ZERO-SHOT CONTROL WITH GAUSSIAN REPRESENTATIONS AND TEST-TIME ADJUSTMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-Shot Reinforcement Learning (ZSRL) pretrains general-purpose policies from reward-free experience to enable deployment on unseen rewards without further learning. Successor Feature (SF)-based methods facilitate this transfer by fitting a linear reward coefficient to retrieve a pretrained policy. However, existing approaches often overlook how representation geometry affects SF evaluation, while direct policy retrieval can be suboptimal under reward misspecification and function-approximation errors. First, we introduce Gaussian Latent Dynamics-based Successor Features (GLDSF), which combines latent dynamics prediction with regularization toward an isotropic Gaussian to learn control-relevant representations with well-conditioned geometry for SF evaluation. To mitigate retrieval suboptimality, we propose a model-based test-time adjustment procedure that searches over the pretrained policy family rather than over action sequences, with no parameter updates or additional environment interaction. We provide analysis supporting this adjustment design. On the ExORL benchmark, planning-free GLDSF is competitive with the strongest state-based baseline and outperforms the strongest pixel baseline by 24.0% in aggregate. Incorporating test-time adjustment further boosts aggregate performance by 7.1% on states and 8.4% on pixels. Ablation studies further examine the effects of representation regularization, search space, and other test-time adjustment design choices.

1 INTRODUCTION

Reinforcement Learning (RL) provides a general framework for sequential decision-making, but most high-performing systems (Mnih et al., 2015; Ouyang et al., 2022; Lee et al., 2023) still rely on task-specific rewards or demonstrations, limiting scalability across objectives. Zero-Shot RL (ZSRL) instead pretrains from reward-free experience and, given a new reward, derives a policy without further learning (Touati et al., 2023). Successor Feature (SF)-based methods, building on successor representations (Dayan, 1993) and successor features for transfer (Barreto et al., 2017), are attractive in this setting because they separate reward specification from pretrained policy-dependent dynamics. Within methods, a learned representation ϕ provides the basis for linear reward fitting, $r(s) \approx \phi(s)^\top w$, while a Universal SF (USF) conditions on a behavior code z that indexes a pretrained policy π_z (Borsa et al., 2019). At deployment, standard usage infers w from the downstream reward and directly sets $z = w$ to retrieve a policy (Borsa et al., 2019). However, this formulation leaves two important issues unresolved.

First, SFs define how features are accumulated but not how the representation itself should be learned (Dayan, 1993; Barreto et al., 2017). Existing SF-based ZSRL representations mainly target temporal or policy-related structure (Touati et al., 2023; Park et al., 2024; Bagatella et al., 2026), while the same features also serve as the basis for downstream least-squares reward inference (Touati et al., 2023; Ollivier, 2025a), making their geometry equally important. We therefore introduce Gaussian Latent Dynamics-based Successor Features (GLDSF), which learns ϕ through multi-step latent dynamics prediction and USF learning while regularizing its distribution toward an isotropic Gaussian. Latent dynamics prediction supplies a direct temporal and control-relevant representation signal; isotropy provides a balanced reward-fitting basis, while Gaussianity offers a maximum-entropy interpretation (Cover & Thomas, 2006; Rakelly et al., 2021) and favorable finite-sample concentration (Vershynin, 2012). Under fixed feature energy, we show that isotropy uniquely minimizes

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.

the worst-case amplification of vector-SF error; under an idealized Gaussian model, this favorable property transfers to finite reward-calibration samples.

Second, directly setting $z = w$ conflates two distinct roles: w specifies how behaviors are evaluated under the downstream task, whereas z is a behavior-conditioning code that indexes a policy π_z in the pretrained family (Borsa et al., 2019). Under reward misspecification, finite-sample estimation, and SF/policy approximation errors, the directly retrieved policy π_w can be suboptimal within this family with respect to the true task reward. We therefore keep w fixed as the task evaluator and search over z to maximize an alternative and potentially more accurate estimate of downstream return. Specifically, each candidate z is evaluated by rolling out its induced policy in the learned latent dynamics, using a nonlinear reward predictor over the rollout prefix and the pretrained SF as a terminal value under w. The search changes no pretrained parameters and requires no additional environment data. Unlike action-space planning (Williams et al., 2017b; Chua et al., 2018; Hansen et al., 2024), it searches over coherent pretrained behaviors rather than primitive action sequences.

Our analysis further shows that near-isotropy controls value changes under behavior code perturbations, motivating reward-centered search, and derives a test-time value bound that separates policy-family suboptimality from candidate-evaluation error while exposing the rollout-horizon trade-off. On ExORL (Yarats et al., 2022) across state and pixel observations and datasets collected by various methods, planning-free GLDSF is competitive with the strongest state-based baseline and improves the strongest pixel-based aggregate baseline by 24.0%. Our test-time adjustment further improves aggregate performance by 7.1% on states and 8.4% on pixels. Ablations show the clearest benefit of Gaussian regularization on pixel representations, consistent gains from latent rollout, and stronger performance of behavior-code search than the common action-space planning variants.

2 Preliminaries

2.1 Reward-free zero-shot control

We consider a Markov Decision Process (MDP) without rewards, $\mathcal{M}_0 = (\mathcal{S}, \mathcal{A}, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition model, and discount factor $\gamma \in (0, 1)$. Pretraining is set on a reward-free offline dataset $\mathcal{D} = \{(s_i, a_i, s'_i)\}$ collected by unintentional policies. A reward function r is only specified on deployment. We freeze all pretrained modules and allow only lightweight task inference and test-time search. This protocol is deliberately slightly more permissive than the original, strict "instantaneous, no-planning" definition of ZSRL (Touati et al., 2023); throughout the paper, *zero-shot* refers to reward-free control pretraining with no downstream policy/value fine-tuning or new task interaction, while test-time adjustment denotes the additional computation we explicitly study.

2.2 Universal successor features

Let $\phi : \mathcal{S} \to \mathbb{R}^d$ be a learned state representation. For a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}$ denotes the distribution, the successor feature is

$$F^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_{t+1}) \,\middle|\, s_0 = s, a_0 = a \right]. \tag{1}$$

If a task reward is linear with the representation, $r_w(s) = \phi(s')^\top w$, where s' is the next state from s, then the value function $Q^\pi_w(s, a) = F^\pi(s, a)^\top w$. A Universal Successor Feature (USF) network $\psi(\phi(s), a, z)$ approximates $F^{\pi_z}(s, a)$ for a policy family $\{\pi_z : z \in \mathcal{Z}\}$ indexed by a code z (Borsa et al., 2019). z and w share the same feature dimenion d. We perform pre-training on a sufficiently rich family of policies by sampling z constantly. When testing, given rewards $\{(s_i, r_i)\}_{i=1}^n$ from a small labeled set, we get an unique w from least square regression,

$$\hat{w} \in \arg\min_w \frac{1}{n} \sum_{i=1}^{n} \left(r_i - \phi(s_i)^\top w \right)^2. \tag{2}$$

A notational distinction is important for our method. The vector w is a *reward coefficient*: it evaluates behavior under the downstream task. The vector z is a *behavior code*: it selects a policy

and conditions the USF. Conventional direct zero-shot usage sets two vectors equal (possibly after normalization) and then retrieve the corresponding policy, but we here intentionally allows z to vary while the task vector w stays fixed. This separation avoids conflating "which task is being evaluated" with "which pretrained behavior is being tried".

3 METHOD

Our framework has two stages. During pretraining, GLDSF learns (i) an encoder ϕ, (ii) a latent action-conditioned dynamics model f, (iii) a vector-valued USF ψ, and (iv) a code-conditioned actor π_η with η as its network parameters. ϕ, f, ψ denote both the network and their parameters for simplicity. During test, all four modules are frozen. Test-time adjustment then search over behavior codes, with each candidate evaluated by latent rollout plus a USF terminal value.

3.1 GAUSSIAN-REGULARIZED LATENT DYNAMICS AND USFS

Latent dynamics. We predict the next latent state rather than the raw observation:

$$\mathcal{L}_f = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \|\hat{\phi}_{t+1} - \mathrm{sg}[\phi(s_{t+1})]\|_2^2\right], \tag{3}$$

where T is the prediction horizon, $\tau = (s_0, a_0, \ldots, s_T)$ denotes a sampled trajectory and the rollout is initialized at $\hat{\phi}_0 = \phi(s_0)$, and proceeds recurrently via $\hat{\phi}_t = f(\hat{\phi}_{t-1}, a_{t-1})$ for all $t \geq 1$. $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. $\mathcal{L}_{\mathrm{dyn}}$ optimizes multi-step errors, using the same latent prediction mechanism later employed for test-time evaluation on state or image observations.

Vectorized USF learning For $z \sim \mathcal{Z}$ and $a' \sim \pi_\eta(\phi(s'), z)$, we train a vector-valued SF as:

$$\mathcal{L}_\psi = \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left\|\psi(\phi(s), a, z) - \mathrm{sg}\left[\phi(s') + \gamma\bar{\psi}(\phi(s'), a', z)\right]\right\|_2^2, \tag{4}$$

where $\bar{\psi}$ is a slowly updated target network. For any reward coefficient w, the learned value is $\hat{Q}_{r_w}(s, a, z) = \psi(\phi(s), a, z)^\top w$. For any vector-based temporal difference error $\Delta\psi$, Cauchy–Schwarz gives $|\Delta\psi^\top w| \leq \|\Delta\psi\|_2\|w\|_2$. Thus SF approximation above controls projection error for every w. This property supports cross-policy evaluation during test-time adjustment.

Policy learning The actor is trained to maximize the value associated with the sampled code ,

$$\mathcal{L}_\eta = -\mathbb{E}_{s\sim\mathcal{D}, z\sim\mathcal{Z}, a\sim\pi_\eta(\phi(s),z)}\left[\psi(\phi(s), a, z)^\top z\right]. \tag{5}$$

Here z indexes a policy and supplies the associated pretraining reward coefficient.

Gaussian-regularized representation learning Representation geometry is important for SFs, as downstream inference fits unseen rewards via $r_w(s) = \phi(s')^\top w$. Inspired by Balestriero & LeCun (2025), we argue using isotropic features to provide a well-conditioned basis, whose theoretical advantage is referred to Section 4.1. We further regularize ϕ toward a standard Gaussian from an information-theoretic perspective. Let $X = \phi(s_{t+1})$ and $Y = (\phi(s_t), a_t)$. We model the variational predictive distribution $\tilde{p}_g(X \mid Y)$ as an isotropic Gaussian whose mean is predicted by the latent dynamics model. The forward mutual information admits the lower bound:

$$\begin{aligned} I(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X \mid Y) \geq \mathcal{H}(X) + \mathbb{E}_{X,Y}\left[\log \tilde{p}_g(X \mid Y)\right] \\ &= \mathcal{H}(X) - c_0\mathbb{E}\left[\|f(\phi(s_t), a_t) - \phi(s_{t+1})\|_2^2\right] + c_1, \end{aligned} \tag{6}$$

where c_0, c_1 are constants and $\mathcal{H}$ denotes the entropy. Latent dynamics prediction optimizes the predictive term, while a standard Gaussian maximizes the entropy term under the zero-mean and identity-covariance constraint induced by isotropic features. Rakelly et al. (2021) showed that maximizing such forward information can yield representations sufficient for optimal control under arbitrary rewards under idealized assumptions. Thus, jointly optimizing dynamics prediction and Gaussian regularization encourages ϕ to preserve control-relevant information.

Complete representation learning objectives We minimize the joint objectives to learn ϕ:

$$\mathcal{L}_\phi = \alpha_1 \mathcal{L}_f + \alpha_2 \mathcal{L}_\psi + \alpha_3 \mathcal{L}_{\text{Reg}}, \tag{7}$$

where α_i balance the components, and L^{G} is the regularization loss, which regularizes the distribution of ϕ toward an isotropic Gaussian using SIGReg-style (Balestriero & LeCun, 2025) characteristic-function discrepancy. The specific formula of L_{Reg} is referred to Appendix A.

3.2 Model-based test-time adjustment in behavior-code space

At test time, a ground-truth reward r is specified, we use a small labeled subset of the pretraining data to infer the linear reward coefficient $\hat{w}$ via Eq. 2 and set $z_{\text{pre}} = \hat{w}$). Concurrently, the same subset is used to fit a non-linear latent reward predictor $\hat{r}_\chi$. While $\hat{w}$ acts as a stable anchor compatible with the pretrained USF, $\hat{r}_\chi$ leverages a richer task model for short-horizon evaluation, bypassing USF reward linearization constraints.

At each state, rather than directly executing z_{pre}, we use it to initialize test-time search over the pretrained policy space. For a candidate behavior code z, we simulate its induced behavior using the same latent dynamics model learned during pretraining: $\hat{\phi}_0 = \phi(s_0)$, $a_t = \text{Mean}[\pi_\eta(\hat{\phi}_t, z)]$, $\hat{\phi}_{t+1} = f(\hat{\phi}_t, a_t)$. The candidate is evaluated by the H-step value estimate

$$\widehat{J}_H(s_0, z) = \sum_{t=0}^{H-1} \gamma^t \hat{r}_\chi(\hat{\phi}_{t+1}) + \gamma^H \psi \left(\hat{\phi}_H, \pi_\eta(\hat{\phi}_H, z), z\right)^\top \hat{w}. \tag{8}$$

The rollout prefix is evaluated by the nonlinear reward model, while the USF provides a long-horizon terminal value under the fixed downstream task $\hat{w}$. We optimize Eq. 8 with a Model Predictive Path Integral (MPPI)-style (Williams et al., 2017a) search. At each iteration, candidate codes are sampled from a Gaussian proposal which is centered at z_{pre} initially, high-value candidates are exponentially reweighted according to their predicted returns, and the proposal is updated by weighted moment matching. After a fixed number of iterations, the highest-scoring code $\hat{z}_H$ is used to produce the real action. The full procedure is provided in Appendix A.

4 Theoretical Analysis

We first show how representation regularization affects SF evaluation. We then relate distances between behavior codes to policy values, to show the search over code space cannot bring arbitrarily poor value and finally analyze the specific value bound of the code selected by Eq. 8. Throughout, for notation, exact values are evaluated under the true environment dynamics; a hat on a value function denotes a learned approximation. Let $\pi_z(s) = \pi_\eta(\phi(s), z)$ and $\Pi = \{\pi_z : z \in \mathcal{Z}\}$. For a reward function g, we define the discounted state value $V_g^\pi(s)$ as $\mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t g(s_{t+1}) | s_0 = s]$, where $\mathbb{E}_\pi[\cdot | s_0 = s]$ denotes following π in the true environment. For any coefficient $w \in \mathbb{R}^d$, let $V_{r_w}^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t \phi(s_{t+1})^\top w | s_0 = s\right]$, where $r_w(s) = \phi(s')^\top w$. We also write $V_g^\Pi(s) = \sup_{\pi \in \Pi} V_g^\pi(s), V_g^*(s) = \sup_\pi V_g^\pi(s)$, where the second one is the true optimal value.

4.1 Representation Geometry and SF Value Accuracy

Let ρ be the pretraining state distribution and $\Sigma_\rho = \mathbb{E}_\rho[\phi(s)\phi(s)^\top]$. Representation regularization plays two roles. Population isotropy controls how SF errors are amplified when projected onto unseen rewards. With idealized full-Gaussianity, this favorable population geometry concentrates to the empirical feature geometry used for pratical finite-sample reward fitting.

Let $\widehat{F}_z(s, a)$ be $\psi(\phi(s), a, z)$ and F^{π_z} denote the exact SF of π_z. Assume $\Sigma_\rho \succ 0$ and the vector SF Bellman residual is uniformly bounded by ϵ_B in the Euclidean norm.

Theorem 4.1. *For the reward-energy class $\mathcal{W}_R(\Sigma_\rho) = \{w : w^\top \Sigma_\rho w \le R^2\}$, we have*

$$\sup_{w \in \mathcal{W}_R(\Sigma_\rho)} \sup_{s,a,z} \left| \widehat{F}_z(s, a)^\top w - Q_{r_w}^{\pi_z}(s, a) \right| \le \frac{\epsilon_B R}{(1-\gamma)\sqrt{\lambda_{\min}(\Sigma_\rho)}}, \tag{9}$$

where λ denotes the eigenvalue. Under a fixed feature-energy budget $\text{tr}(\Sigma_\rho) = d$, the representation-related factor in Eq. 9 is uniquely minimized by $\Sigma_\rho = I_d$.

The reward-energy constraint fixes the scale of the task, and the trace constraint fixes the overall scale of the representation. So the theorem compares representation geometries at equal total feature energy and asks which geometry is most robust to an unseen reward of bounded magnitude. Note that $\lambda_{\min}(\Sigma_\rho) \le 1$, with equality if and only if $\Sigma_\rho = I_d$. Hence isotropy minimizes the worst-case amplification of vector-SF error over the reward class. The proof defers to Appendix B.1.

The above derivation is independent of Gaussianity; its role is to transfer the above favorable population geometry to finite samples. Under the concentration model provided by Gaussianity, a well-conditioned theoretical Σ_ρ yields a well-conditioned empirical feature covariance with high probability, which in turn makes linear reward fitting smaller worst-case error even if the true reward is not exactly linear in ϕ. Please see the detail and related discussion in Appendix B.2.

4.2 Code distance and policy value

The same second-moment geometry also controls how code perturbations translate to value changes. A code z selects the policy π_z and also specifies its associated pretraining reward $r_z(s) = \phi(s')^\top z$. These are distinct roles in $V^{\pi_z}_{r_w}$, the evaluation coefficient w need not equal the policy code z. Define the policy's suboptimality as $\epsilon_{\mathrm{pol}}(s,z) = V^{\Pi}_{r_z}(s) - V^{\pi_z}_{r_z}(s)$.

Theorem 4.2. *Let $\|\Sigma_\rho - I\|_{\mathrm{op}} \le \delta < 1$, the discounted state distributions satisfy the coverage condition in Appendix B.3 with constant $C_\rho(s)$, and $z_{\mathrm{pre}} = \hat{w} \in \mathcal{Z}$. Then, for every $z \in \mathcal{Z}$, we have*

$$V^{\pi_z}_{r_{\hat{w}}}(s) \ge V^{\Pi}_{r_{\hat{w}}}(s) - \frac{C_\rho(s)\sqrt{1+\delta}}{1-\gamma}\|z - z_{\mathrm{pre}}\|_2 - \epsilon_{\mathrm{pol}}(s,z). \tag{10}$$

When $\epsilon_{\mathrm{pol}}(s,z)$ is uniformly small near z_{pre}, the theorem yields a local value loss bound, justifying initializing the search distribution mean as z_{pre}. While a large code distance loosens the lower bound, it does not rule out high value; distant codes may match or outperform the direct policy $\pi_{z_{\mathrm{pre}}}$ if it is suboptimal. However, the theorem does not guarantee such distant codes exist.

Without near-isotropy, the same argument uses $\sqrt{\lambda_{\max}(\Sigma_\rho)}$ instead of $\sqrt{1+\delta}$. Large feature variance can thus amplify even small code perturbations, weakening the value guarantee.

4.3 Value bound of test-time adjustment

Let r denote the true task reward. The gap between the pretrained policy family and the true-task oracle is $\Gamma_\Pi(s) = V^*_r(s) - V^{\Pi}_r(s)$. The fitted linear reward is $r_{\hat{w}}(s) = \phi(s')^\top \hat{w}$. Although $\hat{w}$ is estimated, $V^{\pi_z}_{r_{\hat{w}}}$ denotes the exact value of π_z under that fitted reward in the true environment, not an estimated value function. Define the terminal SF evaluator by $\widehat{V}_{r_{\hat{w}}}(\hat{\phi}; z) = \psi(\hat{\phi}, \pi_\eta(\hat{\phi}, z), z)^\top \hat{w}$.

Theorem 4.3. *Uniformly over relevant states and codes, suppose the one-step expected absolute errors of $r_{\hat{w}}$ and $\hat{r}_\chi \circ \phi$ relative to r are bounded by ϵ_{lin} and ϵ_{nl}, and $\left|\widehat{V}_{r_{\hat{w}}}(\phi(x); z) - V^{\pi_z}_{r_{\hat{w}}}(x)\right| \le \epsilon_{\mathrm{SF}}$. Assume $\hat{r}_\chi$ and $\hat{V}_{r_{\hat{w}}}$ are L_r- and L_V-Lipschitz in their latent input. For the rollout $\hat{\phi}_t$ and the corresponding true s_t, suppose $\max_{1\le t\le H} \mathbb{E}\|\hat{\phi}_t - \phi(s_t)\|_2 \le \epsilon_s(H)$, with $\epsilon_s(0) = 0$. Set*

$$E_H = \frac{1-\gamma^H}{1-\gamma}\epsilon_{\mathrm{nl}} + \left(\frac{1-\gamma^H}{1-\gamma}L_r + \gamma^H L_V\right)\epsilon_s(H) + \gamma^H\left(\epsilon_{\mathrm{SF}} + \frac{\epsilon_{\mathrm{lin}}}{1-\gamma}\right). \tag{11}$$

Then when $\hat{z}_H \in \arg\max_{z\in\mathcal{Z}} \widehat{J}_H(s,z)$, we have

$$\left|V^*_r(s) - V^{\pi_{\hat{z}_H}}_r(s)\right| \le \Gamma_\Pi(s) + 2E_H. \tag{12}$$

Precise assumptions and proofs are in Appendix B.4. This result can be regarded as an extension of Theorem 4.1 in Cheng et al. (2025) to our SF setting. When $H = 0$, $E_0 = \epsilon_{\mathrm{SF}} + \epsilon_{\mathrm{lin}}/(1-\gamma)$ and the result describes SF-only code search without latent rollout. For $H > 0$, the terminal SF and linear-reward errors are discounted by γ^H, at the cost of nonlinear-reward and rollout errors. Accurate latent dynamics and reward predictions can therefore reduce the evaluation-error bound, although improvement is not guaranteed for every horizon.

Table 1: Performance comparison across state-based datasets and domains

Dataset	Domain	HILP	FB	RLDP	TD-JEPA	GLDSF	GLDSF-TTA
RND	Cheetah	429 ± 82	438 ± 10	338 ± 60	**524 ± 52**	404 ± 84	519 ± 47
	Walker	665 ± 9	637 ± 92	578 ± 25	629 ± 12	648 ± 30	**691 ± 25**
	Quadruped	543 ± 29	559 ± 31	699 ± 49	597 ± 24	663 ± 5	**727 ± 11**
	Jaco	18 ± 20	20 ± 10	4 ± 1	**40 ± 4**	28 ± 8	32 ± 12
DIAYN	Cheetah	643 ± 5	621 ± 17	631 ± 22	**654 ± 7**	645 ± 4	648 ± 3
	Walker	478 ± 57	184 ± 4	334 ± 55	446 ± 54	414 ± 29	**525 ± 28**
	Quadruped	**651 ± 22**	448 ± 33	532 ± 18	577 ± 44	559 ± 36	594 ± 17
	Jaco	27 ± 6	19 ± 11	5 ± 2	16 ± 5	**30 ± 21**	28 ± 18
APT	Cheetah	423 ±102	336 ± 61	505 ± 37	**641 ± 42**	459 ± 52	482 ± 63
	Walker	**694 ± 11**	601 ± 70	611 ± 20	680 ± 15	686 ± 8	666 ± 11
	Quadruped	**656 ± 41**	471 ±124	590 ± 22	592 ± 73	630 ± 30	**651 ± 10**
	Jaco	6 ± 4	9 ± 7	7 ± 4	23 ± 4	**35 ± 11**	**34 ± 7**

Table 2: Performance comparison across pixel-based datasets and domains

Dataset	Domain	HILP	FB	RLDP	TD-JEPA	GLDSF	GLDSF-TTA
RND	Cheetah	509 ± 24	186 ± 20	345 ± 11	569 ± 23	529 ± 29	**612 ± 17**
	Walker	297 ± 9	193 ± 39	440 ± 10	461 ± 21	437 ± 42	**477 ± 27**
	Quadruped	222 ± 26	255 ± 54	309 ± 30	217 ± 27	**414 ± 65**	402 ± 81
	Jaco	44 ± 4	**51 ± 11**	3 ± 1	29 ± 2	46 ± 5	45 ± 10
DIAYN	Cheetah	461 ±153	220 ± 42	261 ±223	350 ± 61	**641 ± 24**	618 ± 54
	Walker	344 ± 7	166 ± 27	230 ± 33	355 ± 11	339 ± 18	**381 ± 38**
	Quadruped	101 ± 26	70 ± 17	74 ± 17	94 ± 13	**156 ± 21**	98 ± 49
	Jaco	46 ± 13	31 ± 10	53 ± 6	44 ± 12	48 ± 6	**58 ± 7**
APT	Cheetah	439 ± 62	255 ± 17	213 ±166	458 ± 40	393 ± 71	**590 ± 75**
	Walker	435 ± 36	198 ± 15	469 ± 51	**519 ± 15**	473 ± 22	483 ± 43
	Quadruped	194 ± 12	211 ± 33	105 ± 38	259 ± 28	**451 ± 36**	445 ± 41
	Jaco	31 ± 3	**42 ± 3**	1 ± 0	31 ± 7	33 ± 6	38 ± 12

5 EXPERIMENTS

We evaluate our method on ExORL (Yarats et al., 2022) and focus on four questions: (i) how is the planning-free policy compared with prior state-of-the-art ZSRL methods, and how much does test-time adjustment improve it; (ii) how does representation regularization affect zero-shot performance empirically; (iii) is it more effective to search over behavior codes than to plan directly in the action space ; and (iv) are current designs of test-time adjustment necessary?

5.1 EXPERIMENTAL SETUP

Benchmark We use ExORL on four DeepMind Control (Tassa et al., 2018) domains—*Cheetah*, *Walker*, *Quadruped*, and *Jaco*—covering locomotion and manipulation. We consider both low-dimensional state and pixel observations. Varying data distributions requires methods to generalize over different state-action coverage. Thus, we use datasets collected by RND (Burda et al., 2019), APT (Liu & Abbeel, 2021), and DIAYN (Eysenbach et al., 2019), three methods representing distinct types of unsupervised exploration policies.

Baselines We compare with four representative zero-shot RL methods. HILP (Park et al., 2024) learns a temporally structured Hilbert representation under the USFs framework. FB (Touati & Ollivier, 2021) factorizes successor measures into forward and backward representations for downstream task inference. RLDP (Jajoo et al., 2026) improves FB by learning dynamic-based representations with an orthogonality-based regularizer. TD-JEPA (Bagatella et al., 2026) learns policy-conditioned long-horizon latent-predictive representations using temporal-difference objectives. We denote our pretrained model augmented with test-time adjustment by GLDSF-TTA.

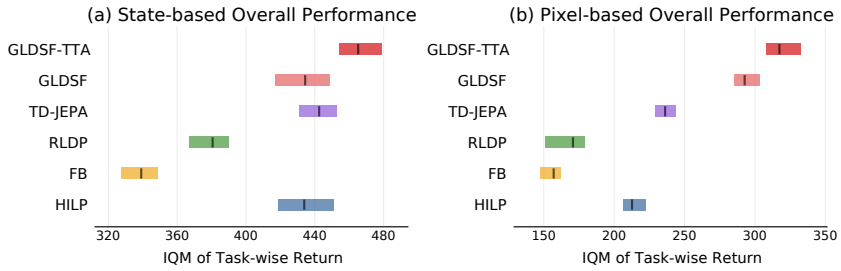


Figure 1: Overall performance under state-based and pixel-based observations. The IQM score are aggregated over 3 datasets, 4 domains, 4 tasks and 6 seeds (288 values in total).

Evaluation protocol Task results are reported based on their return, the sum of rewards collected in one episode. Each task of seed is evaluated over 20 episodes. For aggregate comparisons, Jaco returns are multiplied by four to match the maximum episode return of other domains. Full results, implementation details, and hyperparameters are provided in the Appendices D and E .

5.2 Zero-Shot Performance and Test-Time Adjustment

Tables 1 and 2 show per-dataset-domain comparisons, and Figure 1 shows overall performance. Each table entry reports the mean of task returns over across 4 tasks within that domain (e.g., jump, run, stand, walk for Quadruped) and standard deviation over 6 seeds. In the figure, vertical markers indicate Interquatile Mean (IQM) point estimates across all results, and horizontal bars are 95% percentile intervals with bootstrapped 50,000 resamples. Planning-free and adjusted variants are reported separately to isolate the gains from representation learning versus test-time computation.

Planning-free performance On state observations, GLDSF reaches an aggregate score of 434.4, close to the strongest baseline average of 442.5 from TD-JEPA. The difference is more pronounced from pixels: GLDSF obtains 292.7 compared with 236.1 for TD-JEPA, an improvement of 24.0%. This pixel-based advantage is observed under all three pretraining datasets. For example, GLDSF obtains 451 on APT–Quadruped compared with 259 for the best baseline, and 641 on DIAYN–Cheetah compared with 461. These results show that the planning-free policy remains effective across both observation modalities, with the clearest aggregate gains in the pixel setting.

Effect of test-time adjustment GLDSF-TTA improves aggregate scores from 434.4 to 465.2 (+7.1%) on states and from 292.7 to 317.4 (+8.4%) on pixels. It improves 9 of the 12 state-based dataset–domain entries and 7 of the 12 pixel-based entries. The largest domain-level increases include DIAYN–Walker for states (414 to 525) and APT–Cheetah for pixels (393 to 590). A potential explanation is that limited data coverage of these datasets leaves both the zero-shot policy and the learned SF estimates imperfect. Large policy suboptimality leaves room for better code selection, as captured by $\epsilon_{\text{pol}}(s, z_{\text{pre}})$ in Section 4.2, though inaccurate SF estimates may hinder this identification. While latent rollouts can mitigate the issue by reducing SF reliance, provided that dynamics and reward predictions remain more accurate, consistent with the error trade-off in Theorem 4.3.

5.3 Effect of Representation Regularization

We next isolate the effect of representation regularization while keeping other modules fixed. We compare with two variants: No Regularization applied (NoReg), and isotropy-only regularization, which regularizes the second-order feature geometry solely. Results in Table 3 are also averaged over 6 seeds and the same 4 tasks. They show a different pattern for states and pixels. On state observations, NoReg performs best, likely because additional regularization adds unnecessary optimization complexity to existing well-defined low-dimensional features. Conversely, on pixel observations, Gaussian regularization outperforms both NoReg and Isotropy-only, and NoReg clearly degrades. Overall, the experiments do not suggest that Gaussian regularization is universally superior for control, but it delivers the most consistent behavior across modalities and is particularly

Table 3: Representation regularization comparison under state-based and pixel-based observations.

Domain	State-based			Pixel-based		
	NoReg	Isotropy-only	Gaussian	NoReg	Isotropy-only	Gaussian
Cheetah	433 ± 14	435 ± 44	404 ± 84	252 ± 34	440 ± 14	529 ± 29
Walker	656 ± 43	618 ± 15	648 ± 30	243 ± 43	388 ± 15	437 ± 42
Quadruped	640 ± 45	633 ± 31	663 ± 5	391 ± 81	459 ± 33	414 ± 65
Average	576	562	572	295	430	460

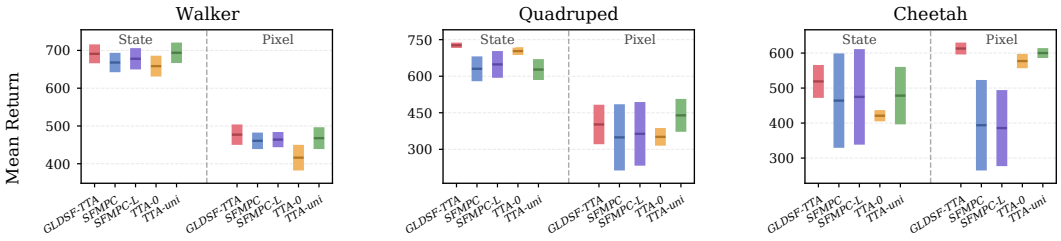


Figure 2: Ablations of search space and test-time adjustment design. For each method and domain, the center mark reports the mean return over 4 tasks and 6 seeds, and the bar shows the standard deviation of this task-mean return across six seeds. The test is on the RND dataset.

effective in higher-dimensional pixel settings, while the decent Isotropy-only performance indicates that the isotropy is a vital regularization component.

5.4 Search Space for Test-Time Adjustment

We here study whether test-time search is better performed over behavior codes or directly over action sequences. As an action-space counterpart, SFMPC performs MPC with the same learned dynamics and SF terminal value, while mixing actions proposed by the pretrained policy into the candidate sequences , following the common practice of policy-guided MPC (Hansen et al., 2024). SFMPC-L further increases the search budget from 256 samples and 6 iterations (which we use for GLDSF-TTA throughout the experiment) to 1,024 samples and 12 iterations.

Figure 2 shows that GLDSF-TTA consistently outperforms both action-space planning variants. Increasing the budget can only provide marginal help. On Quadruped and Cheetah SFMPC also exhibits much higher variance. This performance gap likely occurs because behavior codes specify state-conditioned policies that constrain the search to temporally coherent behaviors from the pretrained family, instead of composing actions arbitrarily. Thus, behavior space serves as an effective search parameterization in our setting. This comparison is not meant to imply that action-space planning is generally inferior; the two optimize different variables and could be combined in future.

5.5 Designs of Test-Time Adjustment

We next ablate two design choices in Figure 2. TTA-0 removes latent rollout and evaluates candidate codes only with the SF estimate, corresponding to $H = 0$ in Theorem 4.3. It underperforms the full method in all settings. This supports using predicted transitions to complement the terminal SF estimate, reducing reliance on SF accuracy when local dynamics and reward prediction are reliable.

TTA-uni instead samples codes uniformly rather than initializing around z_{pre}. It remains competitive in several settings and occasionally improves over GLDSF-TTA, but also degrades noticeably in others. This pattern is consistent with Theorem 4.2: broader search may discover better behaviors farther from z_{pre}, while reward-centered initialization provides a bias toward task-relevant regions. Overall, code-space search appears reasonably robust, while latent rollout and reward-centered initialization improve its reliability.

6 RELATED WORK

6.1 ZERO-SHOT RL

Successor representations and successor features separate policy-dependent dynamics from rewards, enabling transfer through generalized policy improvement and task-conditioned approximators (Dayan, 1993; Barreto et al., 2017; Borsa et al., 2019; Barreto et al., 2020). Recently, research on SF-based methods in the field of ZSRL focus on feature design. Touati et al. (2023) compares diverse feature-learning objectives empirically, Park et al. (2024) propose to learn temporal-distance embeddings, and Bagatella et al. (2026) captures long-horizon dynamics through temporal-difference latent prediction. Hierarchical approach (Chandrasekar & Machado, 2026) identifies Laplacian feature can help compose behaviors beyond their linear reward span. Pure theory studies feature quality and direct zero-shot loss optimization under reward-prior and regularization assumptions (Ollivier, 2025a;b). Our work jointly considers control-relevant information preserving and reward-fitting geometry for feature learning, balancing theoretical insight with practical feasibility.

Beyond SFs, successor-measure learning includes Forward-Backward (FB) occupancy factorization (Touati & Ollivier, 2021), conservative offline variants (Jeen et al., 2024), and regularized latent prediction (Jajoo et al., 2026). Alternative formulations use affine behavior-space bases (Proto Successor Measures, PSM) (Agarwal et al., 2025) or nonlinear Functional Reward Encodings (FRE) (Frans et al., 2024). Nevertheless, standard finite-dimensional FB can discard control-relevant reward distinctions (Cetin et al., 2025; Agarwal et al., 2025), limited PSM bases can omit relevant occupancies, and FRE depends on reward priors and sampled encodings. These limitations differs from SF's linear reward-fitting deficiency, yet implying the theoretical imperfections of various ZSRL methods and motivating studying test-time adjustment to refine pretrained zero-shot policies continuously.

6.2 TEST-TIME OPTIMIZATION AND BEHAVIOR-SPACE ADAPTATION

Model-based control improves deployment-time decisions by optimizing action sequences under learned dynamics, through sampling-based model predictive control (Williams et al., 2017b; Chua et al., 2018) or planning in learned latent dynamics (Hafner et al., 2019; Hansen et al., 2024). Later work shifts optimization from primitive actions to reusable skills, enabling prior-guided downstream learning (Pertsch et al., 2021) and long-horizon planning with skill-level dynamics (Shi et al., 2023). More generally, pretrained behaviors can serve as a basis for constructing downstream policies through skill composition (Barreto et al., 2019). Access to such a repertoire, however, does not by itself resolve which behavior should be deployed for a new reward: in zero-shot agents, direct reward-to-code inference can miss better policies within the same pretrained family (Sikchi et al., 2025). Beyond direct retrieval, downstream behavior can be optimized by selecting achievable long-term outcomes with distributional successor features (Zhu et al., 2024), adapting latent task codes through online interaction (Sikchi et al., 2025), or learning task-specific policies from intention-conditioned occupancy models (Zheng et al., 2026). Our method instead performs computation-only adaptation through model-based search over behavior codes within a frozen policy family, requiring neither fine-tuning of pretrained parameters nor additional environment data.

7 CONCLUSION

We presented GLDSF, an SF framework that couples latent dynamics prediction with Gaussian-regularized representation learning, and GLDSF-TTA, a test-time procedure for adjusting behavior selection. The central idea is to treat representation geometry and policy selection as two distinct sources of zero-shot error: isotropic features provide a balanced basis for reward inference and SF evaluation, while separating the task coefficient w from the behavior code z allows better policy to be chosen within the pretrained family. Our theory characterizes both effects and provides an explicit error trade-off for latent-rollout-based test-time evaluation. Empirically, GLDSF is competitive on state observations and performs strongly from pixels, while test-time adjustment produces further gains. But adjustment cannot recover behaviors absent from the pretrained family or information missing from the learned representation, and its effectiveness depends on accurate dynamics, reward, and value estimates. Addressing these limitations, and combining behavior-space adjustment with complementary action-space planning, are promising directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools for helping develop conceptual framework of isotropic representation, assisting in the writing of all proofs, formulating mathematical claims from Appendix B.2, providing critical ingredients for proving mathematical claims from Appendix B.2, proposing or refining hypotheses of Theorem 4.2, assisting with translation. We have not used generative AI tools for generating synthetic data sets, formulating other mathematical claims, designing or providing feedback on research methodology or experiments, implementing methods, cleaning and reformating dataset, supporting qualitative and thematic data analysis, interpreting results. Additionally, we used generative AI tools for creating code to organize experimental results, editing code to implement methods, drafting Section 2 and Section 6.2, summarizing existing literature about Section 6.2 topic, editing a research paper to improve readability, and formating references. We have reviewed all AI-assisted work. We re-examine the correctness of all the concepts and proofs in the paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, pp. 566–586. PMLR, 2025.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, 2026.

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

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado P. van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 4055–4065. Curran Associates, Inc., 2017.

André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan J. Hunt, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 13052–13062. Curran Associates, Inc., 2019.

André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117 (48):30079–30087, 2020.

Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. *Reinforcement Learning Journal*, 6:1647–1680, 2025.

Siddarth Chandrasekar and Marlos C. Machado. The Laplacian Keyboard: Beyond the linear span. In *Workshop on Reinforcement Learning Beyond Rewards: Towards Scalable General-Purpose Agents at the 3rd Reinforcement Learning Conference (RLC 2026)*, 2026.

Jie Cheng, Ruixi Qiao, Yingwei Ma, Binhua Li, Gang Xiong, Qinghai Miao, Yongbin Li, and Yisheng Lv. Scaling offline model-based RL via jointly-optimized world-action model pretraining. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pp. 4759–4770. Curran Associates, Inc., 2018.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2nd edition, 2006.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, pp. 13927–13942. PMLR, 2024.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*, 2024.

Pranaya Jajoo, Harshit Sikchi, Siddhant Agarwal, Amy Zhang, Scott Niekum, and Martha White. Regularized latent dynamics prediction is a strong baseline for behavioral foundation models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, 2026.

Scott Jeen, Tom Bewley, and Jonathan M. Cullen. Zero-shot reinforcement learning from low quality data. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 16894–16942. Curran Associates, Inc., 2024.

Kuang-Huei Lee, Ted Xiao, Adrian Li, Paul Wohlhart, Ian Fischer, and Yao Lu. PI-QT-Opt: Predictive information improves multi-task robotic reinforcement learning at scale. In *Proceedings of the 6th Conference on Robot Learning (CoRL 2022)*, pp. 1696–1707. PMLR, 2023.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pp. 18459–18473. Curran Associates, Inc., 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Yann Ollivier. Which features are best for successor features? *arXiv preprint arXiv:2502.10790*, 2025a.

Yann Ollivier. Tackling the zero-shot reinforcement learning loss directly. *arXiv preprint arXiv:2502.10792*, 2025b.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 27730–27744. Curran Associates, Inc., 2022.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, pp. 39737–39761. PMLR, 2024.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 4th Conference on Robot Learning (CoRL 2020)*, pp. 188–204. PMLR, 2021.

Kate Rakelly, Abhishek Gupta, Carlos Florensa, and Sergey Levine. Which mutual-information representation learning objectives are sufficient for control? In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pp. 26345–26357. Curran Associates, Inc., 2021.

Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL 2022)*, pp. 2262–2272. PMLR, 2023.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. *Reinforcement Learning Journal*, 6:1048–1074, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pp. 13–23. Curran Associates, Inc., 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*, 2023.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok (eds.), *Compressed Sensing: Theory and Applications*, chapter 5, pp. 210–268. Cambridge University Press, 2012.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017a.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *Proceedings of the 34th IEEE International Conference on Robotics and Automation (ICRA 2017)*, pp. 1714–1721. IEEE, 2017b.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Chongyi Zheng, Seohong Park, Sergey Levine, and Benjamin Eysenbach. Intention-conditioned flow occupancy models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR 2026)*, 2026.

Chuning Zhu, Xinqi Wang, Tyler Han, Simon Shaolei Du, and Abhishek Gupta. Distributional successor features enable zero-shot policy optimization. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 124612–124639. Curran Associates, Inc., 2024.

A SUPPLEMENTARY DETAILS IN METHOD DESIGN

A.1 SPECIFIC FORMULA OF $\mathcal{L}_G$

Given that direct distribution matching in high-dimensional space is not feasible, SIGReg refers to the Cramér-Wold theorem: if the marginal distributions of a random variable follow a standard Gaussian distribution in all one-dimensional projection directions, then this random variable follows an isotropic Gaussian distribution. For implementation, a set of projection vectors $U = \{u_1, ..., u_{N_{\text{proj}}}\}$ is uniformly sampled from the unit hypersphere $\mathcal{S}^{d-1}$. For a batch of features $\Phi \in \mathbb{R}^{B\times d}$, calculate its linear projection in each direction $\phi^{(n)} = \Phi u_n, n = 1, ..., N_{\text{proj}}$. And then the Characteristic Function (CF) is used to measure the difference between the projected one-dimensional distribution and the standard Gaussian distribution. For the n-th projection direction, its empirical CF is defined as: $\tilde{c}_n(\omega) = \frac{1}{B}\sum_{b=1}^{B} \exp(\mathrm{i}\omega\phi_b^{(n)})$, where i is the imaginary unit, and ω is the frequency domain variable. The CF of the standard Gaussian is $c_{\text{g}}(\omega) = \exp(-\omega^2/2)$. The final loss function is defined as the weighted L2 distance between the these two CFs in the frequency domain:

$$\mathcal{L}_{\text{Reg}} = \frac{1}{N_{\text{proj}}} \sum_{n=1}^{N_{\text{proj}}} \int_{\mathbb{R}} \left|\tilde{c}_n(\omega) - c_{\text{g}}(\omega)\right|^2 e(\omega) d\omega \tag{13}$$

where $e(v) = \exp(-\omega^2)$ is the Gaussian weighting function.

A.2 TEST-TIME ADJUSTMENT PROCEDURE

The detailed test-time search algorithm is presented in Algorithm 1.

Brief explanation We optimize Eq. 8 by maintaining a distribution over behavior codes and iteratively shifting its probability mass toward codes with higher predicted return. Following an MPPI-style view, the ideal return-weighted distribution at iteration j is $p_j^\star(z) \propto \nu_{j-1}(z)\exp\Big(\tau \hat{J}_H(s_0, z)\Big)$, where ν_{j-1} is the current proposal over behavior codes. The next Gaussian proposal is obtained by projecting this target distribution back onto a tractable parametric family, $\nu_j = \arg\min_{\nu\in\mathcal{Q}} D_{\text{KL}}\big(p_j^\star \| \nu\big)$. In practice, we approximate this projection using sampled behavior codes and restrict the update to the top-K candidates for robustness. The resulting weighted moment matching yields the mean and variance updates in Algorithm 1.

Algorithm 1 Test-time search in behavior code space

Require: pretrained ϕ, f, ψ, π_η; reward models $\hat{r}_\chi, \hat{w}$; current state s_0; horizon H; samples/elites N, N_{elite}; iterations N_{iter}; temperature β_{temp}
$\mu_z^0 \leftarrow z_{\text{pre}}, \quad \sigma_z^0 \leftarrow \sigma_z^{\max}\mathbf{1}$
for iteration $j = 1, \ldots, N_{\text{iter}}$ **do**
  Sample $z_i \sim \mathcal{N}\big(\mu_z^{j-1}, \text{diag}((\sigma_z^{j-1})^2)\big), i = 1, \ldots, N$
  $\hat{\phi}_i^0 \leftarrow \phi(s_0), \quad \hat{J}_i \leftarrow 0, \qquad \forall i$
  for step $t = 0, \ldots, H-1$ **do**
    $a_i^t \leftarrow \pi_\eta(\hat{\phi}_i^t, z_i), \quad \hat{\phi}_i^{t+1} \leftarrow f(\hat{\phi}_i^t, a_i^t), \quad \hat{J}_i \leftarrow \hat{J}_i + \gamma^t \hat{r}_\chi(\hat{\phi}_i^{t+1}), \qquad \forall i$
  $\hat{J}_i \leftarrow \hat{J}_i + \gamma^H \psi\left(\hat{\phi}_i^H, \pi_\eta(\hat{\phi}_i^H, z_i), z_i\right)^\top \hat{w}, \qquad \forall i$
  Let $\mathcal{I}_j$ index the N_{elite} largest scores and set $e_i \propto \exp(\beta_{\text{temp}}\hat{J}_i)$ for $i \in \mathcal{I}_j$,
  $\mu_z^j \leftarrow \sum_{i\in\mathcal{I}_j} e_i z_i, \quad \sigma_z^j \leftarrow \sqrt{\sum_{i\in\mathcal{I}_j} e_i (z_i - \mu_z^j)^2}$
$i^* \leftarrow \arg\max_{i\in\mathcal{I}_{N^{\text{iter}}}} \hat{J}_i$ using the final population
return $\hat{z}_H \leftarrow z_{i^*}$

B PROOFS FOR SECTION 4

This appendix proves Theorems 4.1,4.2 and 4.3 and gives finite-sample consequences of Gaussian representations for reward fitting and test-time evaluation. We use the same next-state reward convention as in the main text,

$$V_g^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t g(s_{t+1}) | s_0 = s\right], \qquad r_w(s) = \phi(s')^\top w, \forall w \in \mathbb{R}^d \tag{14}$$

and $\pi_z(s) = \pi_\eta(\phi(s), z)$, $\Pi = \{\pi_z : z \in \mathcal{Z}\}$. The encoder and the pretrained policy family are fixed throughout the analysis. We assume that all displayed values and expectations are finite and that the corresponding discounted absolute rewards are integrable.

B.1 PROOF OF THEOREM 4.1

Let $\widehat{F}_z(s,a) = \psi(\phi(s), a, z)$, and F^{π_z} denote the exact successor feature (SF) of π_z. Assume $\Sigma_\rho = \mathbb{E}_\rho[\phi(s)\phi(s)^\top] \succ 0$ and that the vector SF Bellman residual is uniformly bounded by ϵ_{B} in Euclidean norm.

Proof. We divide the proof into three steps.

Step 1: Convert the Bellman residual into a uniform vector-SF error bound. For a fixed behavior code z, define the true SF Bellman operator by

$$(\mathcal{T}_z F)(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\phi(s') + \gamma F(s', \pi_z(s'))\right]. \tag{15}$$

The exact SF F^{π_z} is a fixed point of this operator,

$$F^{\pi_z} = \mathcal{T}_z F^{\pi_z}.$$

The residual assumption is

$$\sup_{s,a,z} \|\widehat{F}_z(s,a) - (\mathcal{T}_z \widehat{F}_z)(s,a)\|_2 \le \epsilon_{\mathrm{B}}. \tag{16}$$

For vector-valued functions, define

$$\|F\|_{\infty,2} = \sup_{s,a} \|F(s,a)\|_2.$$

For any two such functions F_1 and F_2, Jensen's inequality gives, for each (s,a),

$$\begin{aligned}\|(\mathcal{T}_z F_1)(s,a) - (\mathcal{T}_z F_2)(s,a)\|_2 &= \gamma \|\mathbb{E}_{s' \sim P(\cdot|s,a)}\left[F_1(s', \pi_z(s')) - F_2(s', \pi_z(s'))\right]\|_2 \\ &\le \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\|F_1(s', \pi_z(s')) - F_2(s', \pi_z(s'))\|_2\right] \\ &\le \gamma \|F_1 - F_2\|_{\infty,2}.\end{aligned}$$

Taking the supremum over (s,a) therefore yields the contraction property

$$\|\mathcal{T}_z F_1 - \mathcal{T}_z F_2\|_{\infty,2} \le \gamma \|F_1 - F_2\|_{\infty,2}. \tag{17}$$

Using the triangle inequality, the residual bound, the fixed-point identity, and Eq. 17, we obtain

$$\begin{aligned}\|\widehat{F}_z - F^{\pi_z}\|_{\infty,2} &\le \|\widehat{F}_z - \mathcal{T}_z \widehat{F}_z\|_{\infty,2} + \|\mathcal{T}_z \widehat{F}_z - \mathcal{T}_z F^{\pi_z}\|_{\infty,2} \\ &\le \epsilon_{\mathrm{B}} + \gamma \|\widehat{F}_z - F^{\pi_z}\|_{\infty,2}.\end{aligned}$$

Rearranging gives

$$\|\widehat{F}_z - F^{\pi_z}\|_{\infty,2} \le \frac{\epsilon_{\mathrm{B}}}{1-\gamma}. \tag{18}$$

Because the residual assumption Eq. 16 is uniform in z, the same bound holds uniformly over all z.

Step 2: Project the vector-SF error onto a bounded-energy reward class. For every (s, a, z) and every coefficient w,

$$\begin{aligned}\left|\widehat{F}_z(s,a)^\top w - Q^{\pi_z}_{r_w}(s,a)\right| &= \left|\left(\widehat{F}_z(s,a) - F^{\pi_z}(s,a)\right)^\top w\right| \\ &\le \|\widehat{F}_z(s,a) - F^{\pi_z}(s,a)\|_2 \|w\|_2 \\ &\le \frac{\epsilon_{\mathrm{B}}}{1-\gamma}\|w\|_2, \end{aligned} \tag{19}$$

where the first inequality is Cauchy–Schwarz and the second uses Eq. 18.

Since $\Sigma_\rho \succ 0$, its Rayleigh quotient satisfies

$$w^\top \Sigma_\rho w \ge \lambda_{\min}(\Sigma_\rho)\|w\|_2^2. \tag{20}$$

Equivalently,

$$\|w\|_2 \le \frac{\sqrt{w^\top \Sigma_\rho w}}{\sqrt{\lambda_{\min}(\Sigma_\rho)}}. \tag{21}$$

For $w \in \mathcal{W}_R(\Sigma_\rho)$, we have $w^\top \Sigma_\rho w \le R^2$, hence

$$\|w\|_2 \le \frac{R}{\sqrt{\lambda_{\min}(\Sigma_\rho)}}. \tag{22}$$

Substituting Eq. 22 into Eq. 19 yields, simultaneously for every $w \in \mathcal{W}_R(\Sigma_\rho)$ and every (s, a, z),

$$\left|\widehat{F}_z(s,a)^\top w - Q^{\pi_z}_{r_w}(s,a)\right| \le \frac{\epsilon_{\mathrm{B}} R}{(1-\gamma)\sqrt{\lambda_{\min}(\Sigma_\rho)}}.$$

Taking first the supremum over (s, a, z) and then over $w \in \mathcal{W}_R(\Sigma_\rho)$ proves Eq. 9.

Step 3: Optimize the representation-dependent factor under a fixed feature-energy budget. Let $\lambda_1, \dots, \lambda_d > 0$ be the eigenvalues of Σ_ρ. The constraint $\mathrm{tr}(\Sigma_\rho) = d$ gives

$$\sum_{i=1}^{d} \lambda_i = d.$$

Therefore

$$\lambda_{\min}(\Sigma_\rho) \le \frac{1}{d}\sum_{i=1}^{d} \lambda_i = 1. \tag{23}$$

Because $x \mapsto 1/\sqrt{x}$ is strictly decreasing on $(0, \infty)$, Eq. 23 implies

$$\frac{1}{\sqrt{\lambda_{\min}(\Sigma_\rho)}} \ge 1. \tag{24}$$

The identity matrix I_d satisfies $\mathrm{tr}(I_d) = d$ and has $\lambda_{\min}(I_d) = 1$, so it attains the lower bound.

It remains to prove uniqueness. If another positive-definite matrix Σ_ρ attained the same factor, then necessarily $\lambda_{\min}(\Sigma_\rho) = 1$. Hence every eigenvalue obeys $\lambda_i \ge 1$. But their sum is d, so necessarily $\lambda_i = 1$ for every i. Since Σ_ρ is symmetric positive definite, its eigendecomposition then gives $\Sigma_\rho = I_d$. Therefore I_d is the unique minimizer. □

Remark B.1 (Why the trace constraint is the appropriate scale normalization). Because

$$\mathrm{tr}(\Sigma_\rho) = \mathbb{E}_\rho\left[\|\phi(s)\|_2^2\right],$$

fixing $\mathrm{tr}(\Sigma_\rho)$ fixes the total second-moment "energy" of the representation. Without such a scale constraint, multiplying all features by a constant c would replace Σ_ρ by $c^2\Sigma_\rho$ and artificially shrink the factor $1/\sqrt{\lambda_{\min}(\Sigma_\rho)}$ by $1/c$, even though this would be only a rescaling rather than an improvement in geometry.

The particular choice $\mathrm{tr}(\Sigma_\rho) = d$ matches the scale of the target $\mathcal{N}(0, I_d)$. More generally, if $\mathrm{tr}(\Sigma_\rho) = C$, then

$$\lambda_{\min}(\Sigma_\rho) \le \frac{C}{d},$$

and the same argument shows that the factor is uniquely minimized by $\Sigma_\rho = (C/d)I_d$.

Remark B.2 (Role of Gaussianity). No Gaussian assumption is used in Theorem 4.1 or in its proof. The theorem is a second-moment geometric statement. A Gaussian sampling assumption can be introduced separately to obtain finite-sample concentration of an empirical second-moment matrix around Σ_ρ, which is a different role from the population-level geometry established here.

B.2 Finite-Sample Consequences of Gaussian Representations

This section strengthens the second-moment assumption to an idealized distributional model in order to separate population geometry from finite-sample reward fitting. Isotropy discussed before represents the desired population geometry, while Gaussianity provides a sufficient concentration model that allows this geometry to transfer from the population to the finite sample setting; and this empirical conditioning is what truly ensures stable reward fitting.

Gaussian empirical conditioning. Condition on the pretrained modules. Let $s_1, \dots, s_n$ be independent draws from ρ and assume $\phi(s_i) \sim \mathcal{N}(0, \Sigma_\rho)$ with $\Sigma_\rho \succ 0$. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ have rows $\phi(s_i)^\top$, set $\widehat{\Sigma}_n = \mathbf{X}^\top\mathbf{X}/n$, and write $G = \mathbf{X}\Sigma_\rho^{-1/2}$. The entries of G are independent standard Gaussian variables. Gaussian singular-value concentration (Vershynin, 2012, Corollary 5.35) gives, for any $t > 0$, with probability at least $1 - 2e^{-t^2/2}$,

$$\sqrt{n} - \sqrt{d} - t \le s_{\min}(G) \le s_{\max}(G) \le \sqrt{n} + \sqrt{d} + t.$$

Taking $t = \sqrt{2\log(4/\beta)}$ and $u_n = \sqrt{d/n} + \sqrt{2\log(4/\beta)/n} < 1$ gives, with probability at least $1 - \beta/2$,

$$(1 - u_n)^2\Sigma_\rho \preceq \widehat{\Sigma}_n \preceq (1 + u_n)^2\Sigma_\rho.$$

In particular,

$$\lambda_{\min}(\widehat{\Sigma}_n) \ge (1 - u_n)^2\lambda_{\min}(\Sigma_\rho).$$

If additionally $\|\Sigma_\rho - I_d\|_{\mathrm{op}} \le \delta < 1$, then $\lambda_{\min}(\widehat{\Sigma}_n) \ge (1 - \delta)(1 - u_n)^2$ and

$$\|\widehat{\Sigma}_n - I_d\|_{\mathrm{op}} \le \delta + (1 + \delta)(2u_n + u_n^2).$$

Thus a Gaussian population with a balanced spectrum inherits that conditioning in a finite calibration sample with high probability.

Exact reward labels with linear misspecification. In our setting, the deployment can have exact reward relabeling even when the true reward is not exactly linear in ϕ. Define the population projection

$$w^\star \in \arg\min_w \mathbb{E}_\rho[(r(s) - \phi(s)^\top w)^2], \qquad b(s) = r(s) - \phi(s)^\top w^\star.$$

The normal equations give the orthogonality condition

$$\mathbb{E}_\rho[\phi(s)b(s)] = 0.$$

For exact labels, empirical OLS satisfies

$$\widehat{w} - w^\star = \widehat{\Sigma}_n^{-1}\left(\frac{1}{n}\sum_{i=1}^n \phi(s_i)b(s_i)\right).$$

To obtain the finite-sample rate, define $g_i = \Sigma_\rho^{-1/2}\phi(s_i)$ and $v^\star = \Sigma_\rho^{1/2}w^\star$. Then $g_i \sim \mathcal{N}(0, I_d)$ and, by the projection property,

$$\|v^\star\|_2^2 = \mathbb{E}[(\phi(s)^\top w^\star)^2] \le \mathbb{E}[r(s)^2] \le R_{\max}^2.$$

Moreover, $\mathbb{E}[g_i b_i] = 0$. For any unit vector u,

$$(u^\top g_i)b_i = (u^\top g_i)r(s_i) - (u^\top g_i)(g_i^\top v^\star).$$

The first term is dominated by $R_{\max}|u^\top g_i|$, while the second is a product of two sub-Gaussian random variables with scale at most $R_{\max}$. Hence the centered scalar projections of $g_i b_i$ have sub-exponential norm bounded by a universal constant times $R_{\max}$. A standard $1/2$-net argument and Bernstein concentration give, for a universal $C > 0$, with probability at least $1 - \beta/2$,

$$\left\|\frac{1}{n}\sum_{i=1}^n g_i b_i\right\|_2 \le CR_{\max}\left[\sqrt{\frac{d + \log(2/\beta)}{n}} + \frac{d + \log(2/\beta)}{n}\right].$$

When $n \geq C_1(d+\log(2/\beta))$, the second term is absorbed into the first. On the Gram-concentration event, the whitened empirical Gram matrix $\widehat{G}_n = G^\top G/n$ has a smallest eigenvalue bounded below by a universal constant after enlarging C_1. Since

$$\widehat{w} - w^\star = \Sigma_\rho^{-1/2} \widehat{G}_n^{-1} \left(\frac{1}{n} \sum_{i=1}^{n} g_i b_i \right),$$

a union bound yields universal $C_2 > 0$ such that

$$\|\widehat{w} - w^\star\|_2 \leq \frac{C_2 R_{\max}}{\sqrt{\lambda_{\min}(\Sigma_\rho)}} \sqrt{\frac{d + \log(2/\beta)}{n}}$$

with probability at least $1 - \beta$.

Why population isotropy alone is insufficient. Second moments alone do not guarantee finite-sample conditioning. Let

$$U = \begin{cases} 0, & \text{with probability } 1-p, \\ p^{-1/2}\xi, & \text{with probability } p, \end{cases} \qquad \xi \sim \mathcal{N}(0, I_d).$$

Then $\mathbb{E}[U] = 0$ and $\mathbb{E}[UU^\top] = I_d$, so the population distribution is exactly isotropic. Nevertheless, an n-sample empirical Gram matrix is identically zero with probability $(1-p)^n$. For $p \ll 1/n$, this failure probability is large. Thus population isotropy is enough for the geometric result in Theorem 4.1, but not for a finite-sample reward-fitting guarantee without an additional concentration assumption. Gaussianity is one convenient sufficient condition rather than a uniquely necessary one.

Consequence for terminal evaluation. Assume, over the states and codes covered by Theorem 4.3,

$$\sup_{x,z} \mathbb{E}_{x' \sim P(\cdot|x,\pi_z(x))} |b(x')| \leq \epsilon_{\text{app}}, \qquad \sup_{x,z} \|\widehat{F}_z(x, \pi_z(x))\|_2 \leq \epsilon_\psi.$$

Add and subtract $\widehat{F}_z(x,\pi_z(x))^\top w^\star$ and $V_{r_{w^\star}}^{\pi_z}(x)$. The coefficient bound above, Theorem 4.1, and the discounted sum of the one-step approximation error give

$$\left| \widehat{F}_z(x,\pi_z(x))^\top \widehat{w} - V_r^{\pi_z}(x) \right| \leq \frac{\epsilon_B \sqrt{(w^\star)^\top \Sigma_\rho w^\star}}{(1-\gamma)\sqrt{\lambda_{\min}(\Sigma_\rho)}} + \frac{\epsilon_{\text{app}}}{1-\gamma} + \epsilon_\psi \epsilon_{w,n} = \epsilon_{\text{term},n}^G.$$

Replacing the generic terminal term in the fixed-candidate proof of Theorem 4.3 by $\epsilon_{\text{term},n}^G$ yields the Gaussian-specialized bound stated in the main text. This is an alternative decomposition of the combined terminal error, not an additional penalty. Neither Gaussianity nor near-isotropy replaces the separate coverage, rollout-accuracy, or policy-family assumptions used elsewhere in the theory.

B.3 Proof of Theorem 4.2: Code Distance and Policy Value

Discounted coverage For a starting state s and policy $\pi \in \Pi$, define the normalized discounted next-state distribution

$$d_s^\pi(B) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr_s^\pi(s_{t+1} \in B). \tag{25}$$

Assume $d_s^\pi \ll \rho$, let $q_s^\pi = \mathrm{d}d_s^\pi/\mathrm{d}\rho \in L_2(\rho)$, and define

$$C_\rho(s) = \sup_{\pi,\pi' \in \Pi} \left\| q_s^\pi - q_s^{\pi'} \right\|_{L_2(\rho)} < \infty. \tag{26}$$

For example, a bound $\sup_{\pi \in \Pi} \|q_s^\pi\|_{L_2(\rho)} \leq C_0(s)$ implies $C_\rho(s) \leq 2C_0(s)$. This is a coverage assumption on the policy family; it is separate from representation regularization.

Reward geometry induced by the representation Let $\Sigma_\rho = \mathbb{E}_\rho[\phi(s)\phi(s)^\top]$, as in the main text, and for two linear reward coefficients $w_1, w_2 \in \mathbb{R}^d$ define

$$D_\rho(w_1, w_2)^2 = \mathbb{E}_\rho\big[(r_{w_1}(s) - r_{w_2}(s))^2\big] = (w_1 - w_2)^\top \Sigma_\rho (w_1 - w_2). \tag{27}$$

Hence

$$\sqrt{\lambda_{\min}(\Sigma_\rho)}\,\|w_1 - w_2\|_2 \le D_\rho(w_1, w_2) \le \sqrt{\lambda_{\max}(\Sigma_\rho)}\,\|w_1 - w_2\|_2, \tag{28}$$

where the lower bound may be zero if Σ_ρ is singular. Under the near-isotropy condition $\|\Sigma_\rho - I\|_{\mathrm{op}} \le \delta < 1$, we have

$$\sqrt{1-\delta}\,\|w_1 - w_2\|_2 \le D_\rho(w_1, w_2) \le \sqrt{1+\delta}\,\|w_1 - w_2\|_2. \tag{29}$$

The matrix Σ_ρ is the uncentered second moment. If $\mu_\rho = \mathbb{E}_\rho[\phi(s)]$, then $\Sigma_\rho = \mathrm{Cov}_\rho(\phi) + \mu_\rho \mu_\rho^\top$. Thus the zero-mean and identity-covariance targets of the Gaussian regularizer motivate the near-isotropy assumption, although a finite empirical regularization loss alone does not imply a particular population value of δ.

From reward geometry to policy values By Eq. 25, for any coefficient $w \in \mathbb{R}^d$,

$$V^\pi_{r_w}(s) = \frac{1}{1-\gamma} \int \phi(x)^\top w \, q^\pi_s(x)\, \rho(\mathrm{d}x). \tag{30}$$

Therefore, for any $\pi, \pi' \in \Pi$, Cauchy–Schwarz gives

$$\begin{aligned} \left|V^\pi_{r_w}(s) - V^{\pi'}_{r_w}(s)\right| &= \frac{1}{1-\gamma} \left| \int \phi(x)^\top w \left(q^\pi_s(x) - q^{\pi'}_s(x)\right) \rho(\mathrm{d}x)\right| \\ &\le \frac{C_\rho(s)}{1-\gamma} \sqrt{w^\top \Sigma_\rho w}. \end{aligned} \tag{31}$$

Proof of Theorem 4.2 Recall

$$\epsilon_{\mathrm{pol}}(s, z) = V^\Pi_{r_z}(s) - V^{\pi_z}_{r_z}(s). \tag{32}$$

For any $z \in \mathcal{Z}$ and any comparison policy $\pi \in \Pi$, linearity of the reward gives

$$\begin{aligned} V^\pi_{r_{\hat{w}}}(s) - V^{\pi_z}_{r_{\hat{w}}}(s) &= \left[V^\pi_{r_{\hat{w}-z}}(s) - V^{\pi_z}_{r_{\hat{w}-z}}(s)\right] + \left[V^\pi_{r_z}(s) - V^{\pi_z}_{r_z}(s)\right] \\ &\le \frac{C_\rho(s)}{1-\gamma} \sqrt{(z-\hat{w})^\top \Sigma_\rho (z-\hat{w})} + \epsilon_{\mathrm{pol}}(s, z), \end{aligned} \tag{33}$$

where Eq. 31 bounds the first bracket and the definition of ϵ_{pol} bounds the second. Taking the supremum over $\pi \in \Pi$ yields the representation-dependent bound

$$V^{\pi_z}_{r_{\hat{w}}}(s) \ge V^\Pi_{r_{\hat{w}}}(s) - \frac{C_\rho(s)}{1-\gamma} \sqrt{(z-\hat{w})^\top \Sigma_\rho (z-\hat{w})} - \epsilon_{\mathrm{pol}}(s, z). \tag{34}$$

Finally, using Eq. 29 and $z_{\mathrm{pre}} = \hat{w}$ gives

$$V^{\pi_z}_{r_{\hat{w}}}(s) \ge V^\Pi_{r_{\hat{w}}}(s) - \frac{C_\rho(s)\sqrt{1+\delta}}{1-\gamma} \|z - z_{\mathrm{pre}}\|_2 - \epsilon_{\mathrm{pol}}(s, z), \tag{35}$$

which is Theorem 4.2.

What representation regularization controls Eq. 34 remains valid without isotropy. Near-isotropy makes the reward discrepancy induced by a code perturbation comparable to its Euclidean length in every direction. For the one-sided value guarantee alone, an upper spectral bound $\lambda_{\max}(\Sigma_\rho) \le B$ is already sufficient, giving coefficient $C_\rho(s)\sqrt{B}/(1-\gamma)$. The lower eigenvalue bound supplies the reverse comparison in Eq. 29, ruling out directions that move substantially in code space while changing the associated linear reward only weakly. This argument uses only second moments; full Gaussianity is not required for this theorem, and neither near-isotropy nor Gaussianity replaces the separate coverage condition in Eq. 26.

Example: why Euclidean code distance alone can be misleading Consider two actions from the starting state, leading deterministically to absorbing states x_+ and x_-, with

$$\phi(x_+) = (M, 1)^\top, \qquad \phi(x_-) = (-M, 1)^\top, \qquad M > 0, \tag{36}$$

and let ρ assign probability $1/2$ to each absorbing state. Then $\Sigma_\rho = \mathrm{diag}(M^2, 1)$. Suppose π_u selects x_+ when $u_1 \geq 0$ and x_- otherwise, so every π_u is optimal for its own linear reward. Set

$$z_{\mathrm{pre}} = \hat{w} = \begin{pmatrix} 1/(2M) \\ 1/2 \end{pmatrix}, \qquad z = \begin{pmatrix} -1/(2M) \\ 1/2 \end{pmatrix}. \tag{37}$$

The target reward is 1 at x_+ and 0 at x_-, while $\|z - z_{\mathrm{pre}}\|_2 = 1/M \to 0$. Nevertheless,

$$V^{\Pi}_{r_{\hat{w}}}(s) - V^{\pi_z}_{r_{\hat{w}}}(s) = \frac{1}{1-\gamma}, \qquad D_\rho(z, z_{\mathrm{pre}}) = 1. \tag{38}$$

Thus, across representations with uncontrolled spectra, arbitrarily small Euclidean code distance need not imply a uniformly small target-value deficit. This example illustrates a possibility rather than asserting that removing the regularizer necessarily produces such a representation.

B.4 Proof of Theorem 4.3: Value Bound for Test-Time Adjustment

The following proof is adapted from Cheng et al. (2025). We now compare the score used by test-time adjustment with the true value of each fixed candidate code. Recall the terminal SF evaluator from the main text,

$$\widehat{V}_{r_{\hat{w}}}(\hat{\phi}; z) = \psi\Big(\hat{\phi}, \pi_\eta(\hat{\phi}, z), z\Big)^\top \hat{w}. \tag{39}$$

For a fixed z, let the true rollout satisfy

$$s_0 = s, \qquad a_t = \pi_\eta(\phi(s_t), z), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t), \tag{40}$$

and let the imagined latent rollouts follow:

$$\hat{\phi}_0 = \phi(s), \qquad \hat{a}_t = \pi_\eta(\hat{\phi}_t, z), \qquad \hat{\phi}_{t+1} = f(\hat{\phi}_t, \hat{a}_t). \tag{41}$$

The score in Eq. 8 can then be written as

$$\widehat{J}_H(s, z) = \sum_{t=0}^{H-1} \gamma^t \hat{r}_\chi(\hat{\phi}_{t+1}) + \gamma^H \widehat{V}_{r_{\hat{w}}}(\hat{\phi}_H; z). \tag{42}$$

We restate the assumptions here.

Reward approximation errors The fitted linear reward and the nonlinear reward predictor have uniformly bounded one-step expected errors under the true dynamics:

$$\begin{aligned} \sup_{x,z} \mathbb{E}_{x' \sim P(\cdot|x, \pi_z(x))} \left[|r_{\hat{w}}(x') - r(x')| \right] &\leq \epsilon_{\mathrm{lin}}, \\ \sup_{x,z} \mathbb{E}_{x' \sim P(\cdot|x, \pi_z(x))} \left[|\widehat{r}_\chi(\phi(x')) - r(x')| \right] &\leq \epsilon_{\mathrm{nl}}, \end{aligned} \tag{43}$$

The first condition measures the mismatch introduced by representing the downstream reward linearly in the learned feature space, whereas the second measures the prediction error of the nonlinear reward model used to score the imagined rollout prefix.

SF approximation errors The SF approximation error is measured at the true encoded state:

$$\left| \widehat{V}_{r_{\hat{w}}}(\phi(x); z) - V^{\pi_z}_{r_{\hat{w}}}(x) \right| \leq \epsilon_{\mathrm{SF}}. \tag{44}$$

Thus, ϵ_{SF} measures the error of the learned SF terminal evaluator relative to the exact value of the same policy π_z under the fitted linear reward $r_{\hat{w}}$ in the true environment.

Latent rollout errors For each s and z, suppose the latent rollout and a true trajectory (s_t) of the fixed policy π_z, starting at $s_0 = s$, admit a coupling such that, for $H \geq 1$,

$$\max_{1\leq t\leq H} \mathbb{E}\Big[\|\hat{\phi}_t - \phi(s_t)\|_2\Big] \leq \epsilon_s(H), \qquad \epsilon_s(0) = 0. \tag{45}$$

The true trajectory follows $\pi_z(s_t)$ at every step; it does not simply replay the actions generated by the imagined trajectory. The quantity $\epsilon_s(H)$ is a rollout-level error bound: it directly controls the discrepancy between the imagined latent trajectory and the encoded true trajectory over the planning horizon.

B.4.1 Evaluation error for a fixed candidate

The true value admits the horizon-H decomposition

$$V_r^{\pi_z}(s) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(s_{t+1}) + \gamma^H V_r^{\pi_z}(s_H)\right]. \tag{46}$$

For a prefix term, compare $r(s_{t+1})$ with $\hat{r}_\chi$. By the L_r-Lipschitz assumption and the nonlinear reward error in Theorem 4.3,

$$\begin{aligned}
&\Big|\mathbb{E}\Big[\widehat{r}_\chi(\widehat{\phi}_{t+1}) - r(s_{t+1})\Big]\Big| \\
&\quad\leq L_r\, \mathbb{E}\Big[\|\widehat{\phi}_{t+1} - \phi(s_{t+1})\|_2\Big] + \epsilon_{\mathrm{nl}} \\
&\quad\leq L_r \epsilon_s(H) + \epsilon_{\mathrm{nl}}.
\end{aligned} \tag{47}$$

For the terminal term, compare with $\widehat{V}_{r_{\hat{w}}}(\phi(s_H); z)$ and insert $V_{r_{\hat{w}}}^{\pi_z}(s_H)$ subsequently. The Lipschitz assumption and the SF evaluation error give

$$\begin{aligned}
&\Big|\mathbb{E}\Big[\widehat{V}_{r_{\hat{w}}}(\hat{\phi}_H; z) - V_r^{\pi_z}(s_H)\Big]\Big| \\
&\quad\leq L_V \epsilon_s(H) + \epsilon_{\mathrm{SF}} + \mathbb{E}\big[\big|V_{r_{\hat{w}}}^{\pi_z}(s_H) - V_r^{\pi_z}(s_H)\big|\big].
\end{aligned} \tag{48}$$

The one-step linear reward error implies, for any state x,

$$\begin{aligned}
\big|V_{r_{\hat{w}}}^{\pi_z}(x) - V_r^{\pi_z}(x)\big| &\leq \sum_{k=0}^{\infty} \gamma^k \mathbb{E}[|r_{\hat{w}}(s_{k+1}) - r(s_{k+1})|] \\
&\leq \frac{\epsilon_{\mathrm{lin}}}{1-\gamma}.
\end{aligned} \tag{49}$$

Combining Eqs. 47–49 with Eq. 46 yields, uniformly over the relevant candidate codes,

$$\Big|\widehat{J}_H(s,z) - V_r^{\pi_z}(s)\Big| \leq E_H, \tag{50}$$

where

$$E_H = \frac{1-\gamma^H}{1-\gamma}\epsilon_{\mathrm{nl}} + \left(\frac{1-\gamma^H}{1-\gamma}L_r + \gamma^H L_V\right)\epsilon_s(H) + \gamma^H\left(\epsilon_{\mathrm{SF}} + \frac{\epsilon_{\mathrm{lin}}}{1-\gamma}\right). \tag{51}$$

This is exactly the error quantity used in Eq. (12). In particular, $\epsilon_s(H)$ is already a rollout-level discrepancy up to horizon H; no additional accumulation factor should be multiplied onto it in Eq. 51.

B.4.2 From candidate evaluation to the selected-code bound

Let $\widehat{z}_H \in \arg\max_{z\in\mathcal{Z}} \widehat{J}_H(s,z)$, as in Theorem 4.3. The definition

$$\Gamma_\Pi(s) = V_r^*(s) - V_r^\Pi(s) \tag{52}$$

separates the gap between unrestricted optimal control and the best value available within the pre-trained policy family from the error introduced by the learned candidate evaluator.

The supremum defining $V_r^{\Pi}(s)$ need not be attained. For arbitrary $\xi > 0$, choose $z_\xi \in \mathcal{Z}$ such that

$$V_r^{\pi_{z_\xi}}(s) \geq V_r^{\Pi}(s) - \xi. \tag{53}$$

By optimality of $\hat{z}_H$ for the learned score and Eq. 50,

$$\begin{aligned} V_r^{\Pi}(s) - V_r^{\pi_{\hat{z}_H}}(s) &\leq \xi + V_r^{\pi_{z_\xi}}(s) - V_r^{\pi_{\hat{z}_H}}(s) \\ &= \xi + \left[V_r^{\pi_{z_\xi}}(s) - \widehat{J}_H(s, z_\xi)\right] \\ &\quad + \left[\widehat{J}_H(s, z_\xi) - \widehat{J}_H(s, \hat{z}_H)\right] \\ &\quad + \left[\widehat{J}_H(s, \hat{z}_H) - V_r^{\pi_{\hat{z}_H}}(s)\right] \\ &\leq \xi + 2E_H. \end{aligned} \tag{54}$$

Letting $\xi \downarrow 0$ gives

$$V_r^{\Pi}(s) - V_r^{\pi_{\hat{z}_H}}(s) \leq 2E_H. \tag{55}$$

Finally,

$$\begin{aligned} V_r^*(s) - V_r^{\pi_{\hat{z}_H}}(s) &= \left[V_r^*(s) - V_r^{\Pi}(s)\right] + \left[V_r^{\Pi}(s) - V_r^{\pi_{\hat{z}_H}}(s)\right] \\ &\leq \Gamma_{\Pi}(s) + 2E_H. \end{aligned} \tag{56}$$

Because $V_r^*(s) \geq V_r^{\pi_{\hat{z}_H}}(s)$, the left-hand side is nonnegative, so Eq. 56 is equivalent to the absolute-value form stated in Eq. (13). The factor 2 comes from the two possible candidate-evaluation errors: a high-value code can be underestimated while the selected code is overestimated, each by at most E_H.

Special case $H = 0$. When $H = 0$, the rollout prefix is empty and $\epsilon_s(0) = 0$, hence

$$E_0 = \epsilon_{\mathrm{SF}} + \frac{\epsilon_{\mathrm{lin}}}{1-\gamma}. \tag{57}$$

Thus the same argument covers SF-only code search, which just maximizes the SF value at the first step without any latent rollout. For $H > 0$, the terminal SF and linear-reward terms are multiplied by γ^H, while nonlinear reward and rollout errors enter through the prefix. This tradeoff explains why a longer rollout can tighten the bound when the learned dynamics and nonlinear reward model are accurate, without implying monotonic improvement for every horizon.

B.4.3 Relating rollout error to one-step dynamics error

Theorem 4.3 directly assumes the rollout-level quantity

$$\max_{1 \leq t \leq H} \mathbb{E}\left[\|\widehat{\phi}_t - \phi(s_t)\|_2\right] \leq \epsilon_s(H). \tag{58}$$

For completeness, a stronger lower-level assumption on one-step prediction can imply such a bound. Suppose

$$\sup_{s,z} \mathbb{E}\left[\|f(\phi(s), \pi_\eta(\phi(s), z)) - \phi(s')\|_2\right] \leq \epsilon_f, \tag{59}$$

where $s' \sim P(\cdot \mid s, \pi_z(s))$, and suppose the closed-loop latent map is L_f-Lipschitz:

$$\|f(x, \pi_\eta(x, z)) - f(y, \pi_\eta(y, z))\|_2 \leq L_f \|x - y\|_2. \tag{60}$$

Let $d_t = \mathbb{E}\|\widehat{\phi}_t - \phi(s_t)\|_2$. Since $d_0 = 0$,

$$d_{t+1} \leq L_f d_t + \epsilon_f, \tag{61}$$

and therefore

$$d_t \leq \epsilon_f \sum_{j=0}^{t-1} L_f^j. \tag{62}$$

A sufficient choice for the trajectory-level quantity in Theorem 4.3 is consequently

$$\epsilon_s(H) \le \epsilon_f \sum_{j=0}^{H-1} L_f^j. \tag{63}$$

Eq. 63 describes how one-step model errors can accumulate. By contrast, $\epsilon_s(H)$ in the theorem already measures the resulting multi-step discrepancy, which is why it appears only once in Eq. 51.

If per-step discrepancies d_t are available, the fixed-candidate bound can be kept sharper instead of replacing every d_t by the common worst case $\epsilon_s(H)$:

$$\left|\widehat{J}_H(s,z) - V_r^{\pi_z}(s)\right| \le \frac{1-\gamma^H}{1-\gamma}\epsilon_{\mathrm{nl}} + L_r \sum_{t=0}^{H-1} \gamma^t d_{t+1} + \gamma^H \left(L_V d_H + \epsilon_{\mathrm{SF}} + \frac{\epsilon_{\mathrm{lin}}}{1-\gamma}\right). \tag{64}$$

Substituting Eq. 62 into Eq. 64 gives the corresponding bound directly in terms of ϵ_f and L_f, but the rollout-level form in Theorem 4.3 is shorter and separates model-error propagation from candidate evaluation.

C Extended discussion for related work

Successor-measure approaches model discounted occupancies, with standard FB representations using forward and backward factors to infer reward-conditioned policies (Touati & Ollivier, 2021). Conservative objectives address low-quality offline data (Jeen et al., 2024), while regularized latent dynamics prediction offers an alternative route to features for successor-measure models (Jajoo et al., 2026). These developments address learning and data issues; they do not, by themselves, establish that a fixed finite-dimensional reward encoding retains every control-relevant reward distinction.

To make this distinction explicit, consider a fixed backward representation $B : \mathcal{S} \to \mathbb{R}^d$ and the standard reward code

$$w_r = \mathbb{E}_{s\sim\rho}[B(s)r(s)]. \tag{65}$$

The following is a direct consequence of this encoding: for any perturbation h satisfying $\mathbb{E}_{s\sim\rho}[B(s)h(s)] = 0$, we have $w_{r+h} = w_r$. Thus different rewards can be assigned the same code, as discussed in critiques of fixed linear reward encodings (Agarwal et al., 2025; Cetin et al., 2025). This is a control limitation only when the discarded distinctions matter for policy selection. In particular, if the two rewards have no common optimal policy, a task-to-policy map depending only on their shared code cannot be optimal for both. Conversely, non-recoverability of the reward alone does not imply suboptimal control.

Exact reward recovery also requires a representability assumption. If $r(s) = B(s)^\top w$ and $\Sigma_B = \mathbb{E}_{s\sim\rho}[B(s)B(s)^\top]$ is invertible, then

$$w_r = \Sigma_B w, \qquad r(s) = B(s)^\top \Sigma_B^{-1} z_r. \tag{66}$$

The simpler identity $r(s) = B(s)^\top w_r$ follows under whitening, $\Sigma_B = I$; and now r can be recovered by z_r to avoid aliasing but it requires the linear structure once again. Autoregressive features explicitly relax the fixed linear-encoding restriction (Cetin et al., 2025). The preceding argument therefore concerns standard fixed-feature FB, not every extension of the FB family.

An alternative formulation represents the behavior space using an affine successor-measure basis (PSM; Agarwal et al., 2025). Its idealized complete-basis construction should not be equated with the SF assumption that rewards are linear in fixed state features. The practical issue is basis capacity: a limited-dimensional approximation can omit task-relevant occupancies, whereas a larger basis increases representation and optimization costs. This is a limitation of approximate behavior-space coverage, not a blanket restriction to linear reward functions.

Nonlinear task embeddings learned from state–reward samples provide another route to zero-shot policies (FRE; Frans et al., 2024). Pretraining uses a chosen distribution over reward functions, including randomized goal-reaching, linear, and neural-network rewards, rather than only a finite

list of fixed tasks. Accordingly, practical limitations concern the coverage of the pretraining reward prior, information available in the sampled reward encoding, and encoder capacity. Nonlinear encoding avoids imposing a fixed linear state-reward span, but does not itself guarantee generalization to arbitrary downstream rewards.

The preceding limitations arise at different levels: reward encoding, behavior coverage, and generalization under a reward prior. They motivate investigating test-time adjustment beyond SF reward linearization, without establishing that all exact ZSRL constructions necessarily require adjustment.

The search of our framework remains restricted to the pretrained policy family. It cannot by itself supply missing behaviors or recover information absent from all learned representations, and its effectiveness depends on the accuracy of the latent dynamics and reward/value estimates. Our claim is therefore that adjustment can address residual policy-selection errors within the available family, not that it universally removes representation or coverage limitations.

D Experimental Details

D.1 Datasets and Offline Pretraining Regime

We use the standard ExORL datasets (Yarats et al., 2022), which are collected by reward-free exploration policies and contain long trajectories from the DeepMind Control Suite. The original ExORL release contains approximately 10M environment transitions for each domain and exploration method. For state-based experiments, we use the first $5{,}000$ trajectories in the released ordering, corresponding to approximately 5M transitions. For pixel-based experiments, due to the computation constraint, we use the first $1{,}000$ trajectories, corresponding to approximately 1M transitions. All methods compared under the same observation modality are trained on exactly the same subset.

Our pretraining objective does not include an additional conservative value penalty or behavior-cloning regularizer. This choice is motivated by the data regime considered here rather than by assuming that offline distribution shift is absent in general. ExORL showed that sufficiently diverse exploratory data can allow vanilla off-policy RL algorithms, without offline-specific modifications, to match or outperform dedicated offline-RL algorithms (Yarats et al., 2022). More recent zero-shot RL studies also show that the need for behavior regularization becomes increasingly important as coverage is deliberately reduced. In particular, TD-JEPA (Bagatella et al., 2026) studies ExORL subsets ranging from 100 to $5{,}000$ episodes and observes increasingly severe degradation in the small-data regime, where BC- and FQL-style regularization can mitigate extrapolation error.

Our setting remains substantially larger than the low-quality regime explicitly targeted by conservative zero-shot RL methods. For example, Jeen et al. (Jeen et al., 2024) construct their principal low-quality ExORL setting using 100K transitions. Our pixel and state datasets contain approximately $10\times$ and $50\times$ more transitions, respectively. At these scales, we did not observe a consistent need for an additional training-time conservative correction, and therefore keep the reward-free pretraining objective unchanged across all experiments.

Importantly, this observation only concerns pretraining on states and actions drawn from the fixed dataset. Test-time model-based search creates a different source of distribution shift: imagined trajectories may visit latent states that are poorly covered by the pretraining data and evaluation interaction itself can lead agent to unfamiliar states due to its intentionality. We address this issue separately at deployment time, as described next.

D.2 Uncertainty Regularization for Test-Time Search

Although the pretraining dataset is sufficiently large for our main training setting, test-time model rollouts may extrapolate beyond its coverage. A candidate behavior code can induce imagined states that are rare or absent in the offline data, in which case its terminal SF value may be overestimated. Following simple and general principle of test-time uncertainty regularization used in TD-MPC2 (Hansen et al., 2024), we penalize candidate behaviors whose SF-based value predictions exhibit high ensemble disagreement. Unlike TD-MPC2, we use a simple standard-deviation penalty rather than its task-adaptive value-scaled penalty.

Concretely, we implement the successor-feature estimator with an ensemble of E heads. For a candidate behavior code z, let

$$\hat{Q}^{(i)}_{\hat{w}}(\hat{\phi}, z) = \psi^{(i)}\Big(\hat{\phi}, \pi_\eta(\hat{\phi}, z), z\Big)^\top \hat{w}, \qquad i = 1, \dots, E,$$

denote the value predicted by ensemble member i for the fixed downstream reward coefficient $\hat{w}$. We compute

$$\bar{Q}_{\hat{w}}(\hat{\phi}, z) = \frac{1}{i}\sum_{e=1}^{i} \hat{Q}^{(i)}_{\hat{w}}(\hat{\phi}, z), \qquad \sigma_Q(\hat{\phi}, z) = \sqrt{\frac{1}{i}\sum_{i=1}^{E}\Big(\hat{Q}^{(i)}_{\hat{w}}(\hat{\phi}, z) - \bar{Q}_{\hat{w}}(\hat{\phi}, z)\Big)^2}.$$

The test-time score in Eq. 8 is then implemented as

$$\hat{J}^{\mathrm{unc}}_H(s_0, z) = \sum_{t=0}^{H-1} \gamma^t \hat{r}_\chi(\hat{\phi}_{t+1}) + \gamma^H \left[\bar{Q}_{\hat{w}}(\hat{\phi}_H, z) - \lambda_{\mathrm{unc}}\sigma_Q(\hat{\phi}_H, z)\right],$$

where $a_t = \pi_\eta(\hat{\phi}_t, z)$ and $\hat{\phi}_{t+1} = f(\hat{\phi}_t, a_t)$. Setting $\lambda_{\mathrm{unc}} = 0$ recovers the unregularized score used in the main derivation.

The uncertainty term is a stabilizer rather than a prerequisite for test-time adjustment. We find that code-space adjustment remains effective when $\lambda_{\mathrm{unc}} = 0$, while incorporating ensemble disagreement further improves its reliability and performance. To ensure that gains in our ablations are not caused by giving one search method a more conservative evaluator, we apply the same uncertainty mechanism to all test-time search variants, including GLDSF-TTA, TTA-0, TTA-uni, and the action-space planning baselines.

D.3 Network Architectures and Implementation Details

Observation and latent encoders For state observations, the environment state is passed directly to the learned representation network. For pixel observations, we first normalize pixel intensities to $[-0.5, 0.5]$ and use a convolutional encoder. For 64×64 observations, the encoder contains four 3×3 convolutional layers with 32 channels; the first layer uses stride 2 and the remaining layers use stride 1, with ReLU activations throughout. The flattened visual representation is subsequently processed by the same latent feature network used for state observations.

The latent feature network consists of two 512-unit hidden layers with LayerNorm and ReLU activations, followed by a d-dimensional output layer. Unless stated otherwise, we use $d = 50$. Gaussian regularization is implemented using the SIGReg-style characteristic-function discrepancy described in Appendix A. In implementation, the regularizer samples random one-dimensional projections of the feature batch and compares their empirical characteristic functions to that of a standard Gaussian.

Latent dynamics Actions are first mapped through a small action encoder consisting of a linear layer, a Tanh activation, and a second linear layer. The latent dynamics model takes the concatenation of the latent state and encoded action, and uses two 1024-unit hidden layers, each followed by LayerNorm and ReLU, to predict the next d-dimensional latent state. The same model is applied recurrently for the multi-step dynamics objective during pretraining and for imagined rollouts at test time. Stop-gradient targets are used for latent prediction.

Policy and successor features Both the actor and the successor-feature network architecture keep consistent with prior work (Park et al., 2024; Bagatella et al., 2026). They use separate preprocessing branches before combining state information with the behavior code. The actor separately embeds $\phi(s)$ and $[\phi(s), z]$, concatenates the two representations, and predicts a bounded action through a final Tanh output. During evaluation and model-based search we use the mean action of the resulting truncated-Gaussian policy.

The SF network similarly embeds $[\phi(s), a]$ and $[\phi(s), z]$ before concatenation. Its output layer is implemented as an ensemble of $E = 5$ vector-valued SF heads. During training, the actor maximizes the ensemble mean of the projected SF value. We maintain a slowly updated target SF network; for each target update, two target ensemble members are randomly selected and averaged to construct the bootstrapped SF target. At test time, the ensemble mean provides the terminal value and its disagreement provides the uncertainty estimate in Eq. D.2.

Task inference Given the reward-labeled deployment subset, we first solve the linear least-squares problem in Eq. 2. For numerical consistency with pretraining, policy-conditioning codes are constrained to the sphere of radius $\sqrt{d}$. If $\hat{w}_0$ denotes the unconstrained least-squares solution, we use $z_{\mathrm{pre}} = \sqrt{d}\,\frac{\hat{w}_0}{\|\hat{w}_0\|_2}$ for policy conditioning and retain the corresponding scale factor so that SF values remain equivalent to evaluation under the original $\hat{w}_0$. Candidate codes sampled during test-time search are projected onto the same sphere.

Pixel augmentation and target networks For visual observations we use random-shift augmentation. When processing a multi-step trajectory for the latent dynamics objective, the same sampled spatial shift is reused across the corresponding sequence so that temporal prediction is not confounded by independently sampled image translations. The SF and representation target networks are updated by exponential moving average.

Baselines For HILP and FB, we use the implementations provided in the official HILP zero-shot RL codebase: `https://github.com/seohongpark/HILP/tree/master/hilp_zsrl`. For TD-JEPA and RLDP, we use the implementations provided in the official TD-JEPA repository: `https://github.com/facebookresearch/td_jepa`.

Hyperpatameter The related hyperparameters are listed in Table 4.

Table 4: Hyperparameters used for pretraining and test-time adjustment.

Hyperparameter	**State**	**Pixel**
Training gradient steps	1,000,000	500,000
Batch size	1,024	512
Learning rate	1×10^{-4}	1×10^{-4}
Feature dimension d	50	50
Discount factor γ	0.98	0.98
Dynamics training horizon T	5	5
SF target EMA coefficient	0.001	0.001
Gaussian regularization coefficient	0.1, 0.005 (Quadruped)	0.1, 0.005 (Quadruped)
TTA rollout horizon H	3	3
Candidate codes N	256	256
Number of elites N_{elite}	32	32
Search iterations N_{iter}	6	6
Initial / maximum code std.	0.7	0.5
Search temperature β_{temp}	0.5	0.5
Uncertainty coefficient λ_{unc}	10	1

E Full Results of Tables 1 and 2

The specific per-task results are shown in Tables 5 and 6. Our results looks different from the results from TD-JEPA original paper, which may be caused by that we use different dataset type (of different collection RL algorithm) and different task to aggregate results. The specific dataset type and evaluation task are not disclosed in (Bagatella et al., 2026). Also, we use a relatively small pixel dataset as mentioned above, and our pretraining gradient steps is less, compared to Bagatella et al. (2026). We can ensure that all the comparative experiments presented here were conducted under identical conditions.

Based on these results, we observe that the gains achieved by our method vary across different tasks. For example, adjustment improves DIAYN–Walker–Walk from 481 to 763 in the state setting and APT–Cheetah–Walk Backward from 322 to 795 in the pixel setting, while some Quadruped settings decrease after adjustment. We therefore view test-time adjustment as an average improvement in policy selection rather than a monotonic improvement on every task.

Table 5: Per-task performance comparison across state-based datasets

Dataset	Domain	Task	HILP	FB	RLDP	TD-JEPA	GLDSF	GLDSF-TTA
RND	Walker	Flip	575 ± 39	555 ± 70	497 ± 24	538 ± 58	590 ± 33	555 ± 38
		Run	386 ± 23	414 ± 45	291 ± 39	358 ± 24	409 ± 42	397 ± 22
		Stand	865 ± 17	733 ±252	819 ± 43	386 ± 23	912 ± 16	948 ± 10
		Walk	833 ± 27	846 ± 25	706 ± 75	729 ± 37	680 ± 58	867 ± 49
	Cheetah	Run	253 ± 17	169 ± 31	88 ± 16	224 ± 29	155 ± 39	204 ± 36
		Run Backward	98 ± 45	172 ± 12	186 ± 53	285 ± 31	205 ± 78	264 ± 63
		Walk	898 ± 58	651 ± 82	385 ± 61	624 ±141	531 ± 41	745 ±132
		Walk Backward	469 ±235	758 ± 75	691 ±170	965 ± 21	724 ±283	863 ±100
	Quadruped	Jump	599 ± 75	601 ± 46	734 ± 33	648 ± 18	719 ± 9	803 ± 35
		Run	379 ± 22	375 ± 31	450 ± 61	415 ± 13	484 ± 10	494 ± 16
		Stand	713 ± 29	736 ± 56	961 ± 6	771 ± 33	860 ± 60	968 ± 7
		Walk	479 ± 37	524 ± 12	649 ±170	555 ± 50	589 ± 72	646 ± 40
	Jaco	Reach Bottom Left	14 ± 18	17 ± 8	1 ± 1	37 ± 16	31 ± 21	30 ± 11
		Reach Bottom Right	36 ± 50	21 ± 19	2 ± 2	55 ± 30	36 ± 23	19 ± 6
		Reach Top Left	6 ± 6	20 ± 16	7 ± 4	30 ± 9	21 ± 10	47 ± 27
		Reach Top Right	15 ± 12	22 ± 7	6 ± 3	39 ± 21	26 ± 9	34 ± 26
DIAYN	Walker	Flip	499 ± 46	98 ± 15	143 ± 29	329 ± 53	331 ± 82	351 ±109
		Run	179 ± 54	75 ± 3	100 ± 16	126 ± 34	130 ± 20	151 ± 11
		Stand	583 ±134	478 ± 14	588 ± 49	647 ±165	714 ±123	833 ± 19
		Walk	649 ± 46	86 ± 3	503 ±197	681 ± 41	481 ±224	763 ± 77
	Cheetah	Run	340 ± 18	297 ± 11	312 ± 75	344 ± 6	354 ± 14	348 ± 7
		Run Backward	260 ± 17	277 ± 15	243 ± 10	299 ± 28	254 ± 10	273 ± 13
		Walk	986 ± 2	980 ± 8	985 ± 4	989 ± 1	987 ± 1	987 ± 1
		Walk Backward	984 ± 1	929 ± 46	983 ± 1	985 ± 1	983 ± 1	985 ± 1
	Quadruped	Jump	767 ± 53	507 ± 56	627 ± 56	665 ± 76	577 ± 44	588 ± 32
		Run	469 ± 10	319 ± 31	364 ± 43	387 ± 45	385 ± 32	398 ± 58
		Stand	959 ± 5	599 ± 54	836 ± 27	829 ± 65	835 ± 73	868 ± 50
		Walk	410 ± 75	366 ± 35	302 ± 10	429 ± 26	440 ± 49	520 ± 7
	Jaco	Reach Bottom Left	22 ± 6	11 ± 11	5 ± 3	7 ± 3	8 ± 4	22 ± 20
		Reach Bottom Right	16 ± 6	19 ± 27	4 ± 4	15 ± 9	25 ± 26	10 ± 9
		Reach Top Left	9 ± 10	32 ± 21	5 ± 3	19 ± 15	57 ± 47	43 ± 38
		Reach Top Right	61 ± 16	14 ± 4	6 ± 1	23 ± 5	30 ± 16	23 ± 9
APT	Walker	Flip	639 ± 12	415 ± 88	479 ± 45	598 ± 32	610 ± 13	514 ± 87
		Run	405 ± 27	346 ± 35	313 ± 28	395 ± 19	404 ± 20	333 ± 53
		Stand	878 ± 16	741 ±174	888 ± 19	911 ± 23	881 ± 23	902 ± 25
		Walk	856 ± 37	903 ± 30	767 ± 30	815 ± 50	850 ± 23	917 ± 15
	Cheetah	Run	254 ± 24	220 ± 43	264 ± 73	366 ±102	314 ± 35	375 ± 16
		Run Backward	135 ±111	98 ± 33	208 ± 77	339 ± 19	117 ± 58	102 ± 41
		Walk	785 ± 93	583 ± 83	821 ±108	881 ±152	912 ± 95	972 ± 18
		Walk Backward	520 ±349	445 ±139	729 ±194	980 ± 5	495 ±274	480 ±242
	Quadruped	Jump	719 ± 23	594 ± 53	631 ± 24	623 ± 92	722 ± 32	746 ± 32
		Run	462 ± 21	357 ± 79	436 ± 26	408 ± 28	455 ± 18	475 ± 5
		Stand	941 ± 10	620 ±247	907 ± 17	798 ± 49	909 ± 27	930 ± 20
		Walk	503 ±123	314 ±146	385 ± 64	539 ±153	432 ± 78	457 ± 28
	Jaco	Reach Bottom Left	11 ± 12	4 ± 4	3 ± 2	23 ± 19	15 ± 10	17 ± 17
		Reach Bottom Right	5 ± 5	12 ± 11	6 ± 6	14 ± 2	8 ± 4	14 ± 4
		Reach Top Left	7 ± 9	12 ± 11	5 ± 3	17 ± 13	49 ± 14	54 ± 14
		Reach Top Right	2 ± 1	6 ± 5	12 ± 9	38 ± 19	67 ± 28	55 ± 28

Table 6: Per-task performance comparison across pixel-based datasets

Dataset	Domain	Task	HILP	FB	RLDP	TD-JEPA	GLDSF	GLDSF-TTA
RND	Walker	Flip	239 ± 38	91 ± 20	280 ± 25	387 ± 11	353 ± 46	388 ± 36
		Run	113 ± 2	73 ± 22	192 ± 3	217 ± 12	198 ± 12	201 ± 27
		Stand	558 ± 23	362 ±101	732 ± 35	699 ± 75	691 ± 97	729 ± 49
		Walk	278 ± 13	248 ± 61	555 ± 27	543 ± 6	506 ± 40	589 ± 26
	Cheetah	Run	208 ± 23	109 ± 38	252 ±154	226 ± 57	231 ± 27	294 ± 45
		Run Backward	263 ± 51	45 ± 7	56 ± 40	315 ± 13	223 ± 69	296 ± 20
		Walk	662 ± 64	381 ± 82	732 ±206	804 ± 68	811 ± 41	893 ± 29
		Walk Backward	901 ± 86	206 ± 28	340 ±283	931 ± 30	850 ±102	964 ± 15
	Quadruped	Jump	225 ± 30	263 ± 58	332 ± 56	212 ± 49	424 ± 83	409 ± 61
		Run	153 ± 22	192 ± 44	232 ± 31	182 ± 23	314 ± 63	265 ± 47
		Stand	305 ± 31	372 ± 76	442 ± 37	321 ± 73	579 ±110	615 ±217
		Walk	204 ± 40	191 ± 49	231 ± 42	154 ± 31	339 ± 27	317 ± 59
	Jaco	Reach Bottom Left	47 ± 9	31 ± 8	1 ± 1	36 ± 7	40 ± 9	47 ± 18
		Reach Bottom Right	52 ± 14	52 ± 13	1 ± 1	29 ± 10	32 ± 9	36 ± 18
		Reach Top Left	39 ± 13	61 ± 15	8 ± 6	20 ± 8	61 ± 19	54 ± 27
		Reach Top Right	38 ± 17	60 ± 34	2 ± 2	31 ± 3	50 ± 15	43 ± 26
DIAYN	Walker	Flip	213 ± 7	117 ± 12	129 ± 4	225 ± 27	186 ± 46	292 ± 55
		Run	124 ± 34	66 ± 10	87 ± 3	113 ± 10	106 ± 18	116 ± 24
		Stand	543 ± 18	368 ± 63	511 ± 15	615 ±117	542 ± 72	531 ±119
		Walk	495 ± 6	114 ± 25	192 ±117	467 ±135	519 ± 73	584 ± 37
	Cheetah	Run	259 ± 26	129 ± 48	167 ±149	188 ± 54	382 ± 44	445 ± 59
		Run Backward	141 ±138	15 ± 6	65 ± 87	79 ± 34	298 ± 6	244 ± 95
		Walk	922 ± 20	667 ±116	502 ±358	595 ± 74	912 ± 52	939 ± 55
		Walk Backward	523 ±490	68 ± 24	311 ±413	539 ±231	974 ± 13	843 ±243
	Quadruped	Jump	114 ± 27	75 ± 18	72 ± 24	99 ± 18	126 ± 17	62 ± 18
		Run	69 ± 11	52 ± 9	54 ± 16	66 ± 11	119 ± 17	76 ± 49
		Stand	154 ± 48	103 ± 32	105 ± 23	156 ± 34	235 ± 42	186 ±123
		Walk	66 ± 25	51 ± 14	67 ± 13	55 ± 9	146 ± 29	69 ± 40
	Jaco	Reach Bottom Left	61 ± 22	47 ± 17	18 ± 11	36 ± 14	47 ± 15	57 ± 32
		Reach Bottom Right	51 ± 21	27 ± 17	42 ± 10	48 ± 11	47 ± 24	46 ± 28
		Reach Top Left	42 ± 7	31 ± 8	71 ± 19	44 ± 15	53 ± 7	76 ± 20
		Reach Top Right	29 ± 12	18 ± 9	81 ± 13	48 ± 19	44 ± 19	52 ± 14
APT	Walker	Flip	336 ± 42	119 ± 27	249 ± 45	400 ± 16	310 ± 56	330 ± 91
		Run	177 ± 6	77 ± 6	181 ± 12	224 ± 9	190 ± 4	201 ± 4
		Stand	609 ± 66	480 ± 23	943 ± 27	818 ± 21	793 ± 41	807 ± 63
		Walk	619 ± 35	116 ± 14	503 ±188	633 ± 36	601 ± 43	595 ± 36
	Cheetah	Run	283 ± 25	197 ± 29	85 ± 12	383 ± 64	305 ± 56	370 ± 31
		Run Backward	134 ± 68	48 ± 21	103 ±129	80 ± 35	70 ± 27	224 ± 91
		Walk	819 ± 21	543 ± 52	287 ± 88	891 ± 75	874 ± 81	972 ± 5
		Walk Backward	522 ±220	233 ±102	378 ±442	476 ±177	325 ±125	795 ±204
	Quadruped	Jump	227 ± 24	239 ± 36	109 ± 39	307 ± 18	497 ± 53	487 ± 99
		Run	136 ± 8	150 ± 25	79 ± 25	183 ± 23	333 ± 25	285 ± 76
		Stand	280 ± 23	315 ± 75	154 ± 67	359 ± 61	636 ± 63	717 ±102
		Walk	135 ± 19	138 ± 23	76 ± 27	185 ± 17	337 ± 21	290 ± 65
	Jaco	Reach Bottom Left	19 ± 11	7 ± 3	0 ± 0	36 ± 7	30 ± 11	26 ± 22
		Reach Bottom Right	24 ± 6	22 ± 14	1 ± 1	16 ± 7	23 ± 8	30 ± 17
		Reach Top Left	39 ± 9	66 ± 27	1 ± 1	49 ± 11	32 ± 17	52 ± 32
		Reach Top Right	40 ± 21	72 ± 26	2 ± 1	23 ± 14	48 ± 23	45 ± 41