')$ *denote the relabeled transition counts. Under Definition 4, the latent-action batch MDP* $\mathcal{M}_{\mathcal{B}^z} = (S \cup \{s_{\text{abs}}\}, Z, p_{\mathcal{B}^z}, r_{\mathcal{B}^z}, \gamma)$ *has, for* $z \in Z_{\mathcal{B}}(s)$*,*

$$p_{\mathcal{B}^z}(\cdot \mid s, z) = \delta_{F_{\mathcal{B}}(s,z)},$$
$$r_{\mathcal{B}^z}(s, z) = \frac{\sum_a N(s, a, F_{\mathcal{B}}(s, z)) r(s, a, F_{\mathcal{B}}(s, z))}{\sum_a N(s, a, F_{\mathcal{B}}(s, z))}.$$

The transition kernel is equivalently obtained by normalizing the relabeled counts. The reward averages the physical rewards associated with the represented successor, using their empirical frequencies. For unsupported pairs, $p_{\mathcal{B}^z}(s_{\text{abs}} \mid s, z) = 1$ *and* $r_{\mathcal{B}^z}(s, z) = 0$*, with the same absorbing-sink convention as in* $\mathcal{M}_{\mathcal{B}^a}$*. For every latent policy* π_z*,* $\|V^{\pi_z}_{\mathcal{M}_{\mathcal{B}^z}}\|_\infty \le R_{\max}/(1-\gamma)$*.*

Batch-exact decoder. For a supported pair (s, z), write $s'_z = F_{\mathcal{B}}(s, z)$. The batch-exact decoder selects physical actions according to their observed frequencies among transitions from s to s'_z:

$$\Psi^*(a \mid s, z) = \frac{N(s, a, s'_z)}{\sum_b N(s, b, s'_z)} = \frac{\pi_{\beta,\mathcal{B}}(a \mid s) p_{\mathcal{B}^a}(s'_z \mid s, a)}{\sum_b \pi_{\beta,\mathcal{B}}(b \mid s) p_{\mathcal{B}^a}(s'_z \mid s, b)}, \tag{6}$$

Here, $\pi_{\beta,\mathcal{B}}(a \mid s) = \sum_{s'} N(s, a, s') / \sum_{b,s'} N(s, b, s')$ is the empirical action distribution at s. The denominators in Equation (6) are positive on the supported domain.

Equation (6) is an empirical Bayes decoder: it conditions the batch action distribution on the represented successor using $p_{\mathcal{B}^a}$, rather than the ground transition kernel p. Under deterministic ground dynamics, these kernels agree on observed state–action pairs, so the decoded actions reproduce the represented successor. Under stochastic dynamics, an action observed to reach s'_z may reach a different successor when executed again. This analytical decoder $\Psi^*(a \mid s, z)$ is distinct from the learned state-marginal decoder $\Psi(a \mid z)$ used by LAPIS; their relationship is discussed in Appendix D.4.

Illustrative example. Figure 4 illustrates how the same observed transitions define the physical- and latent-action batch MDPs. The physical-action batch MDP retains the empirical successor distribution of each observed action and routes the unobserved action a_3 to the absorbing sink, even though it is valid in the ground MDP. Relabeling by $z = I(s, s')$ separates the two observed successors of a_1 into distinct latents z_1 and z_2, while assigning the transitions of both a_1 and a_2 to s_2 the same latent z_2. The batch-exact decoder selects physical actions according to their frequencies for that successor: the three a_1-transitions and five a_2-transitions to s_2 give decoding probabilities $3/8$ and $5/8$, respectively. Because a_1 is stochastic, decoding z_2 need not reproduce s_2, despite its unique successor in the latent batch MDP. This distinction motivates the deterministic-dynamics assumption used for exact transfer in the following subsection.

C.2 Proof of Theorem 1

We first establish that batch constraints are preserved by decoding: selecting only observed latent actions at a state also restricts physical execution to actions observed at that state.

Remark 1 (Batch constraints under decoding). At any state s, if $\pi_z(\cdot \mid s)$ is supported on $Z_{\mathcal{B}}(s)$, then the decoded physical policy $(\Psi^* \circ \pi_z)(\cdot \mid s)$ is supported on $A_{\mathcal{B}}(s)$.

Proof. By Equation (6), $\Psi^*(a \mid s, z) > 0$ only if $N(s, a, F_{\mathcal{B}}(s, z)) > 0$, which implies $a \in A_{\mathcal{B}}(s)$. Averaging these decoder distributions over $z \sim \pi_z(\cdot \mid s)$ therefore preserves the physical-action support restriction. □

Under deterministic dynamics, these decoded actions also reproduce the successor represented by the selected latent. Together with the batch reward construction, this correspondence yields equality between latent and decoded physical returns.

Theorem 1 (On-support policy improvement transfer). *Under Assumption 1, every batch-constrained latent policy π_z satisfies $J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) = J_{\mathcal{M}}(\Psi^* \circ \pi_z)$. Consequently, return improvement between two such latent policies transfers to their batch-exact decoded physical policies.*

Proof. Fix a supported pair (s, z) and write $s'_z = F_{\mathcal{B}}(s, z)$. Every action selected by $\Psi^*(\cdot \mid s, z)$ was observed to produce s'_z from s. Under Assumption 1, each such action satisfies $T(s, a) = s'_z$, giving

$$\sum_a \Psi^*(a \mid s, z) p(\cdot \mid s, a) = \delta_{s'_z} = p_{\mathcal{B}^z}(\cdot \mid s, z).$$

The decoder weights also match those used to define the latent reward:

$$\sum_a \Psi^*(a \mid s, z) r(s, a, s'_z) = \frac{\sum_a N(s, a, s'_z) r(s, a, s'_z)}{\sum_a N(s, a, s'_z)} = r_{\mathcal{B}^z}(s, z).$$

Thus, the one-step transition and expected reward agree for every supported pair. Individual decoded actions may receive different rewards, but their expectation matches the latent reward; Assumption 2 is not required.

Starting from the common initial distribution μ_0, the latent policy selects only supported latents, and Remark 1 ensures that decoding selects observed physical actions. The transition identity above then implies, by induction, that the two processes have the same state distribution at every step. The continuation and terminal conventions in Appendix C.1 ensure that both processes remain well-defined on this common reachable support.

Averaging over π_z, the two processes therefore share the state-level transition kernel and expected reward

$$P_{\pi_z}(\cdot \mid s) = \sum_z \pi_z(z \mid s) \delta_{F_{\mathcal{B}}(s,z)}, \qquad \bar{r}_{\pi_z}(s) = \sum_z \pi_z(z \mid s) r_{\mathcal{B}^z}(s, z).$$

Both $V^{\pi_z}_{\mathcal{M}_{\mathcal{B}^z}}$ and $V^{\Psi^* \circ \pi_z}_{\mathcal{M}}$ satisfy

$$V = \bar{r}_{\pi_z} + \gamma P_{\pi_z} V$$

on the common reachable support. Since $\gamma < 1$, this Bellman equation has a unique bounded solution, so the two value functions agree there. Averaging over μ_0 establishes return equality. Applying this equality separately to π_z and π'_z proves the improvement implication. □

C.3 Proof of Theorem 2 and the decoder-covered policy class

Theorem 1 establishes return equality between a latent policy and its own decoded physical policy. This does not yet imply that an optimal latent policy attains the best return available to batch-constrained physical policies.

To see why, consider a deterministic one-step problem with two equally frequent observed actions satisfying $T(s, a_1) = T(s, a_2) = s_{\text{term}}$, but receiving rewards 1 and 0, respectively. Both actions share one latent, and the batch-exact decoder selects each with probability $1/2$. The latent and decoded returns are therefore both $1/2$, consistent with Theorem 1. A physical policy can instead

select a_1 and obtain return 1. Thus, outcome-based abstraction can create an optimality gap even without off-support actions or decoder error.

To establish alignment, we construct a value-equivalent latent counterpart for each physical policy in the comparison class. We first define the required coverage in terms of represented successors.

Definition 6 (Decoder-covered policy)**.** *Under deterministic dynamics, a physical policy π is* decoder-covered *from μ_0 if, at every nonterminal state reachable under π from μ_0, each action in its support satisfies*

$$T(s,a) = F_{\mathcal{B}}(s,z) \quad \textit{for some } z \in Z_{\mathcal{B}}(s).$$

This condition requires the successor itself to be represented in the batch, rather than merely requiring its code $I(s,T(s,a))$ to have appeared. Under Assumption 1, every batch-constrained physical policy is decoder-covered because each observed action reaches its recorded successor. Coverage can also include unobserved actions that reach the same successors as observed actions, although Theorem 2 compares only the batch-constrained subclass.

Under Assumption 2, actions reaching the same successor from a given state also receive the same reward. We can therefore group a decoder-covered policy's action probabilities by successor without changing its return.

Lemma 1 (Decoder-covered lift)**.** *Under Assumptions 1–2, a decoder-covered physical policy π has the latent lift*

$$\pi_z(z \mid s) = \sum_a \pi(a \mid s)\mathbf{1}[T(s,a) = F_{\mathcal{B}}(s,z)], \qquad z \in Z_{\mathcal{B}}(s),$$

on its reachable support. This lift is batch-constrained and satisfies $J_{\mathcal{M}}(\pi) = J_{\mathcal{M}}(\Psi^* \circ \pi_z) = J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)$. The lift may be extended arbitrarily outside the reachable domain. For a continuous physical-action policy, the sum over a is interpreted as an integral; the supported latent set remains finite.

Proof. Decoder coverage ensures that every selected action reaches a represented successor. Since the encoder is deterministic and supported outcome identifiability holds, each action is assigned to exactly one supported latent. The displayed probabilities therefore sum to one and define a policy supported on $Z_{\mathcal{B}}(s)$.

By construction, at every reachable nonterminal state,

$$\sum_z \pi_z(z \mid s)\delta_{F_{\mathcal{B}}(s,z)} = \sum_a \pi(a \mid s)\delta_{T(s,a)}.$$

The left-hand side is also the successor distribution induced by $\Psi^* \circ \pi_z$, as established in the proof of Theorem 1. Thus, π and its lifted-and-decoded policy have the same successor distribution. Under Assumption 2, they also have the same expected reward.

Starting from μ_0, these identities imply that the two policies induce the same state distributions at every step. Their value functions satisfy the same Bellman equation on the common reachable support and hence agree there, giving

$$J_{\mathcal{M}}(\pi) = J_{\mathcal{M}}(\Psi^* \circ \pi_z).$$

The lift selects only observed latents along this reachable support and satisfies the continuation convention of Appendix C.1, so it is batch-constrained. Theorem 1 then gives the remaining equality with $J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)$. □

Every batch-constrained physical policy therefore has a latent counterpart with the same return. Conversely, decoding a batch-constrained latent policy preserves its return and produces a batch-constrained physical policy. These two correspondences establish optimal-policy alignment.

Theorem 2 (Optimal-policy alignment). Under Assumptions 1–2, for the same initial distribution μ_0,

$$\pi_z^* \in \arg\max_{\pi_z \in \Pi_{\mathcal{B}^z}} J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z) \quad \Longrightarrow \quad (\Psi^* \circ \pi_z^*) \in \arg\max_{\pi \in \Pi_{\mathcal{B}^a}} J_{\mathcal{M}}(\pi).$$

Proof. Let π_z^* maximize $J_{\mathcal{M}_{\mathcal{B}^z}}$ over $\Pi_{\mathcal{B}^z}$. For any $\pi \in \Pi_{\mathcal{B}^a}$, decoder coverage follows from Assumption 1, and Lemma 1 provides a lift $\widetilde{\pi}_z \in \Pi_{\mathcal{B}^z}$ with the same return. Consequently,

$$J_{\mathcal{M}}(\pi) = J_{\mathcal{M}_{\mathcal{B}^z}}(\widetilde{\pi}_z) \leq J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z^*) = J_{\mathcal{M}}(\Psi^* \circ \pi_z^*),$$

where the inequality follows from latent optimality and the last equality follows from Theorem 1.

Moreover, $(\Psi^* \circ \pi_z^*) \in \Pi_{\mathcal{B}^a}$ by Remark 1 and the reachable-support argument in Theorem 1. The decoded policy therefore belongs to the physical comparison class and attains at least as much return as every policy in that class, proving the claimed optimality. In particular,

$$J_{\mathcal{M}}(\Psi^* \circ \pi_z^*) = \max_{\pi \in \Pi_{\mathcal{B}^a}} J_{\mathcal{M}}(\pi) = J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z^*).$$

The alignment concerns the stated batch-constrained policy classes, not the unrestricted physical optimum. □

C.4 Supported reweighting guarantee

We now formalize the policy-improvement principle underlying critic-guided filtering. The update reweights a fixed reference policy toward higher-value latent actions without introducing actions outside its support. Throughout this subsection, we fix a goal g and use the common closed supported comparison domain of Appendix C.1.

We first show that a reweighting that favors higher-value actions cannot decrease the expected action value under the reference policy.

Lemma 2 (Co-monotone reweighting)**.** *Fix (s, g) and a reference latent policy $\mu(\cdot \mid s, g)$ on the finite supported latent set. Let $\omega(z)$ be bounded, nonnegative, and nondecreasing in $Q^\mu(s, z, g)$, with $C_\omega := \mathbb{E}_\mu[\omega] > 0$. Then the reweighted policy*

$$\mu^\omega(z \mid s, g) = \frac{\mu(z \mid s, g)\omega(z)}{C_\omega}$$

satisfies

$$\mathbb{E}_{\mu^\omega}[Q^\mu(s, z, g)] \geq \mathbb{E}_\mu[Q^\mu(s, z, g)] = V^\mu(s, g).$$

Proof. Let $z, \tilde{z}$ be independent samples from $\mu(\cdot \mid s, g)$, and abbreviate $Q_z = Q^\mu(s, z, g)$. Since the weights are nondecreasing in value,

$$0 \leq \tfrac{1}{2}\mathbb{E}[(Q_z - Q_{\tilde{z}})(\omega(z) - \omega(\tilde{z}))] = \mathbb{E}_\mu[Q\omega] - \mathbb{E}_\mu[Q]\mathbb{E}_\mu[\omega].$$

Dividing by $C_\omega > 0$ gives

$$\mathbb{E}_{\mu^\omega}[Q] = \frac{\mathbb{E}_\mu[Q\omega]}{C_\omega} \geq \mathbb{E}_\mu[Q],$$

which proves the claim. □

The weights need not depend directly on the numerical value of Q^μ. Advantage-based weights and thresholded scores also satisfy the required ordering.

Corollary 1. *A bounded, nonnegative, nondecreasing function h of the advantage*

$$A^\mu(s, z, g) = Q^\mu(s, z, g) - V^\mu(s, g)$$

defines weights $\omega(z) = h(A^\mu(s, z, g))$ satisfying Lemma 2 whenever the normalizing mass is positive. More generally, the thresholded weights

$$\omega(z) = \mathbf{1}[f(s, z, g) > b(s, g)]$$

satisfy Lemma 2 whenever $f(s, \cdot, g)$ is nondecreasing in $Q^\mu(s, \cdot, g)$, b is independent of z, and the normalizing mass is positive.

Proof. Subtracting a baseline independent of z preserves the value ordering. Applying a nondecreasing function or thresholding a nondecreasing score at a z-independent cutoff likewise preserves this ordering. The resulting weights therefore satisfy Lemma 2 under the stated positivity condition. □

This observation connects value-based reweighting to the contrastive score used to motivate LAPIS.

Remark 2 (Density-ratio scores and value ordering). For the binary density-ratio objective introduced in Section 3, the optimal score is

$$f^\star(s, z, g) = \log Q^\mu(s, z, g) - \log p_{\mathcal{B}}(g)$$

wherever the ratio is defined. At fixed (s, g), the term $-\log p_{\mathcal{B}}(g)$ is independent of z, so $f^\star$ preserves the ordering of Q^μ. A z-independent score threshold therefore produces nonnegative weights that are nondecreasing in value; the threshold need not equal $V^\mu(s, g)$. Appendix D.7 relates this score-space filtering principle to the learned critic.

Applying the statewise inequality at every state yields a return guarantee for the reweighted policy. When all weights vanish, retaining the reference policy makes the update well-defined.

Proposition 1 (Goal-conditioned supported improvement)**.** *Let π^z_β be a batch-constrained reference policy, with advantage A^z_β in $\mathcal{M}_{\mathcal{B}^z}$. For $\eta > 0$, define*

$$\pi^z_\eta(z \mid s, g) \propto \pi^z_\beta(z \mid s, g) h(A^z_\beta(s, z, g))^\eta,$$

where h is bounded, nonnegative, and nondecreasing. At states with zero normalizing mass, set $\pi^z_\eta = \pi^z_\beta$, and define $\pi^z_0 = \pi^z_\beta$. Then

$$\operatorname{supp} \pi^z_\eta(\cdot \mid s, g) \subseteq \operatorname{supp} \pi^z_\beta(\cdot \mid s, g), \qquad J_{\mathcal{M}_{\mathcal{B}^z}}(\pi^z_\eta; g) \geq J_{\mathcal{M}_{\mathcal{B}^z}}(\pi^z_\beta; g).$$

Under the conditions of Theorem 1, the same return ordering holds for their batch-exact decoded physical policies.

Proof. Reweighting cannot assign positive probability to a latent action outside the reference policy's support, and the fallback leaves the reference policy unchanged. The updated policy therefore remains batch-constrained on the common supported domain.

For $\eta > 0$, the weights $h(A^z_\beta)^\eta$ remain nonnegative and nondecreasing in the reference action value. At each state with positive normalizing mass, Lemma 2 gives

$$\mathbb{E}_{z \sim \pi^z_\eta(\cdot \mid s, g)} \left[Q^{\pi^z_\beta}_{\mathcal{M}_{\mathcal{B}^z}}(s, z, g) \right] \geq V^{\pi^z_\beta}_{\mathcal{M}_{\mathcal{B}^z}}(s, g).$$

At fallback states, or when $\eta = 0$, equality holds. Writing $V_\beta = V^{\pi^z_\beta}_{\mathcal{M}_{\mathcal{B}^z}}(\cdot, g)$ and $\mathcal{T}^{\pi^z_\eta}$ for the updated policy's Bellman operator, this inequality is

$$\mathcal{T}^{\pi^z_\eta} V_\beta \geq V_\beta.$$

By monotonicity, repeated application of $\mathcal{T}^{\pi^z_\eta}$ produces a nondecreasing sequence starting from V_β. Since the operator is a γ-contraction, this sequence converges to $V^{\pi^z_\eta}_{\mathcal{M}_{\mathcal{B}^z}}(\cdot, g)$, establishing pointwise value improvement. Averaging over μ_0 proves the return inequality.

Both policies are batch-constrained, so Theorem 1 transfers this ordering to their decoded physical returns under its assumptions. The argument holds for each fixed goal and therefore also after averaging over a common goal distribution. □

The exponent η controls distributional reweighting and is distinct from the flow-guidance scale w in Section 6. In particular, for the hard gate $h = \mathbf{1}[\cdot > 0]$, we have $h^\eta = h$ for every $\eta > 0$, so the reweighted policy is independent of η over this range. The proposition concerns this exact distributional update; guided flow samples need not follow the same distribution. Their relationship is discussed in Appendix D.8.

D THEORY–PRACTICE CORRESPONDENCE

The analysis in Section 5 isolates the conditions under which latent policy improvement remains valid after physical execution. Practical LAPIS realizes these conditions through learned latent representations, decoding, value estimation, and policy extraction. This appendix makes each correspondence explicit and extends the exact analysis to the learned components used in practice. Exact transfer arises as the zero-mismatch case of the transition and reward correspondence below; supported reweighting defines the target distribution approximated by the learned policy.

D.1 LEARNED REPRESENTATIONS AND SUPPORTED OUTCOME IDENTIFIABILITY

Supported outcome identifiability (Definition 4) requires each supported state–latent pair (s, z) to identify a unique observed successor $F_{\mathcal{B}}(s, z)$. In practice, this structure is learned through the inverse model I and forward model F. Reconstruction connects the learned representation to outcome identification. If two observed successors are assigned the same latent, $I(s, s'_1) = I(s, s'_2) = z$, then

$$\|s'_1 - s'_2\|_2 \leq \|s'_1 - F(s, z)\|_2 + \|s'_2 - F(s, z)\|_2.$$

Thus, when both transitions are reconstructed accurately, successors sharing a latent are close in the reconstruction metric. The theorem states supported outcome identifiability as an exact structural condition, while the reconstruction objective encourages this structure on observed transitions.

The experimental policy operates on quantized latent codes. Identifiability concerns whether the code supplied to the policy and decoder distinguishes the observed outcomes relevant for control. The learned predictor F and the exact supported lookup $F_{\mathcal{B}}$ play complementary roles: F is trained to reconstruct outcomes, whereas $F_{\mathcal{B}}$ states the exact correspondence on the observed transition graph. Accurate reconstruction encourages outcome consistency; exact identifiability additionally requires that the supported code distinguish these outcomes.

D.2 STATE REPRESENTATIONS AND MARKOV CONTROL

The analysis in Section 5 uses Markov states to isolate the relationship between latent decisions and physical execution from perceptual representation learning. The experiments use single-frame RGB observations processed by learned visual encoders. Practical LAPIS applies the latent policy improvement principle to this learned observation representation.

Different underlying states can produce indistinguishable observations, so the observation-level process need not be exactly Markov. This concerns the state representation, whereas latent-action realizability concerns whether a selected latent transition can be physically executed from the current state. The exact results characterize the control-level correspondence under Markov state information. Temporal or recurrent representations provide possible ways to retain additional state information; the analysis applies when the resulting representation satisfies the stated Markov and transition conditions.

D.3 APPROXIMATE TRANSFER UNDER EXECUTION MISMATCH

Theorem 1 gives exact return equality when supported latent transitions are executed through the batch-exact decoder under deterministic dynamics. More generally, the same correspondence admits a quantitative extension whenever decoded execution induces transitions and rewards close to those used for latent evaluation. We state the bound for a decoder $\widehat{\Psi}(a \mid s, z)$, which includes both Ψ^* and the state-marginal choice $\widehat{\Psi}(a \mid s, z) = \Psi(a \mid z)$.

Fix a latent policy π_z that is batch-constrained in $\mathcal{M}_{\mathcal{B}^z}$. We compare the two processes on a common Markov state space and decision interval, with the same γ, reward bound $R_{\max}$, and initial distribution μ_0. Use the ambient-space batch kernels of Appendix C.1, extend the ground process with the same absorbing sink, and define π_z and $\widehat{\Psi}$ as probability kernels throughout this space. Where an empirical policy or decoder is undefined, fix an arbitrary probability-kernel extension. The occupancy below includes any off-support visits under this specified execution. Sums over continuous variables denote integrals.

The decoder induces

$$\tilde{p}_{\widehat{\Psi}}(s' \mid s, z) := \sum_a \widehat{\Psi}(a \mid s, z)\, p(s' \mid s, a),$$
$$\tilde{r}_{\widehat{\Psi}}(s, z) := \sum_a \widehat{\Psi}(a \mid s, z) \sum_{s'} p(s' \mid s, a)\, r(s, a, s').$$

For this fixed decoder, define the mismatches

$$\epsilon_p(s, z) := D_{\mathrm{TV}}\big(p_{\mathcal{B}^z}(\cdot \mid s, z), \tilde{p}_{\widehat{\Psi}}(\cdot \mid s, z)\big),$$
$$\epsilon_r(s, z) := \big|r_{\mathcal{B}^z}(s, z) - \tilde{r}_{\widehat{\Psi}}(s, z)\big|,$$

where $D_{\mathrm{TV}}(P, Q) = \sup_A |P(A) - Q(A)|$. Let $d^{\widehat{\Psi}\circ\pi_z}$ be the normalized γ-discounted occupancy of (s, z) under decoded execution in $\mathcal{M}$ from μ_0, with $z \sim \pi_z(\cdot \mid s)$. Write $\bar{\epsilon}_p = \mathbb{E}_{d^{\widehat{\Psi}\circ\pi_z}}[\epsilon_p]$ and $\bar{\epsilon}_r = \mathbb{E}_{d^{\widehat{\Psi}\circ\pi_z}}[\epsilon_r]$.

Proposition 2 (Approximate transfer under execution mismatch)**.** *For a batch-constrained latent policy π_z and the fixed, globally defined decoder $\widehat{\Psi}$ above,*

$$\big|J_{\mathcal{M}}(\widehat{\Psi} \circ \pi_z) - J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)\big| \le \frac{\bar{\epsilon}_r}{1-\gamma} + \frac{2\gamma R_{\max}}{(1-\gamma)^2}\,\bar{\epsilon}_p. \tag{7}$$

Zero transition and reward mismatch yields exact return equality. In particular, $\widehat{\Psi} = \Psi^*$ under Assumption 1 recovers Theorem 1.

Proof. Step 1: Bellman residual. Write $V_1 = V_{\mathcal{M}}^{\widehat{\Psi}\circ\pi_z}$ and $V_2 = V_{\mathcal{M}_{\mathcal{B}^z}}^{\pi_z}$. Their Bellman equations are

$$V_1(s) = \sum_z \pi_z(z \mid s)\Big[\tilde{r}_{\widehat{\Psi}}(s, z) + \gamma \sum_{s'} \tilde{p}_{\widehat{\Psi}}(s' \mid s, z) V_1(s')\Big],$$
$$V_2(s) = \sum_z \pi_z(z \mid s)\Big[r_{\mathcal{B}^z}(s, z) + \gamma \sum_{s'} p_{\mathcal{B}^z}(s' \mid s, z) V_2(s')\Big].$$

Let $(\bar{P}_{\widehat{\Psi}} h)(s) = \sum_{z,s'} \pi_z(z \mid s)\tilde{p}_{\widehat{\Psi}}(s' \mid s, z)h(s')$. Subtracting the equations gives $\Delta = \delta + \gamma \bar{P}_{\widehat{\Psi}}\Delta$, where $\Delta = V_1 - V_2$ and

$$\delta(s) = \sum_z \pi_z(z \mid s)\Big[\tilde{r}_{\widehat{\Psi}}(s, z) - r_{\mathcal{B}^z}(s, z) + \gamma \sum_{s'} (\tilde{p}_{\widehat{\Psi}} - p_{\mathcal{B}^z})(s' \mid s, z) V_2(s')\Big].$$

Step 2: bounding the residual. Definition 5 gives $\|V_2\|_\infty \le R_{\max}/(1-\gamma)$. The total-variation inequality $|\mathbb{E}_P h - \mathbb{E}_Q h| \le 2\|h\|_\infty D_{\mathrm{TV}}(P, Q)$ therefore yields

$$|\delta(s)| \le \sum_z \pi_z(z \mid s)\Big[\epsilon_r(s, z) + \frac{2\gamma R_{\max}}{1-\gamma}\epsilon_p(s, z)\Big].$$

Step 3: occupancy-weighted bound. Since $\bar{P}_{\widehat{\Psi}}$ is a stochastic operator and $\gamma < 1$, unrolling the residual along decoded execution gives

$$\begin{aligned}\big|J_{\mathcal{M}}(\widehat{\Psi} \circ \pi_z) - J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)\big| &\le \sum_{t\ge 0} \gamma^t \mathbb{E}[|\delta(s_t)|] \\ &\le \frac{1}{1-\gamma}\mathbb{E}_{d^{\widehat{\Psi}\circ\pi_z}}\left[\epsilon_r + \frac{2\gamma R_{\max}}{1-\gamma}\epsilon_p\right],\end{aligned}$$

which is the stated bound. If $\widehat{\Psi} = \Psi^*$ and Assumption 1 holds, the supported transition and expected-reward identities in the proof of Theorem 1 make both mismatches zero along the rollout. This recovers exact transfer without requiring Assumption 2. □

For the fixed decoder, let $\varepsilon_{\widehat{\Psi}}(\pi_z)$ denote the right-hand side of Equation (7), using the occupancy of $\widehat{\Psi} \circ \pi_z$.

Corollary 2 (Approximate improvement transfer)**.** *For batch-constrained latent policies π_z, π_z' and the same decoder $\widehat{\Psi}$, define $\Delta_z = J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z') - J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)$. Then*

$$J_{\mathcal{M}}(\widehat{\Psi} \circ \pi_z') - J_{\mathcal{M}}(\widehat{\Psi} \circ \pi_z) \ge \Delta_z - \varepsilon_{\widehat{\Psi}}(\pi_z') - \varepsilon_{\widehat{\Psi}}(\pi_z).$$

Thus a latent improvement exceeding the two execution-gap bounds transfers to physical improvement.

Proof. Add and subtract the two latent returns and apply Proposition 2 separately to π_z' and π_z. □

Interpretation. The bound accounts jointly for stochastic dynamics and decoder approximation through their induced transition and reward mismatch. Its quantitative usefulness depends on this mismatch being small in the stated metric: a point mass and an atomless transition law have total-variation distance one even if they are geometrically close. Appendix D.6 describes the experimental controller.

D.4 Learned decoding under scarce action supervision

The batch-exact decoder establishes an exact correspondence, while successful execution fundamentally requires realizing the represented successor. Recovering a particular recorded action distribution is unnecessary when alternative controls induce the same transition and reward. Under Assumptions 1–2, a decoder satisfying

$$\widehat{\Psi}\big(\{a : T(s,a) = F_{\mathcal{B}}(s,z)\} \mid s, z\big) = 1$$

on visited supported pairs induces the same successor and reward as the latent transition. Hence $\epsilon_p = \epsilon_r = 0$ in Proposition 2, even if its action distribution differs from the empirical weights of Ψ^*. In this outcome-compatible setting, consistent execution of the latent outcome, rather than recovery of a particular recorded action, is sufficient for transfer.

Explicit state conditioning is therefore not a requirement of the transfer argument when the decoder realizes the represented outcome on every visited supported pair. LAPIS learns a state-marginal decoder $\Psi(a \mid z)$ from scarce action labels. Table 5 compares it with a state-conditioned decoder using the same frozen 5%-label LAM across decoder-label budgets; within each budget, the remaining policy and training setup is shared between decoder variants. Marginal decoding achieves higher success across all three domains at 1% and 5% labels, while state conditioning performs better with full supervision. These results support marginal decoding for the scarce-label setting studied here.

D.5 Goal-conditioned objectives and outcome alignment

Theorem 2 requires physical controls that produce the same transition outcome to receive the same reward. The goal-conditioned objective used by LAPIS satisfies this condition under the deterministic dynamics of Assumption 1. Contrastive RL uses the goal-reaching reward

$$r_g(s,a) = (1-\gamma)p(s' = g \mid s, a).$$

Under Assumption 1, $s' = T(s,a)$, so

$$r_g(s,a) = (1-\gamma)\mathbf{1}[T(s,a) = g].$$

Consequently, if two physical actions produce the same successor,

$$T(s,a_1) = T(s,a_2) = s',$$

then

$$r_g(s,a_1) = r_g(s,a_2) = (1-\gamma)\mathbf{1}[s' = g].$$

Thus, in the deterministic setting analyzed in Section 5, the goal-conditioned reward depends on a physical control only through its transition outcome. Outcome-based abstraction therefore preserves the reward distinctions relevant to the goal-conditioned objective, providing the objective-alignment condition used in Theorem 2.

D.6 Representation horizon and receding-horizon control

The analysis in Section 5 is stated at the policy decision interval, where a latent action represents the successor of a single transition. In visual latent-action learning, however, the temporal gap used to construct the latent representation also controls how much transition information is visible to the encoder. We distinguish three quantities: the LAM temporal gap H in $z_t = I(s_t, s_{t+H})$; the physical-action chunk size C, the number of actions predicted by a real-action policy or a latent-action decoder; and the execution horizon E, the number of actions executed before replanning. The symbol H refers only to the LAM temporal gap and does not apply to real-action policies without a LAM.

Our default LAM uses a representation horizon of $H = 3$,

$$z_t = I(s_t, s_{t+3}),$$

but LAPIS remains a one-step receding-horizon controller: the decoder predicts a chunk of $C = 3$ actions, only the first action is executed ($E = 1$), and the policy replans from s_{t+1}. The critic likewise propagates value at this one-step interval using s_{t+1} and the recorded continuation latent $z_{t+1} = I(s_{t+1}, s_{t+4})$. Thus, the longer temporal gap changes the representation used for control, rather than requiring longer open-loop execution.

Representation and execution horizons. We empirically separate these two roles by varying the representation horizon and the number of decoded actions executed before replanning. Both $H = 1$ and our default $H = 3$ controller execute a single action before replanning. With this execution scheme, reducing the representation horizon from $H = 3$ to $H = 1$ sharply reduces task success across all three representative domains. Conversely, keeping the $H = 3$ representation but executing the full three-action chunk does not account for the benefit of the longer horizon: one-step replanning achieves higher mean performance in all three domains, with a particularly large difference on AntMaze.

Table 7: LAM temporal gap H and execution horizon E. Success rate (%) on representative tasks. The decoder chunk size is $C = H$ in these ablations; E is the number of decoded actions executed before replanning.

Setting	H	E	AntMaze-medium	Cube-single	Scene
One-step representation	1	1	4 ± 1	2 ± 1	0 ± 0
LAPIS	3	1	$\mathbf{49} \pm \mathbf{3}$	$\mathbf{62} \pm \mathbf{5}$	$\mathbf{29} \pm \mathbf{4}$
Full-chunk execution	3	3	9 ± 5	56 ± 3	26 ± 7

The control results are consistent with the representation diagnostics. With matched frozen LAMs, increasing the temporal gap from $H = 1$ to $H = 3$ raises held-out action-decodability R^2 from 0.501 to 0.597 on AntMaze, 0.586 to 0.778 on Cube, and 0.628 to 0.811 on Scene (Table 17). Adjacent visual observations can contain only small action-dependent changes, whereas a longer gap exposes more visible transition structure to the latent bottleneck. Together, the representation probes and control results indicate that $H = 3$ is useful primarily as a *representation horizon*, while one-step replanning remains effective for closed-loop execution.

Connection to the one-step analysis. When $H = 1$, the encoded endpoint coincides with the successor at the control interval, giving the direct latent–successor correspondence analyzed in Section 5. For $H > 1$, an endpoint latent can in principle summarize trajectories with different intermediate successors even though execution and value propagation remain one-step. The appropriate practical object is therefore the one-step successor distribution conditioned on the endpoint latent,

$$p^{(1)}_{\mathcal{B},H}(s^+ \mid s, z) := \Pr_{\mathcal{B}}\big(s_{t+1} = s^+ \mid s_t = s,\, I(s_t, s_{t+H}) = z\big)\,.$$

The corresponding physical process is the one-step transition induced by the first decoded action. Appendix D.3 provides the approximate-transfer framework for comparing these induced transition and reward processes; the exact result of Section 5 is recovered when the representation directly identifies the control-interval successor.

This distinction allows the analytical and practical roles of the temporal horizon to be separated cleanly: the one-step formulation isolates the conditions required for exact latent-to-physical correspondence, while the longer observation gap provides a substantially more informative representation for the visual domains studied here.

First-step execution fidelity. Although the $H = 3$ latent represents an endpoint three steps ahead, the deployed controller executes only the first decoded action before replanning. We therefore directly evaluate whether this longer-horizon representation preserves accurate one-step execution. On 100 held-out transitions per environment, we encode the corresponding latent, decode the action chunk, execute only its first action, and measure the resulting next-state cube-position error. The deployed $H = 3$ configuration achieves lower error than the $H = 1$ configuration on both Cube (1.51 vs. 3.26 mm) and Scene (0.80 vs. 1.22 mm). The same trend holds on transitions where the cube actually moves, with 3.59 vs. 7.69 mm on Cube and 4.20 vs. 6.42 mm on Scene (Table 8). These results

provide direct evidence that the practical $H = 3$ representation can preserve accurate first-step execution of the measured object transition despite encoding a longer-horizon endpoint, supporting its use with one-step receding-horizon control. This diagnostic compares existing configurations using held-out dataset latents and a geometric next-state metric; the configurations differ in temporal gap, decoder chunking, and training augmentation.

Table 8: First-step cube-position error (mm, ↓) on matched held-out transitions using seed-0 models ($C = H$, $E = 1$). Moving subsets have recorded cube displacement > 1 mm.

Environment	Subset	n	$H = 1$	$H = 3$
`v-cube-single-play`	All	100	3.26	**1.51**
	Moving	42	7.69	**3.59**
`v-scene-play`	All	100	1.22	**0.80**
	Moving	19	6.42	**4.20**

D.7 Learned value ordering and critic-guided reweighting

Proposition 1 formalizes policy improvement as reweighting a supported reference distribution toward higher-value latent actions. LAPIS implements this principle through the learned critic and the gate in Equation (4). For a fixed critic and threshold snapshot, let $q_{\mathcal{B}}(z \mid s, g)$ denote the conditional distribution of observed latent transitions presented to the actor, including its goal-relabeling procedure. Define the selected mass

$$Z_G(s, g) = \mathbb{E}_{z \sim q_{\mathcal{B}}(\cdot \mid s, g)} \left[G(s, z, g) \right],$$

and, whenever $Z_G(s, g) > 0$, the gated distribution

$$q_G(z \mid s, g) = \frac{q_{\mathcal{B}}(z \mid s, g) G(s, z, g)}{Z_G(s, g)}. \tag{8}$$

Because the gate only removes examples,

$$\operatorname{supp} q_G(\cdot \mid s, g) \subseteq \operatorname{supp} q_{\mathcal{B}}(\cdot \mid s, g) \subseteq Z_{\mathcal{B}}(s)$$

at sampled state–goal pairs. Thus, at the target-distribution level, critic-guided filtering is exactly a supported reweighting of observed state–latent transitions.

The improvement result does not require the critic score to be a calibrated numerical estimate of Q; it requires the score to preserve the ordering of the reference action values over z at fixed (s, g). For the ideal binary density-ratio objective in Remark 2,

$$f^\star(s, z, g) = \log Q^\mu(s, z, g) - \log p_{\mathcal{B}}(g),$$

and the second term is independent of z. Hence $f^\star$ preserves the ordering of Q^μ, and thresholding it at any z-independent state–goal threshold produces weights that are nondecreasing in value.

In the supported comparison of Appendix C.4, take the reference policy to be

$$\mu(\cdot \mid s, g) = q_{\mathcal{B}}(\cdot \mid s, g),$$

with $q_G = \mu$ when $Z_G(s, g) = 0$. Whenever the score preserves the ordering of

$$Q^\mu_{\mathcal{M}_{\mathcal{B}^z}}(s, \cdot, g),$$

Corollary 1 and Proposition 1 give

$$J_{\mathcal{M}_{\mathcal{B}^z}}(q_G; g) \geq J_{\mathcal{M}_{\mathcal{B}^z}}(q_{\mathcal{B}}; g).$$

Under the conditions of Theorem 1, this ordering transfers to batch-exact decoded execution, while Corollary 2 provides the corresponding approximate-transfer statement for learned decoding.

Practical LAPIS uses the TD-InfoNCE critic and the learned score-space threshold described in Appendix E.3 as a learned surrogate for this value ordering. The gate is applied only to observed

state–latent transitions, so for every fixed critic and threshold snapshot it defines an exact supported reweighting q_G of the actor-training distribution $q_{\mathcal{B}}$. During joint training, the critic and hence the reweighted target evolve, while the actor is trained toward the target defined by the current gate. The improvement result applies whenever the learned score preserves the corresponding reference-value ordering.

Consistent with this supported-update view, Table 4 shows that replacing direct max-Q actor optimization with critic-guided filtering improves performance for both deterministic and flow actors.

D.8 Flow-based extraction and guided inference

The remaining correspondence is between the supported target distribution defined by critic-guided filtering and the learned flow policy. For a fixed gate snapshot, define

$$\ell_\theta^{(o)}(s,z,g) = \mathbb{E}_{x_0,t}\left[\|v_\theta(x_t,t;s,g,o) - (z-x_0)\|_2^2\right], \qquad x_t = (1-t)x_0 + tz,$$

where noise and time are sampled as in Equation (5). Conditioning the actor objective on (s,g) gives

$$\begin{aligned}\mathcal{L}_{\text{actor}}(s,g) &= \mathbb{E}_{q_{\mathcal{B}}}\left[G\,\ell_\theta^{(1)}\right] + \lambda\mathbb{E}_{q_{\mathcal{B}}}\left[\ell_\theta^{(0)}\right] \\ &= Z_G(s,g)\mathbb{E}_{q_G}\left[\ell_\theta^{(1)}\right] + \lambda\mathbb{E}_{q_{\mathcal{B}}}\left[\ell_\theta^{(0)}\right].\end{aligned} \tag{9}$$

Thus, the high-value branch is trained toward q_G, while the behavior branch is trained toward $q_{\mathcal{B}}$. The full loss averages over sampled (s,g), with $Z_G(s,g)$ retaining the original weighting across state–goal contexts. When $Z_G(s,g) = 0$, the gated loss term vanishes; the reference-policy fallback in Appendix D.7 is used only to define the ideal policy comparison. Equation (9) therefore gives the direct correspondence between the practical flow-matching objective and the supported reweighting analyzed in Appendix C.4.

The support inclusion established in Appendix D.7 is exact for the target distributions $q_{\mathcal{B}}$ and q_G. A finite-capacity neural flow approximates these conditional distributions while sharing statistical strength across visual states. At inference, generated continuous latents are projected to the learned product codebook before decoding. This projection restricts samples to the learned code space, but does not by itself guarantee state-conditioned support; that depends on how faithfully the learned policy represents its conditional target.

Classifier-free guidance combines the two learned velocity fields,

$$v_w^{\text{CFG}} = (1-w)v_\theta(\cdot;0) + wv_\theta(\cdot;1).$$

At $w = 0$, inference uses the branch trained toward $q_{\mathcal{B}}$, while $w = 1$ uses the branch trained toward q_G. Values $w > 1$ extrapolate the velocity field toward the high-value branch and therefore need not preserve the exact support guarantee of q_G. This extrapolation differs from direct max-Q actor optimization, however. Direct max-Q updates the selected latent by following critic values in the continuous latent space, including regions where the critic is not grounded by observed transitions. In LAPIS, the critic is used first to select high-value examples only among observed state–latent transitions, producing $q_G \subseteq q_{\mathcal{B}}$; guidance then extrapolates between policy fields trained on $q_{\mathcal{B}}$ and this supported high-value reweighting. The guidance direction is therefore anchored to distributions constructed entirely from observed state–latent transitions, although sufficiently strong extrapolation can still move generated samples away from their exact support.

The supported-reweighting result thus identifies q_G at $w = 1$ as the theoretical anchor, while $w > 1$ provides an inference-time extension that trades stricter adherence to this target for stronger emphasis on the high-value branch. The guidance scale w is distinct from the distributional exponent η in Proposition 1, and the supported-improvement guarantee does not by itself extend to arbitrary $w > 1$.

The experiments provide complementary evidence for these stages. Table 4 shows that the flow actor improves policy extraction when paired with critic-guided filtering. Table 13 further separates the supported high-value branch from guidance extrapolation. Across the visual manipulation tasks, $w = 1$, which uses the branch trained toward q_G, already captures most of the improvement from the behavior branch $w = 0$ to the default $w = 3$ policy. Stronger guidance therefore provides an additional empirical refinement rather than being solely responsible for the gains of LAPIS.

Section 7.3 further compares the effect of stronger direct max-Q optimization and stronger guidance through behavior-support distance d_{BC}, realization round-trip error d_{round}, and task success. As CRL becomes more aggressive, its support distance and realization error increase while success collapses, whereas LAPIS remains closer to the behavior-supported region over the guidance sweep. These results show that guided inference stays closer to the behavior-supported region than unconstrained critic maximization in this comparison.

The direct CFGRL comparison in Appendix E.5 further illustrates why this distinction matters in a learned latent action space. Goal-conditioned CFGRL benefits from increasing guidance when applied to physical actions, most clearly on AntMaze, but provides no consistent improvement when the same construction is applied directly to learned latent actions. In contrast, LAPIS obtains large gains already at $w = 1$, before extrapolative guidance, with its high-value branch trained toward the critic-filtered supported target q_G rather than the unfiltered goal-conditioned behavior distribution. Together with the component ablation in Table 4, this provides empirical evidence that support-aware target construction, rather than guidance alone, is consequential for latent policy improvement.

E EXPERIMENTAL DETAILS AND ADDITIONAL ANALYSES

This appendix provides the experimental setup and supplementary analyses. We describe the benchmarks and evaluation protocols (Appendix E.1), latent-action representation learning and decoding (E.2), LAPIS critic and policy learning (E.3), and baseline implementations (E.4). We then report additional sampling, control, and guidance analyses (E.5), analyze action decodability and temporal-gap effects in the learned latent representation (E.6), and present qualitative rollouts (E.7).

E.1 DATASETS AND EVALUATION

Benchmarks and label regimes. We use the OGBench (Park et al., 2025) visual `cube`, `scene`, and `antmaze-navigate` domains with 64×64 RGB observations. Manipulation actions have five dimensions (end-effector position change, yaw change, and gripper control), while antmaze actions are eight-dimensional joint torques. For the ablations in Tables 4–5, we use one representative task from each family: `visual-cube-single-play`, `visual-scene-play`, and `visual-antmaze-medium-navigate`. The diagnostics in Figure 3 focus on `visual-antmaze-medium-navigate`. Latent-action methods use all observed transitions for LAM and latent-policy learning, while the data used by the action-scarce baselines are specified in Appendix E.4.

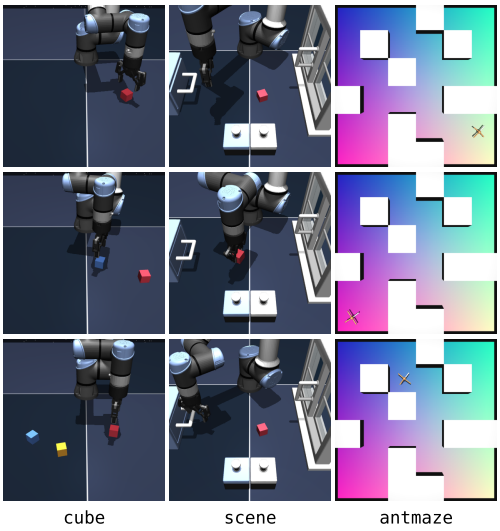

Figure 5: Example observations, rendered from our own environment instances, spanning the `cube` and `scene` manipulation families and the `antmaze` navigation family.

OGBench evaluation. Each of four seeds is evaluated on 20 episodes per task at the 300k, 400k, and 500k checkpoints. We average these checkpoint scores within each seed, then report the mean and standard deviation across seeds. The fully supervised GCBC values are taken from Park et al. (2025) with their original evaluation protocol.

Visual Franka Kitchen. Kitchen uses 64×64 rendered RGB observations, nine-dimensional bounded actions, and the original D4RL simulator with a 280-step episode limit. The `partial` variant targets microwave, kettle, light switch, and slide cabinet; `mixed` targets microwave, kettle, bottom burner, and light switch. Goal images are rendered with the evaluation camera after setting the required objects to their target configurations and the robot to its initial posture. Completed subtasks are removed from the remaining-task set, and an episode terminates when all required subtasks are completed or the time limit is reached.

Table 2 reports the percentage of required subtasks completed, averaged over evaluation episodes:

$$\bar{C} = \frac{1}{N_{\text{eval}}} \sum_{j=1}^{N_{\text{eval}}} 100 \frac{n_{c,j}}{N_j},$$

where $n_{c,j}$ of the $N_j = 4$ required subtasks are completed in episode j. All latent policies use the canonical `partial` training split and are evaluated against both target-task sets; `mixed` does not denote a separately trained latent policy. Each seed uses its own frozen LAM and decoder trained for 500k updates, with the $H = 3$, $C = 3$, $E = 1$ receding-horizon setup described in Appendix D.6. For seeds 0–3, we average the 300k, 400k, and 500k policy-checkpoint scores (20 episodes per target-task set per checkpoint), then report the mean and population standard deviation across the four seed averages. Table 2 reports LAPIS at guidance $w = 3$; the latent LAPIS run uses a TD-InfoNCE critic and a log-sum-exp value baseline. The real-action oracle results are reported separately from the latent policies.

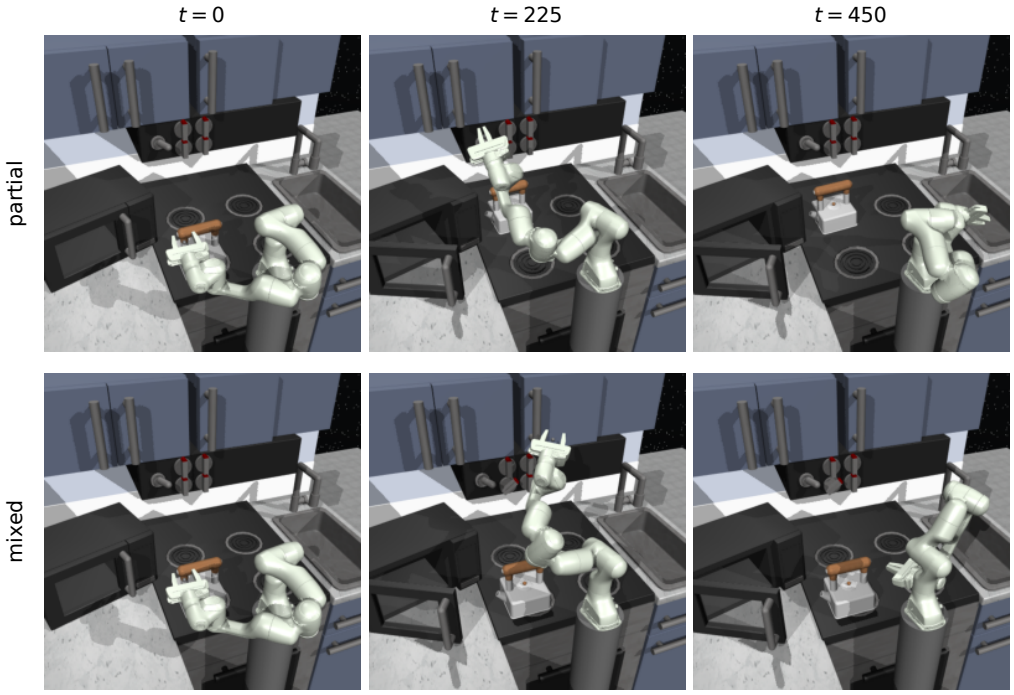

Figure 6: Example observations from visual Franka Kitchen. Rows show `partial` (top) and `mixed` (bottom); columns show steps $t = 0, 225, 450$ of illustrative LAPIS rollouts with guidance $w = 3$ and reset seed 0. The displayed simulator states are rendered at 512×512 using the evaluation camera, while policy observations remain 64×64. These illustrative 450-step rollouts are separate from the 280-step D4RL evaluations reported in Table 2.

E.2 Latent action model and decoder

Representation learning. The IDM/FDM pair is a product-quantized transition autoencoder trained by pixel reconstruction with independent cross-crop inputs. It uses single-frame observations, temporal gap $H = 3$, and IDM encoder scale 4. The codebook configuration is $M{:}H_z{:}D{:}K = 2{:}4{:}8{:}128$: two codebooks, four slots per codebook, eight embedding coordinates, and 128 entries per codebook, giving 64 flattened latent coordinates. We denote the continuous encoder output by z_e and its quantization by z_q.

Latent coordinates and codebook projection. The policy, critic, and decoder use quantized latents standardized per codebook and embedding dimension:

$$[\mathcal{N}(z_q)]_{mhd} = \frac{(z_q)_{mhd} - \mu_{md}}{\sigma_{md}}, \tag{10}$$

where the frozen means μ_{md} and stabilized standard deviations σ_{md} are shared across slots h. The same transform is applied to the codebook entries. Generated continuous latents are projected to the nearest product-codebook entries in these coordinates before decoding.

Action decoding. The experimental state-marginal flow decoder $\Psi(\mathbf{a} \mid z)$ is trained on labeled examples to predict three-action chunks from standardized quantized latents and is frozen for deployment. Predicted actions are clipped to the environment bounds and executed with the receding-horizon procedure of Appendix D.6. The state-conditioned variant in Table 5 additionally takes s as input. For Table 5, we keep the same frozen 5%-label LAM across all decoder-label budgets and vary only decoder supervision.

E.3 Critic and policy learning

Shared policy-learning settings. Table 9 consolidates the optimization, architecture, critic, flow-policy, and evaluation hyperparameters used by LAPIS on OGBench. In Table 1, the physical-action chunk sizes C for F-GCBC, GCIQL, CRL, and LAPIS are $(3, 1, 1, 3)$, respectively, while their latent-action counterparts use $C = 3$. All latent-action methods use the $H = 3$ LAM of Appendix E.2, and all settings execute only the first physical action before replanning ($E = 1$; Appendix D.6).

Table 9: LAPIS hyperparameters for OGBench policy learning.

Setting	Visual manipulation	Visual antmaze
Adam learning rate	3×10^{-4}	3×10^{-4}
Batch size	256	256
Gradient steps	500k	500k
Visual encoder	small IMPALA	small IMPALA
Policy MLP trunk	$(512, 512, 512)$	$(512, 512, 512)$
Layer normalization	yes	yes
p_{aug} (random crop)	0.5	0.0
Discount γ	0.99	0.99 (0.995 giant)
LAPIS critic	TD-InfoNCE	TD-InfoNCE
Critic ensemble size	2	2
Critic feature dimension	512	512
Critic temperature τ_{rep}	0.1	0.1
Gate expectile τ	0.9	0.9
Flow time-embedding dimension	64	64
Flow training-time distribution	$\text{Beta}(1.5, 1.0)$	$\text{Beta}(1.5, 1.0)$
Maximum training time $t_{\max}$	0.999	0.999
Flow sampling steps	10 Euler	10 Euler
Behavior-branch weight λ	0.1	0.1
Evaluation guidance w	3	3
Evaluation episodes	20	20

Categorical TD objective. LAPIS uses categorical TD-InfoNCE on both visual manipulation and antmaze. For a minibatch of n recorded transitions, let $s_i^+ = s_{t_i+1}$ and let z_i^+ be the LAM label inferred from its recorded continuation. Critic candidate goals are sampled from a mixture of current states, future states in recorded trajectories, and random dataset states. With ensemble index $e \in \{1, 2\}$, define

$$L^{\text{next}}_{ije} = f_{\phi,e}(s_i, z_i, s_j^+), \qquad L^{\text{goal}}_{ije} = f_{\phi,e}(s_i, z_i, g_j),$$
$$W_{ij} = \text{sg}\Big[\text{softmax}_j\Big(\min_e f_{\bar{\phi},e}(s_i^+, z_i^+, g_j)\Big)\Big],$$

where sg denotes stop-gradient and $\bar{\phi}$ is the target critic. For $\text{CE}(\ell, y) = -\sum_j y_j \log \text{softmax}_j(\ell)$ and $I_{ij} = \mathbf{1}[i = j]$, the loss is

$$\mathcal{L}_{\text{TD}} = \frac{1}{2n} \sum_{i,e} \Big[(1 - \gamma)\, \text{CE}(L^{\text{next}}_{i,:,e}, I_{i,:}) + \gamma\, \text{CE}(L^{\text{goal}}_{i,:,e}, W_{i,:})\Big]. \tag{11}$$

The first term predicts immediate successors; the second propagates future-state occupancy through behavior continuations using a detached categorical bootstrap (Zheng et al., 2024). Appendix E.5 reports the goal-candidate sampling ablation.

Score-space gate. The score f_ϕ in Equation (4) is the ensemble minimum $f_{\min}(s, z, g) = \min_e f_{\phi,e}(s, z, g)$. For actor-training goals g, we fit the state–goal threshold by expectile regression,

$$\mathcal{L}_{\hat{V}} = \mathbb{E}\left[\rho_\tau\Big(\mathrm{sg}[f_{\min}(s, z, g)] - \hat{V}(s, g)\Big)\right], \qquad \rho_\tau(u) = |\tau - \mathbf{1}[u < 0]|u^2, \tag{12}$$

with the expectile τ specified in Table 9. The gate is $\mathrm{sg}[\mathbf{1}[f_{\min} > \hat{V}]]$. The critic, score threshold, and actor are updated jointly as in Algorithm 1.

Critic features and flow policy. The bilinear critic uses L_2-normalized features with temperature τ_{rep}:

$$f_{\phi,e}(s, z, g) = \frac{\langle \phi_e(s, z)/\|\phi_e(s, z)\|_2,\ \psi_e(g)/\|\psi_e(g)\|_2 \rangle}{\tau_{\mathrm{rep}}}.$$

The critic dimensions and all flow-training and guided-inference hyperparameters are summarized in Table 9.

E.4 Baseline implementations

Table 1 baselines. The fully supervised GCBC entries are taken verbatim from Park et al. (2025). F-GCBC replaces the Gaussian GCBC head with our flow-matching actor and is not an OGBench baseline. GCIQL in Table 1 follows standard goal-conditioned IQL, learning an action-conditioned critic $Q(s, a, g)$ and value function $V(s, g)$ and extracting the actor from their advantage. CRL follows the upstream OGBench implementation with a DDPG+BC actor and InfoMax/InfoNCE critic, with hyperparameters tuned per domain. The domain-specific BC coefficient is reported in Table 10. F-GCBC, CRL, and LAPIS use the shared visual encoder, batch size, and augmentation settings within a domain. The method-specific action-chunk sizes used in Table 1 are summarized in Appendix E.3, and their sensitivity is examined in Table 12.

Table 10: CRL-specific OGBench hyperparameter.

Setting	Visual manipulation	Visual antmaze
BC coefficient α	3.0	0.1

Action-scarce baselines. Table 3 uses goal-conditioned physical-action adaptations designed for limited action supervision. In particular, its GCIQL entry is distinct from the standard action-conditioned GCIQL used in Table 1; Appendix F.2 discusses this protocol difference. Let $\mathcal{D}_L$ contain genuinely labeled transitions and $\mathcal{B}$ the action-free transitions, with goal-reaching reward $r(s, g)$. We use the following value-learning, labeled-action, and inverse-prediction objectives:

$$\begin{aligned}
\mathcal{L}_V &= \mathbb{E}_{s,s',g}\left[L_2^\tau\big(r(s, g) + \gamma \bar{V}(s', g) - V(s, g)\big)\right], \\
L_2^\tau(u) &= |\tau - \mathbf{1}[u < 0]|u^2, \\
\mathcal{L}_\pi^{\mathrm{AWR}} &= \mathbb{E}_{(s,a,g)\sim\mathcal{D}_L}\left[-\exp(\alpha A)\log \pi(a \mid s, g)\right], \\
\mathcal{L}_{\mathrm{IDM}} &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}_L}\|I_a(s, s') - a\|_2^2, \qquad \hat{a} = I_a(s, s') \text{ on } \mathcal{B}, \\
\mathcal{L}_{\mathrm{BC}} &= \mathbb{E}[-\log \pi(a \mid s, g)],
\end{aligned}$$

where $\bar{V}$ is the target value and τ is the upper expectile. The methods combine these objectives as follows:

- **GCIQL** (Kostrikov et al., 2022): $\mathcal{L}_V$ plus a flat AWR actor with the action-free advantage $A = V(s', g) - V(s, g)$.
- **HIQL** (Park et al., 2023): $\mathcal{L}_V$ plus two AWR actors—a high-level actor over subgoals ($A^h = V(s_{t+k}, g) - V(s, g)$) and a low-level actor over actions toward the subgoal ($A^\ell = V(s', s_{t+k}) - V(s, s_{t+k})$, on $\mathcal{D}_L$).
- **VPT-RL (VPT-style action inference)** (Baker et al., 2022): train I_a via $\mathcal{L}_{\mathrm{IDM}}$, pseudo-label $\mathcal{B}$, then extract a value-guided (classifier-free-guided flow) RL policy from the (pseudo-)labeled real actions—the LAPIS policy objective of Equation (5) applied to actions instead of latents.

- **SS-ORL** (Zheng et al., 2023): offline RL ($\mathcal{L}_V$ + AWR actor) on the union $\mathcal{D}_L \cup \{(s, \hat{a}, s') : (s, s') \in \mathcal{B}\}$ of true and IDM-pseudo-labeled actions.

Action-free value objectives use all transitions. GCIQL's physical-action actor and HIQL's low-level actor use genuine labels; HIQL's subgoal learning also uses action-free trajectories. The inverse-action predictor is fitted on genuine labels, and pseudo-label-based policy objectives use the union of genuine and inferred actions. Architecture and optimization settings follow the shared recipe where applicable.

E.5 Additional control and sampling ablations

Goal-candidate sampling. Because the LAM represents $H = 3$ endpoint transitions, we test whether future-state candidates for critic training should also be restricted to the same temporal grid. We compare the standard sampling scheme, which allows arbitrary future offsets, with a grid-restricted variant that rounds each trajectory-future offset k to $H \max(1, \mathrm{round}(k/H))$, with $H = 3$. Current-state and random-state candidates, actor-goal sampling, the next-state positive, and one-step bootstrapping are unchanged. The ablation therefore changes only the temporal sampling of future candidates in Equation (11). Table 11 uses the standard LAPIS reference from Table 1. Standard sampling achieves higher success than grid-restricted sampling on cube (62 vs. 53), scene (29 vs. 23), and antmaze (49 vs. 34), supporting future-goal candidates without a temporal-grid restriction.

Table 11: Goal-candidate sampling ablation for LAPIS (success rate %). Only the trajectory-future candidate component is grid-restricted; the standard reference is from Table 1.

Task	Standard candidates	Grid-restricted candidates
`v-cube-single-play`	62 ±5	53 ±4
`v-scene-play`	29 ±4	23 ±3
`v-antmaze-medium-navigate`	49 ±3	34 ±7

Action chunk size in real-action policies. Table 12 compares action chunk sizes $C = 1$ and $C = 3$ on three representative tasks, with one action executed before replanning in both settings. This ablation helps separate the policy-improvement mechanism from the method-specific action chunking used in the fully supervised block of Table 1.

Table 12: Success rate (%) of real-action policies with 100% action labels (mean ± standard deviation over four seeds). Results for $C = 1$ and LAPIS are reproduced from Table 1. New $C = 3$ results average the 300k, 400k, and 500k checkpoints within each seed.

Method	C	AntMaze-medium	Cube-single	Scene
GCIQL	1	11 ±1	30 ±5	12 ±2
GCIQL	3	19 ±2	24 ±4	3 ±1
CRL	1	95 ±2	18 ±10	13 ±3
CRL	3	95 ±1	23 ±10	13 ±1
LAPIS	3	48 ±3	45 ±4	45 ±1

Guidance scale. Table 13 evaluates LAPIS at $w \in \{0, 1, 3\}$. Here $w = 0$ uses the behavior branch, $w = 1$ uses the high-value branch trained toward the supported target q_G, and $w = 3$ further extrapolates in the high-value direction. On all visual manipulation tasks, most of the improvement from $w = 0$ to the default $w = 3$ is already obtained at $w = 1$. For example, success increases from 13 to 53 to 62 on cube-single, from 13 to 29 to 31 on cube-double, and from 18 to 25 to 29 on scene. Thus, stronger guidance provides additional gains in some tasks, but the supported high-value branch itself accounts for the dominant improvement in these manipulation domains.

Table 13: Success rate (%) of latent-action LAPIS with 5% action labels at different guidance scales.

Task	$w = 0$	$w = 1$	$w = 3$
v-antmaze-medium-navigate	30 ±6	51 ±5	49 ±3
v-antmaze-large-navigate	7 ±2	8 ±2	10 ±2
v-antmaze-giant-navigate	0 ±0	0 ±0	0 ±0
v-antmaze-teleport-navigate	14 ±4	16 ±4	21 ±3
v-cube-single-play	13 ±1	53 ±3	62 ±5
v-cube-double-play	13 ±1	29 ±4	31 ±6
v-cube-triple-play	18 ±1	20 ±1	20 ±0
v-cube-quadruple-play	10 ±4	13 ±3	14 ±3
v-scene-play	18 ±3	25 ±4	29 ±4

Table 14: **Goal-conditioned CFGRL in physical and learned latent action spaces.** Success rate (%). The comparison focuses on the effect of increasing guidance within each action representation.

Task	Action space	$w = 0$	$w = 1$	$w = 3$
v-antmaze-medium-navigate	Real	7.2 ±1.9	15.0 ±1.0	24.1 ±4.0
	Latent	6.7 ±1.5	5.0 ±0.7	6.1 ±0.9
v-cube-single-play	Real	0.7 ±0.2	1.5 ±1.7	4.0 ±5.8
	Latent	0.5 ±0.5	0.6 ±0.3	0.2 ±0.2
v-scene-play	Real	2.8 ±0.5	4.4 ±1.0	3.9 ±1.0
	Latent	1.8 ±0.5	1.6 ±0.8	1.7 ±0.9

CFGRL in physical and learned latent action spaces. Because LAPIS adopts classifier-free guided policy extraction, we additionally compare with the value-free goal-conditioned formulation of CFGRL (Frans et al., 2025). In this setting, standard flow-based GCBC corresponds to $w = 1$, while $w > 1$ strengthens goal guidance at inference without learning an explicit value function. We apply the same CFGRL construction both to physical actions and directly to learned latent actions, and sweep $w \in \{0, 1, 3\}$. In the latent-action setting, CFGRL uses the same latent action model and decoder as LAPIS. Table 14 reports checkpoint-averaged success rates.

Increasing guidance improves physical-action CFGRL most clearly on AntMaze, where success rises from 7.2 at $w = 0$ to 24.1 at $w = 3$. In contrast, directly applying the same goal-conditioned guidance mechanism in the learned latent space produces no consistent improvement across the three domains: performance remains nearly unchanged, or decreases, as w increases. Thus, the gains associated with classifier-free guidance in an executable physical action space do not automatically transfer when the guided variable is a learned latent action.

This difference is consistent with the cross-representation issue studied in Section 5. CFGRL's goal-conditioned construction treats the generated variable as the action whose future-goal occupancy defines the improvement signal. For a physical action, this action is executed directly. A learned latent action, however, is an outcome label inferred from an observed transition and must subsequently be decoded into physical control. Improving a distribution over latent actions therefore does not by itself guarantee improvement after decoding: the selected latent must be appropriate for the current state and its represented transition must be faithfully realized by the decoder. These are precisely the additional correspondences captured by supported outcome identifiability and latent-to-physical execution in Theorem 1.

LAPIS addresses this issue before guided inference. Rather than using guidance itself to construct an improved latent policy, LAPIS first scores only observed state–latent transitions and forms the critic-filtered distribution $q_G(z \mid s, g)$, whose support is contained in the observed conditional distribution $q_{\mathcal{B}}(z \mid s, g)$. The high-value flow branch is then trained toward q_G, and classifier-free guidance is applied only after this support-aware target has been constructed. This distinction is already visible without extrapolative guidance: at $w = 1$, latent CFGRL remains weak on AntMaze-

medium, Cube-single, and Scene, whereas the LAPIS high-value branch reaches 51%, 53%, and 25%, respectively (Table 13). Thus, the main difference is not the use of classifier-free guidance itself, but the construction of the latent policy-improvement target on observed state–latent transitions.

E.6 Action decodability of the latent representation

Probe protocol. We fit a two-hidden-layer MLP (256 units per layer, ReLU activations) from the frozen, unquantized latent z_e to the current action a_t. For each OGBench task, we use independently trained $H = 3$ LAMs at 500k training steps. The MLP is trained on the labeled 5% of training trajectories. For OGBench, probe validation and test trajectories are disjoint subsets of the official validation dataset. For D4RL Kitchen, we use four independently trained partial-data LAMs (seeds 0–3) at 500k steps with $H = 3$, and split held-out rendered episodes into disjoint probe validation and test sets. Latent standardization uses probe training data only; we select the MLP by validation MSE over at most 200 epochs and report test R^2 in Tables 15 and 16.

For action coordinate d, let $\mathrm{SSE}_d = \sum_i (a_{id} - \hat{a}_{id})^2$ and $\mathrm{SST}_d = \sum_i (a_{id} - \bar{a}_d)^2$ on the test set. We report $R^2_d = 1 - \mathrm{SSE}_d/\mathrm{SST}_d$ and the variance-weighted aggregate

$$R^2_{\mathrm{Overall}} = 1 - \frac{\sum_d \mathrm{SSE}_d}{\sum_d \mathrm{SST}_d}.$$

For Tables 15–17, we first average the three probe runs within each LAM seed. Per-coordinate entries are then averaged across LAM seeds, while Overall reports the mean ± sample standard deviation across LAM seeds.

Table 15: MLP action-decodability R^2 on held-out OGBench trajectories ($z_e \to a_t$, $H = 3$). Overall is the variance-weighted R^2. Cube/scene actions have five coordinates; antmaze actions have eight.

Task	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Overall R^2
v-antmaze-medium-navigate	0.68	0.47	0.61	0.62	0.70	0.49	0.62	0.49	0.597 ±0.001
v-antmaze-large-navigate	0.69	0.46	0.60	0.61	0.70	0.42	0.64	0.53	0.596 ±0.001
v-antmaze-giant-navigate	0.71	0.50	0.63	0.59	0.68	0.45	0.58	0.47	0.588 ±0.003
v-antmaze-teleport-navigate	0.65	0.50	0.59	0.61	0.63	0.41	0.63	0.48	0.575 ±0.002
v-cube-single-play	0.83	0.92	0.96	0.03	0.83	–	–	–	0.778 ±0.000
v-cube-double-play	0.80	0.91	0.95	−0.10	0.81	–	–	–	0.740 ±0.001
v-cube-triple-play	0.84	0.94	0.97	−0.01	0.83	–	–	–	0.773 ±0.000
v-cube-quadruple-play	0.85	0.94	0.97	0.19	0.84	–	–	–	0.806 ±0.001
v-scene-play	0.90	0.92	0.94	0.57	0.79	–	–	–	0.811 ±0.000

Table 16: MLP action-decodability R^2 on held-out D4RL Kitchen trajectories ($z_e \to a_t$, $H = 3$). Overall is the variance-weighted R^2.

Task	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	Overall R^2
D4RL Kitchen	0.91	0.46	0.43	0.81	0.50	0.45	0.53	0.38	0.33	0.660 ±0.008

Temporal gap and action decodability. We compare $H = 1$ and $H = 3$ representations using matched samples and frozen 500k LAMs. Both probes predict the same current action a_t from the unquantized latent z_e using the MLP protocol above. Table 17 reports variance-weighted test R^2. The $H = 3$ representation gives higher action decodability in all three domains. A plausible explanation is that a longer observation gap exposes more visible action-dependent changes, making the transition signal easier to retain under visual augmentation and quantization. These probes quantify the action information recoverable from the learned representations.

Table 17: Temporal gap and action decodability. Variance-weighted test R^2 for an MLP predicting a_t from z_e, using matched samples and frozen 500k LAMs.

Environment	$H = 1$	$H = 3$
`v-antmaze-medium-navigate`	0.501 ± 0.002	0.597 ± 0.001
`v-cube-single-play`	0.586 ± 0.001	0.778 ± 0.000
`v-scene-play`	0.628 ± 0.001	0.811 ± 0.000

E.7 Qualitative rollouts

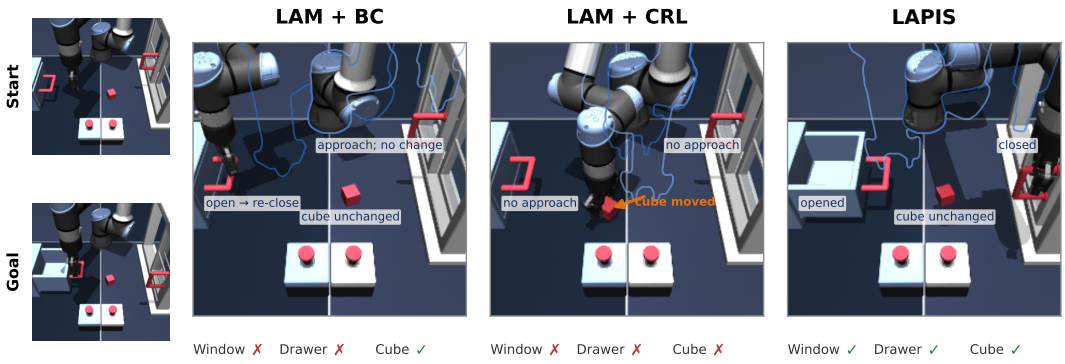

Figure 7: Qualitative rollouts on `scene` (5% labels), shared start and goal.

Figure 7 compares rollouts from a shared start state with the goal of closing the window, opening the drawer, and keeping the cube unchanged. Each panel overlays the gripper trajectory and two interaction frames on the final state, with per-subtask outcomes below. Recorded simulator positions confirm that LAM+BC opens then re-closes the drawer and leaves the window unchanged; LAM+CRL neither opens the drawer nor closes the window and moves the cube off-target; LAPIS completes all three subtasks.

F Additional Discussion of Experimental Settings

This appendix provides additional context for several experimental comparisons whose interpretation involves differences in policy-improvement mechanisms, supervision protocols, decoder training, or environment dynamics. These details are complementary to the implementation specifications in Appendix E and the theory–practice correspondence in Appendix D.

F.1 Fully supervised AntMaze performance

Table 1 shows a substantial gap between fully supervised real-action CRL and real-action LAPIS on AntMaze, particularly on the larger mazes. These results highlight a tradeoff of the support-aware update used by LAPIS under full physical-action supervision.

LAPIS is motivated by a failure specific to learned latent-action spaces: direct value maximization can select latent actions whose represented transitions are not physically realizable after decoding. The latent realizability failure underlying this mismatch is absent when the policy directly outputs physical actions, although ordinary offline-RL distribution shift remains. In this setting, restricting policy improvement toward behavior-supported transitions can be unnecessarily conservative, while more aggressive critic-driven optimization may be beneficial. Conversely, in the learned latent-action setting, direct maximization can move the policy toward unrealizable latents, motivating the supported update used by LAPIS.

The larger gap on more difficult AntMaze tasks is therefore consistent with a cost of conservative policy extraction in long-horizon navigation. Accordingly, the fully supervised results provide physical-action references for the limited-supervision comparisons. Our main question is whether effective policy improvement can be retained after replacing physical actions with learned latent actions under limited action supervision.

F.2 Protocol differences across baseline tables

The GCIQL entries in Tables 1 and 3 differ in their value-learning and policy-extraction objectives. Table 1 uses an action-conditioned critic and a state–goal value function, with policy extraction based on the $Q - V$ advantage. This formulation is applied to physical actions at the stated label budgets and to learned latent actions in the LAM setting.

Table 3 uses an action-scarce adaptation that learns a state–goal value function from all observed transitions, without requiring action labels for value learning. Its physical-action policy is trained only on labeled transitions, weighted by the value difference between successive states. The two tables therefore compare different GCIQL formulations, rather than repeated evaluations of the same procedure.

F.3 Interpreting decoder label-budget trends

Table 5 isolates decoder supervision from latent representation learning: all label budgets use the same frozen LAM trained with 5% action labels, while only the amount of action supervision available to the decoder is varied. Thus, the non-monotonic performance of the state-marginal decoder across 1%, 5%, and 100% decoder labels is not caused by changes in the learned latent representation.

The state-marginal decoder $\Psi(a \mid z)$ aggregates physical realizations across states without observing the current state. If the same latent code admits different appropriate controls at different states, broader coverage can make the empirical conditional $p(a \mid z)$ more heterogeneous, partially offsetting the benefit of additional labeled examples. By contrast, the state-conditioned decoder $\Psi(a \mid s, z)$ can use the current state to resolve such variation when sufficient supervision is available.

This provides a plausible interpretation of the pattern in Table 5: the marginal decoder performs better under 1% and 5% supervision, where sharing action information across states can be beneficial, while the state-conditioned decoder performs better with full supervision. The fact that the marginal cube score at 5% exceeds its 100%-label score should therefore not be interpreted as evidence that fewer labels are intrinsically preferable; rather, performance reflects the interaction between decoder conditioning, labeled-data coverage, and finite-sample optimization.

F.4 STOCHASTIC TELEPORT DYNAMICS

The visual-antmaze-teleport benchmark provides a concrete case in which the deterministic-dynamics assumption used for the exact transfer result does not hold globally. In the OGBench teleport environment, entering an inbound teleport region sends the agent to a randomly selected outbound portal. Thus, the same physical state and control can induce different successor states at a teleport transition. This violates Assumption 1 and means that the exact equality in Theorem 1 is not expected to apply at such transitions. The stochastic teleport implementation can be verified directly in OGBench.

This case is instead described by the approximate-transfer analysis in Appendix D.3. For a decoded latent policy, Proposition 2 bounds the return discrepancy by

$$\left|J_{\mathcal{M}}(\widehat{\Psi} \circ \pi_z) - J_{\mathcal{M}_{\mathcal{B}^z}}(\pi_z)\right| \leq \frac{\bar{\epsilon}_r}{1-\gamma} + \frac{2\gamma R_{\max}}{(1-\gamma)^2}\bar{\epsilon}_p,$$

where $\bar{\epsilon}_p$ measures the transition mismatch between latent evaluation and decoded physical execution and $\bar{\epsilon}_r$ measures the corresponding reward mismatch. Random teleportation can increase ϵ_p because a latent transition inferred from one observed teleport outcome need not reproduce that same outcome when the decoded action is executed again. The exact deterministic result is recovered when these mismatches vanish.

The teleport task evaluates performance under stochastic transitions, for which the transition- and reward-mismatch framework in Appendix D.3 provides the relevant theoretical perspective.

F.5 STANDALONE F-GCBC AND THE LAPIS BEHAVIOR BRANCH

Table 1 reports F-GCBC as a standalone flow-based goal-conditioned behavioral-cloning baseline, whereas $w = 0$ in Table 13 evaluates the behavior-conditioned branch of a jointly trained LAPIS actor. Although both are trained from observed latent transitions using flow matching, they are not the same trained policy. LAPIS uses a single conditional velocity model $v_\theta(\cdot; o)$ whose $o = 0$ branch is trained toward the observed distribution $q_{\mathcal{B}}$ and whose $o = 1$ branch is simultaneously trained toward the critic-filtered distribution q_G through Equation (5). The two losses therefore update shared model parameters.

Consequently, setting $w = 0$ at inference selects the behavior-conditioned velocity field of the jointly trained two-branch model. Table 13 is an inference-time guidance ablation within a fixed LAPIS actor, whereas Table 1 evaluates a separately trained F-GCBC policy. Their different training objectives and parameter sharing explain the difference between their scores.

G Future Work and Limitations

G.1 Future Work

A promising direction is to extend LAPIS from goal-conditioned to language-conditioned control. Prior work has shown that language instructions can be grounded into visual goal representations and executed through goal-conditioned policies (Myers et al., 2023; Cachet et al., 2024). This suggests that LAPIS could retain its latent-space policy-improvement framework while replacing explicit visual goals with language-derived goal representations, enabling policy improvement for instruction-following and vision-language-action settings. Extending the framework and its transfer analysis to partially observed settings using temporal or recurrent state representations is another important direction.

G.2 Limitations

Our experiments are limited to simulated visual manipulation and navigation benchmarks, and we focus on goal-conditioned control with limited physical-action supervision. Consequently, we do not empirically study objectives with action-dependent rewards, where outcome-based abstraction can create a mismatch between latent and physical policy objectives. Real-robot deployment and language-conditioned control also remain outside the scope of the current evaluation.

Our exact transfer and alignment results rely on deterministic dynamics, supported outcome identifiability, batch-constrained execution through the batch-exact decoder, and Markov state information. Practical LAPIS uses learned visual state representations, a learned state-marginal decoder, an $H = 3$ latent representation with one-step replanning, and classifier-free guidance, so these conditions are not instantiated exactly. We explicitly study these departures in Appendix D and through targeted empirical analyses. The approximate-transfer analysis accounts for transition and decoder mismatch, while decoder-conditioning, temporal-horizon, and first-step execution analyses examine learned decoding and the $H > 1$ receding-horizon controller. For guided inference, the $w = 1$ high-value branch provides the supported target underlying our analysis, and the guidance sweep shows that it already captures most of the improvement on the manipulation tasks; $w > 1$ provides an additional inference-time extrapolation beyond this exact support guarantee. The direct CFGRL comparison further shows that guidance alone does not provide consistent improvement when applied directly in the learned latent space, reinforcing the role of support-aware target construction in practical LAPIS.