LEARNING REUSABLE SPACES OF FEEDBACK POLICIES

Adam Thorpe, Tyler Ingebrand, Mason Russell, Ufuk Topcu
The University of Texas at Austin
Austin, TX 78712, USA

ABSTRACT

Learning policies that combine diverse skills remains a challenge for robotics and embodied systems, particularly when a controller must reuse and adapt learned behaviors across new tasks. Existing approaches learn latent or task-conditioned policies and behavioral priors for diverse skills, but they do not explicitly learn a shared functional representation of the underlying feedback policies. We present a method for learning a low-dimensional space of feedback policies, where each policy is represented by a small coefficient vector, enabling identification, synthesis, interpolation, and analysis. Specifically, we learn neural basis functions from expert trajectories to span a low-dimensional subspace of feedback policies. The linear coefficient structure reduces policy identification to coefficient estimation, policy synthesis to low-dimensional optimization, and interpolation to direct operations in coefficient space. Across humanoid, quadruped, and planar-quadrotor control, the learned space identifies held-out policies from sparse demonstrations, supports synthesis and interpolation, and enables faster, more accurate adaptation than prior methods while maintaining competitive closed-loop performance.

1 INTRODUCTION

A central challenge in multi-task control is reusing learned behaviors rather than solving a new high-dimensional control problem for every task. Policies for related tasks typically share substantial structure, varying along a small number of underlying directions, which suggests that a family of behaviors could be captured by a compact representation. Yet most learning-based approaches place structure on the task rather than on the policies themselves—task-conditioned policies require a well-defined task representation (Borsa et al., 2018; Touati & Ollivier, 2021; Liu et al., 2022), behavior cloning learns individual policies without exposing a reusable policy space (Torabi et al., 2018), policy optimization operates in computationally demanding high-dimensional parameter spaces (Wang et al., 2024), and motion-capture models require a target trajectory for every behavior (Peng et al., 2018). As a result, policy identification, control synthesis, policy interpolation, and closed-loop geometric analysis are handled by separate mechanisms. What is missing is a single low-dimensional representation of feedback policies that supports all of these operations.

We propose a learned coordinate system for families of feedback policies. We fit neural basis functions to expert demonstrations and represent each policy with a small coefficient vector. After training, we freeze the basis. Then, in these coordinates, policy identification reduces to estimating coefficients from state–action data, and control synthesis reduces to optimizing coefficients for a new objective. The linear structure also makes interpolation between policies straightforward and induces a local geometry that enables closed-loop analysis (Figure 1).

Our main contribution is an explicit linear representation of a policy space, learned from demonstrations and used directly for control. Unlike task-conditioned, meta-learned, or parameter-subspace approaches, the same policy coordinates support both inference from data and synthesis from objectives. Specifically: (i) We learn a low-dimensional linear space of feedback policies from independently trained experts to be embedded without a shared network parametrization. (ii) We show that these coordinates unify policy identification, synthesis, interpolation, and closed-loop analysis as operations on a single representation. (iii) Across humanoid, quadruped, planar-quadrotor, and linear-quadratic systems, we demonstrate that the learned space supports held-out policy identifica-

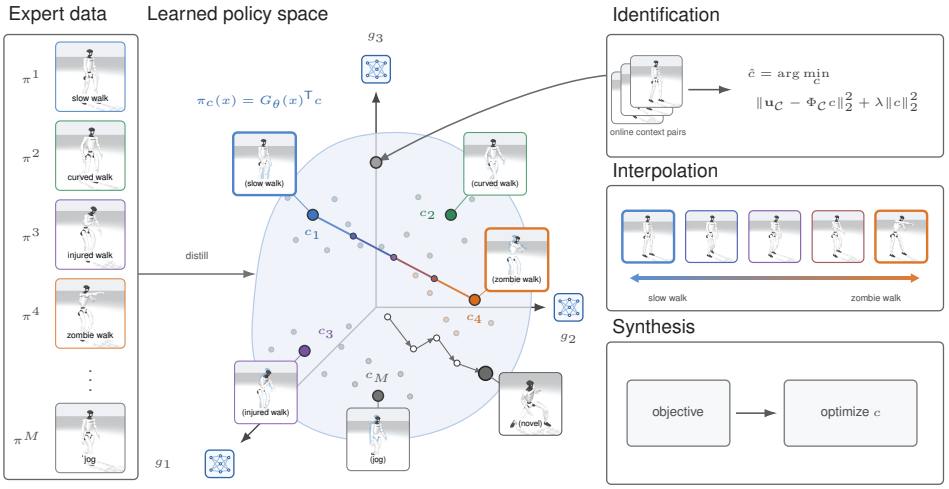


Figure 1: Overview of the proposed workflow. Offline demonstrations from related control tasks define a shared space of feedback policies. At deployment, we identify a policy by projection with a short state–action context and we synthesize new policies by coefficient-space synthesis.

tion, objective-driven synthesis, and smooth interpolation between behaviors while retaining strong closed-loop performance.

2 Related Work

Task and latent conditioning. Task- and latent-conditioning methods condition a value function or policy on an explicit task description, such as a goal, reward, or context variable (Schaul et al., 2015; Barreto et al., 2018; Touati & Ollivier, 2021; Liu et al., 2022; Sodhani et al., 2021), or learn a latent representation of the task or behavior from demonstrations, interaction, or motion data (Hausman et al., 2018; Rakelly et al., 2019; Duan et al., 2017; Pirotta et al., 2024; Tirinzoni et al., 2025; Peng et al., 2022; Luo et al., 2024; Tessler et al., 2023). These representations allow one model to express multiple behaviors, but the training objective usually provides learned latent variables with little mathematical structure. The latents may separate behaviors well, but their directions and distances do not have intrinsic meaning, which makes interpolation and comparison difficult to interpret. Some methods explicitly impose policy-space structure through low-dimensional parameter subspaces, parameter composition, or mixtures of experts (Wortsman et al., 2021; Gaya et al., 2022; 2023; Sun et al., 2022; Yang et al., 2020). These methods make the adaptation space explicit, but their coordinates still act on network weights or whole policies. This choice makes coordinate inference from state–action data, direction interpretation, and policy-level interpolation more difficult. Instead, we explicitly learn a subspace of feedback policies that encodes structure directly.

Trajectory basis representations. Movement primitives go a step further and represent behavior directly in a shared coefficient space. Dynamic and probabilistic movement primitives express trajectories as weighted combinations of basis functions, which enables fitting, interpolation, optimization, and uncertainty in the same coordinates (Ijspeert et al., 2013; Paraschos et al., 2013; Kober & Peters, 2011). This construction gives the representation explicit mathematical structure over time or phase rather than over the state-feedback policy. We use the same idea of a shared coefficient space but learn it directly for feedback policies.

Learned function-space representations. Function encoders (FE) learn a shared set of basis functions for a family of functions and recover each member through a small coefficient vector (Ingebrand et al., 2025a). Prior work uses this idea for reward and transition functions, predictive control, and vision-language-action policies (Ingebrand et al., 2025b; Iqbal et al., 2026; Zhao et al., 2025). In these settings, prior methods mainly use the coefficients to identify a function or condition another model (Ingebrand et al., 2025b; Iqbal et al., 2026). Here, we use the learned function space directly as the policy representation. The same coordinates define a feedback policy, support estimation from state–action data, enable optimization for new objectives and interpolation between

policies, and expose variation and uncertainty across the policy family. Our ridge objective relates to learned-feature linear-head meta-learning (Bertinetto et al., 2019; Lee et al., 2019; Harrison et al., 2018; Raghu et al., 2020), but our matrix-valued head parameterizes feedback policies and supports synthesis and geometric analysis.

Policy synthesis in low-dimensional spaces. Direct policy search can optimize a controller from rollout cost using methods such as cross-entropy optimization, evolution strategies, Bayesian optimization, or random search (Mannor et al., 2003; Salimans et al., 2017; Calandra et al., 2016; Marco et al., 2016; Mania et al., 2018). However, these optimization approaches work best when a small number of variables describe the policy, since a large parameter space can make the search intractable. By reducing the optimization to a small coefficient vector, we enable direct search over a learned space of feedback policies.

3 Learning a Shared Policy Space from Expert Policies

Consider a continuous-time controlled dynamical system $\dot{x}(t) = F\big(x(t), u(t)\big)$, where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ denotes the state, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ denotes the control input, and $F : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_x}$ defines the system dynamics. We assume that F and the admissible control laws satisfy the regularity conditions required for the closed-loop system to admit a unique solution over the time horizon of interest.

A deterministic state-feedback policy is a map $\pi : \mathcal{X} \to \mathcal{U}$ that induces the control law $u(t) = \pi\big(x(t)\big)$. Let Π denote the set of admissible feedback policies. For notational simplicity, we restrict attention to stationary policies, though our formulation extends directly to time-dependent feedback policies $\pi : \mathcal{X} \times [0, T] \to \mathcal{U}$. A control task $\tau \in \mathcal{T}$ is specified by a finite horizon $T_\tau > 0$, an initial-state distribution $\rho_{0,\tau}$, a running cost $\ell_\tau : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, and a terminal cost $\Phi_\tau : \mathcal{X} \to \mathbb{R}$. Given dynamics F, task τ, and policy $\pi \in \Pi$, define the expected finite-horizon cost

$$J_{F,\tau}(\pi) = \mathbb{E}_{x_0 \sim \rho_{0,\tau}} \left[\Phi_\tau\big(x(T_\tau)\big) + \int_0^{T_\tau} \ell_\tau\big(x(t), \pi(x(t))\big)\, dt \right], \tag{1}$$

where $x(0) = x_0$ and $x(\cdot)$ is the closed-loop trajectory. The optimal feedback policy for the system F and task τ is any solution of $\pi^*_{F,\tau} \in \arg\min_{\pi \in \Pi} J_{F,\tau}(\pi)$.

Directly optimizing over the full policy class is computationally demanding, particularly when each task $\tau \in \mathcal{T}$ requires a new policy. We therefore seek to use control behaviors observed across previously encountered tasks to learn a compact policy representation. We learn this representation offline and subsequently use it to reduce the dimensionality of the online adaptation problem.

Learning a Subspace of Useful Policies. During offline training, we assume access to N expert datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$. Each dataset $\mathcal{D}_i = \big\{\big(x_{i,m}, u_{i,m}\big)\big\}_{m=1}^{M_i}$ contains state–action observations from one or more demonstrations generated by expert policy π_i^{E}. We do not require the objective used to train each expert, and the expert may be suboptimal. We require only demonstrations of behaviors relevant to the target task family.

Our objective is to encode these expert policies into a useful subspace. Let $\mathcal{H}$ be a Hilbert space of functions from $\mathcal{X}$ to $\mathbb{R}^{n_u}$. Our objective is to learn a subspace $\mathcal{V} \subset \mathcal{H}$ such that $\mathcal{V}$ contains policies useful for as yet unspecified tasks. In particular, let $g_1(\,\cdot\,;\theta), \ldots, g_k(\,\cdot\,;\theta)$ be neural basis functions, and define $\mathcal{V} = \operatorname{span}\{g_1(\,\cdot\,;\theta), \ldots, g_k(\,\cdot\,;\theta)\}$. The parameters θ determine the basis, and we share them across all policies. A coefficient vector $c \in \mathbb{R}^k$ selects the feedback policy

$$\pi_c(x) = \sum_{j=1}^{k} c_j g_j(x;\theta) = G_\theta(x)c, \tag{2}$$

where $G_\theta(x) = [g_1(x;\theta), \ldots, g_k(x;\theta)] \in \mathbb{R}^{n_u \times k}$. Each basis function is vector-valued and describes a state-dependent, coordinated change across the control inputs.

Since the offline datasets encode useful policies, we train $\mathcal{V}$ such that projecting π_i^{E} onto the span has minimal reconstruction error. Following Ingebrand et al. (2025a), we train a set of neural basis

functions via the empirical learning problem

$$\min_{\theta,\{c_i\}_{i=1}^N} \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{D}_i|}\sum_{(x,u)\in\mathcal{D}_i}\|u - G_\theta(x)c_i\|_2^2 + \lambda\,\mathcal{R}(\theta,\{c_i\}_{i=1}^N), \tag{3}$$

where $\mathcal{R}$ is a regularizer. The shared parameters determine the basis of the policy space, and c_i gives the coordinates of the policy observed in $\mathcal{D}_i$.

For a fixed basis, we obtain the coefficient associated with $\mathcal{D}_i$ by projecting its observed state–action pairs onto $\mathcal{V}$. With quadratic coefficient regularization, this projection is the least-squares problem

$$c_i(\theta) \in \arg\min_{c\in\mathbb{R}^k} \frac{1}{|\mathcal{D}_i|}\sum_{(x,u)\in\mathcal{D}_i}\|u - G_\theta(x)c\|_2^2 + \lambda_c\|c\|_2^2. \tag{4}$$

We train the basis to reduce the reconstruction error obtained after this projection across all offline policy instances. Function encoders implement this idea by repeatedly inferring coefficients from the current basis and updating the neural basis functions through the resulting reconstruction loss. We may also optimize the same empirical objective jointly over θ and offline coefficients. After offline training, we freeze θ. Acquiring a new policy then requires selecting only k coordinates.

Policy Identification. During the online phase, we seek the optimal policy for task τ. Suppose we receive a dataset $\mathcal{D}_\tau = \{(x_{\tau,m}, u_{\tau,m})\}_{m=1}^{M_\tau}$ collected from an expert policy π_τ^{E}. Then we simply project the expert policy π_τ^{E} onto the learned basis using least squares. This yields an approximation of π_τ^{E} that we can use for control. We evaluate identification in 3 stages: (i) whether sparse context recovers the policy coefficients, (ii) whether those coefficients reproduce expert actions, and (iii) whether the identified policy remains stable and tracks the desired behavior.

Policy Synthesis. If instead we know the new task τ but receive no expert demonstration, we can perform policy search directly in the coefficient space. We restrict the search to the learned subspace because the basis already contains useful behaviors in our environment. Substituting (2) into (1) yields the lower-dimensional optimization problem

$$\begin{aligned}\arg\min_{c\in\mathbb{R}^k} \quad & \mathbb{E}_{x_0\sim\rho_{0,\tau}}\left[\Phi_\tau(x(T_\tau)) + \int_0^{T_\tau}\ell_\tau(x(t),u(t))\,dt\right] \\ \text{s.t.} \quad & \dot{x}(t) = F(x(t),u(t)), \qquad u(t) = \sum_{j=1}^{k} c_j g_j(x(t);\theta).\end{aligned} \tag{5}$$

Policy synthesis therefore optimizes the objective over the coefficient vector c while keeping the learned basis G_θ fixed. Because $\pi_c(x) = G_\theta(x)c$ is linear in c, its coefficient Jacobian is simply $D_c\pi_c(x) = G_\theta(x)$. For differentiable dynamics and costs, we compute $\nabla_c J_{F,\tau}(\pi_c)$ by automatic differentiation through the closed-loop rollout and update only c. For nondifferentiable simulators or physical systems, we optimize c with the cross-entropy method (CEM), Bayesian optimization (BO), or the covariance matrix adaptation evolution strategy (CMA-ES).

Policy geometry and uncertainty. The ridge solve used for policy identification gives a local coefficient covariance $P_{\mathcal{C}}$ under an action-noise model. We map any coefficient covariance P to the action covariance $G_\theta(x)PG_\theta(x')^\top$. This expression is exact at fixed states because the policy is linear in c. We propagate the same covariance through the closed-loop dynamics using coefficient sensitivities to obtain first-order control and trajectory covariances. These quantities identify policy directions that the demonstrations constrain weakly and states or actuators that vary together. Appendix A.2 derives the kernels and states the limits of this local uncertainty interpretation.

4 Empirical Results

We evaluate the learned policy space across (i) humanoid, (ii) quadruped, (iii) planar-quadrotor, and (iv) linear-quadratic control problems. The humanoid experiments use the Unitree G1 in MuJoCo Playground (Zakka et al., 2025). We consider 2 complementary sources of expert behavior: (i) a publicly released Brax PPO checkpoint for command-conditioned locomotion (McGartoll, 2025); and (ii) the ProtoMotions motion-tracking framework (Tessler et al., 2025) with the BONES-SEED

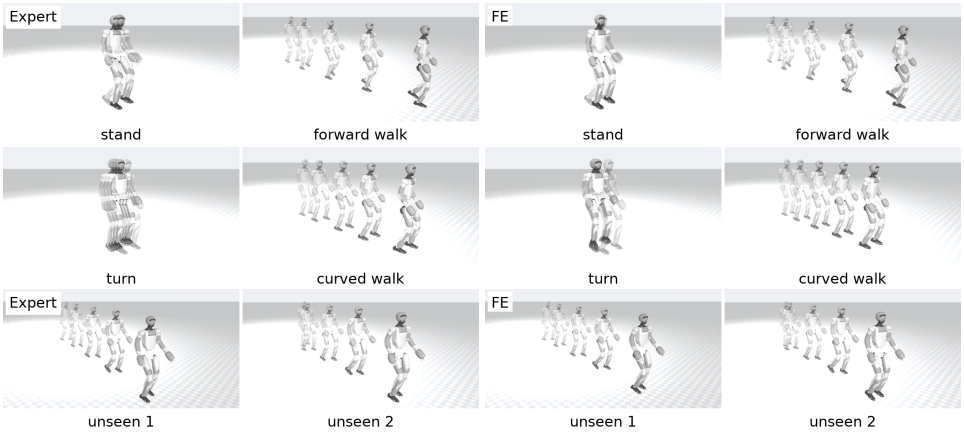


Figure 2: Expert policies (left) and corresponding identified FE policies (right). The upper panels show 4 commands represented during offline training, and the lower panels show 2 simultaneous longitudinal–lateral–turning commands absent from the finite offline task set. Each panel overlays 5 evenly spaced poses from a 300-step rollout. For the unseen commands, we infer coefficients from 64 context pairs and hold them fixed throughout the rollout.

deployment tracker (Bones Studio, 2025). The first setting tests whether a low-dimensional policy space can recover a known, smooth family, while the second asks whether the same representation can summarize richer variation normally specified by a high-dimensional motion reference. For quadruped locomotion, we use the 12-actuator Unitree Go1 in MuJoCo Playground (Zakka et al., 2025). We train PPO experts (Schulman et al., 2017) that span standing, crouching, walking, turning, curved walking, crouched walking, and leaning. We consider both a single task-conditioned teacher and separately trained experts. Finally, we use a planar quadrotor as a controlled nonlinear system with analytical expert feedback laws, and an infinite-horizon discrete LQR problem as a setting in which the unrestricted optimal controller is known exactly from the algebraic Riccati equation.

Across systems, we collect expert state–action trajectories and train the basis according to (3), without providing the task descriptor or reference variable that generated each behavior. During episodic training, we infer the coefficient associated with a context set using (4), then update the shared basis using prediction error on a disjoint query dataset. After training, we freeze the basis in (2).

We compare FE with baselines selected for each setting: a learned-feature linear-head baseline adapted by ridge regression, first-order model-agnostic meta-learning (FOMAML) (Finn et al., 2017), behavior cloning, and PULSE-style latent policies (Luo et al., 2024) for sparse identification; retrieval, matched random spaces, and parameter-space subspaces (Gaya et al., 2022) for synthesis; and functional PCA and stored-policy mixtures when complete policies are available. No baseline supports every operation we evaluate. Appendix B provides complete experimental details.

4.1 Sparse policy identification

We test whether we can identify a feedback policy from a short state–action demonstration. On the command-conditioned G1 family, we fit an 8-dimensional policy basis from 48 fixed joystick commands spanning standing, forward and backward walking, lateral walking, turning, and curved walking. After training, we freeze the basis and infer only the coefficient vector c from 64 state–action pairs using (4). We evaluate action prediction on a disjoint expert trajectory and execute the identified policy from a fresh reset without refitting. Table 1 reports the full comparison. The FE has low action and tracking error, and the identified motions qualitatively match the experts, as shown in Figure 2, which shows that the FE distills the expert policies into the shared basis. For an 8-basis G1 model, identifying c from 128 context pairs takes 0.675 ms on CPU and 0.714 ms on GPU, enabling online updates at more than 1.4 kHz after context collection. We also evaluate 2 behaviors that were unseen during training and combine longitudinal, lateral, and yaw motion. Their action RMSE remains low, and every rollout completes the horizon, which indicates that the FE generalizes to similar but unseen combinations of the learned behaviors.

Table 1: Principal policy-identification comparisons.

Benchmark	Method / split	Action RMSE ↓	Closed-loop success/survival ↑	Tracking RMSE / descriptor error ↓	State–action pairs
G1 joystick	FE (ours), training commands	**0.0427**	97.6%	**0.436**	64
	FOMAML, training commands	0.0556	**99.3%**	0.477	64
	FE (ours), held-out commands	**0.0438**	100.0%	**0.471**	64
	FOMAML, held-out commands	0.0567	100.0%	0.513	64
G1 ProtoMotions	FE (ours)	**0.170**	97.71%	**0.213**	64
	Static PULSE	0.277	**99.14%**	0.253	64
	Dynamic PULSE prior†	0.257	84.71%	0.288	—
	Dynamic PULSE posterior†	0.130	87.71%	0.282	—
Go1	FE (ours)	**0.0521**	**97.8%**	**0.133**	32
	Learned-feature ridge	0.1398	88.1%	0.442	32
	FOMAML	0.0621	87.8%	0.606	32
Planar quadrotor	FE (ours)	**0.000128**	**65.1%**	—	128
	Learned-feature ridge	0.001305	61.3%	—	128
	FOMAML	0.00946	63.3%	—	128
	Random 8-D subspace	0.02327	32.6%	—	128

Training-command rows use fresh trajectories from commands included offline; held-out commands were excluded offline. Closed-loop reports full-horizon success where available and mean survival otherwise. †The dynamic prior receives no expert context, while the posterior receives the desired reference continuously, so a fixed pair count does not apply. Bold marks the better value only within information-matched comparisons.

We compare this sparse identification procedure against FOMAML. On Go1 and the planar quadrotor, we additionally compare against a learned-feature linear head adapted by ridge regression. Table 1 shows that FE more faithfully represents the expert behavior than FOMAML on both G1 splits, while FOMAML has a small survival advantage on the G1 training commands; both methods succeed on the held-out commands. On Go1, FE attains lower action and tracking error than both baselines and improves mean survival from 88.1% for the ridge head to 97.8%; it also adapts $4.4\times$ faster than FOMAML. On the planar quadrotor, FE attains lower held-out action error and higher closed-loop success than both FOMAML and the ridge head.

We then compare FE with nonlinear latent policies on the G1 motion-tracking family, where 7 qualitatively distinct whole-body behaviors replace the low-dimensional joystick commands above. For the comparison, we train a static PULSE-style latent policy (Luo et al., 2024) on the same expert trajectories and provide both methods with the same 64 state–action pairs at test time. This baseline predicts actions with the latent-conditioned state-feedback policy $D_\phi(x, z)$. A context encoder initializes the 8-dimensional latent z, which we refine by gradient descent and hold fixed throughout the rollout. Relative to static PULSE, the FE represents the expert policies more faithfully, with lower behavior-descriptor error and teacher-action RMSE. Static PULSE has a marginally higher survival rate, but this difference is not statistically significant after correction. Its inferred latent has norm approximately 1.3×10^{-4} at test time, indicating that the decoder relies primarily on state feedback rather than the inferred behavior code. This collapse likely reflects the limited diversity of the matched 7-behavior dataset, which allows the decoder to learn a pooled state–action map without using the latent to distinguish behaviors. We also evaluate a dynamic PULSE-style controller that matches the deployment structure of the original method. Its prior receives no expert demonstration, while its posterior receives the motion reference continuously during execution. Providing this reference reduces teacher-action RMSE, but the posterior remains less successful in closed loop than the FE and produces substantially higher action jerk.

To test whether the FE merely reconstructs an existing task-conditioning scheme, we remove known structure from the expert policy family. We first use a 3-dimensional command to index the G1 joystick policies. We then replace that command with a high-dimensional motion reference in the ProtoMotions experiment. Finally, we train the Go1 experts independently, removing shared policy parameters. Across all 3 experiments, the FE identifies useful closed-loop policies. The learned basis therefore does more than recover a hidden command or reuse shared teacher parameters.

To determine whether the FE benefits from reducing adaptation to a few variables, we replace the learned basis with 8 random policy directions around the same pooled behavior-cloned policy and infer their coefficients from the same context. On the quadrotor, this random subspace has substantially higher action RMSE and lower closed-loop success than the FE. The FE learns directions that capture useful policy variation rather than benefiting from low-dimensional adaptation alone.

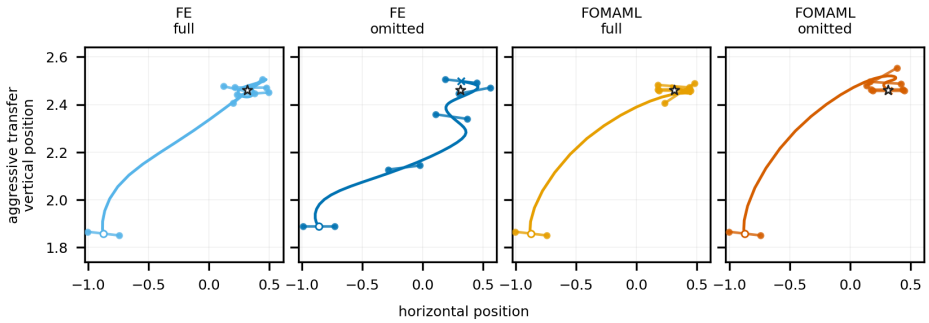


Figure 3: Representative aggressive-transfer trajectories after objective-only adaptation. Stars mark target positions; circles and crosses mark initial and final positions. Full and omitted indicate whether aggressive transfer was present during offline training.

Finally, we replicate the pendulum and HalfCheetah experiments of Gaya et al. (2022) and find lower sparse-demonstration identification error with a 4-basis FE, but mixed results in a controlled 2-point comparison: the function-space and parameter-space coordinates perform similarly overall, while the parameter-space method performs better on some reward-only perturbations. Function-space coordinates are therefore not uniformly better for policy search, but they provide comparable adaptation performance while also supporting identification, interpolation, and the other policy-space operations we evaluate.

4.2 Objective-driven policy synthesis

We next consider the case where no target demonstration is available and select the coefficient vector directly from the control objective. We optimize only c and keep the learned basis fixed.

We first demonstrate synthesis on an infinite-horizon discrete LQR family, where the unrestricted optimal controller is given by the algebraic Riccati equation. For 12 held-out objectives across 3 seeds, optimizing only the 8 coefficients without an expert demonstration attains a mean cost ratio of 1.000433 ± 0.000254 relative to exact LQR. Projecting the unavailable optimal controller into the learned span gives 1.000572 ± 0.000373. This ideal case shows that FE captures the optimal-controller family and synthesizes near-optimal policies for new objectives with negligible loss from restricting the search to learned coordinates.

We next use the planar quadrotor to separate adaptation efficiency from policy-space coverage. We consider hover, cruise, aggressive-transfer, and vertical-rate objectives and compare an FE trained on all 4 families with a matched FE trained without the evaluated family. At test time, we provide no expert demonstration. We first use a generous coefficient-search budget to test what the frozen FE space can synthesize. When aggressive transfer appears in the training data, the FE produces a smooth aggressive maneuver that reaches the target. When we omit that family, the optimized FE still reaches the target but follows a less direct, high-angle trajectory, as shown in Figure 3. This degradation is consistent with the remaining demonstrations containing no comparable high-angle states and with the target behavior lying partly outside the learned span. FOMAML reaches the target in both conditions by adapting the full policy network. We also compare FE and FOMAML under a fixed online budget of 500 gradient updates in Table 13. FOMAML succeeds more often; FE remains competitive on all but vertical-rate tracking.

We also test coefficient search on the nonlinear Go1 system using a crouched-turn target that is absent from training but combines 3 behaviors represented separately in the training set: forward motion, turning, and reduced body height. Searching the frozen 8-basis FE produces the postures and curved ground tracks shown in Figure 4. We compare CEM, BO, and CMA-ES against a task-conditioned PPO reference. CMA-ES performs best, completing every rollout with linear-velocity and yaw-rate RMSE of approximately $0.11\,\mathrm{m\,s^{-1}}$ and $0.10\,\mathrm{rad\,s^{-1}}$, respectively. The closest stored FE policy remains stable but does not reproduce the combined target. This means that we can synthesize the unseen target behavior by searching the learned policy space. Interestingly, we find that coefficient arithmetic sometimes produces recognizable behaviors. On Go1, for exam-

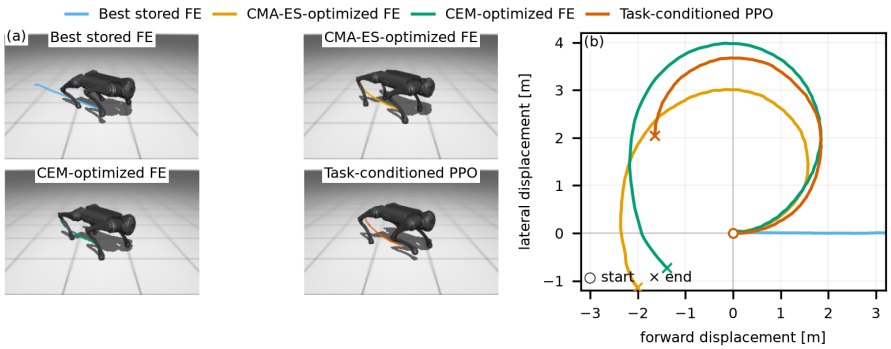


Figure 4: Go1 crouched-turn synthesis. (a) Rollout postures for the best stored FE policy, FE policies synthesized with CMA-ES and CEM, and a task-conditioned PPO reference. (b) Complete center-of-mass trajectories from above after aligning the initial body heading.

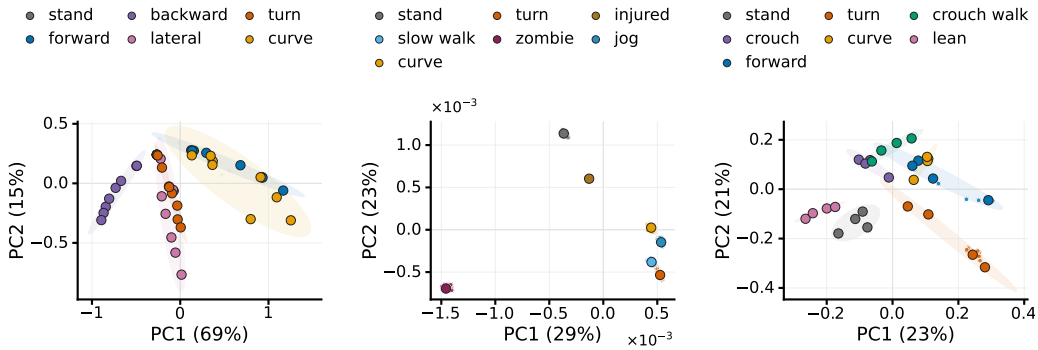


Figure 5: Learned policy coefficients projected onto their first 2 principal components. From left to right: metric-whitened command-conditioned G1 policies, ProtoMotions G1 behaviors, and independently trained Go1 policies.

ple, $c_{\text{walk}} - c_{\text{stand}} + c_{\text{crouch}}$ can produce a stable crouched walk. However, this behavior is not an intrinsic property of the learned space and we do not enforce vector arithmetic during training.

4.3 Local structure of the learned policy space

We next examine whether the learned coefficients have meaningful structure beyond serving as adaptation variables. Raw Euclidean distances between coefficients are not invariant to a change of basis, so we instead measure coefficient variation through the induced change in policy output. On the command-conditioned G1 family, the global occupancy-weighted policy metric shows that variation is distributed nonuniformly across the representation: although it has 8 coefficients, its participation rank is 5.58 ± 0.54, which measures the effective number of directions carrying variation. PCA of the metric-whitened policy coefficients shows that its leading component explains $68.6 \pm 0.3\%$ of the policy variation. Movement along this direction is almost perfectly aligned with changes in commanded forward velocity ($|r| = 0.9812 \pm 0.0002$), while the first 4 directions together explain $95.25 \pm 0.18\%$ of the variation. Thus, the dominant directions in the learned space correspond to changes in locomotion rather than arbitrary coefficient axes. Figure 5 compares this coefficient-space organization across the command-conditioned G1, ProtoMotions G1, and independently trained Go1 policy families.

A practical use of this structure is to interpolate between learned policies without additional demonstrations or optimization. On command-conditioned G1, interpolation between nearby behaviors such as forward walking, turning, lateral walking, and curved walking produces stable intermediate motions. On the ProtoMotions dataset, interpolation degrades clearly between separated behavior pairs. Among the 2 paths evaluated quantitatively, slow-walk–zombie-walk interpolation succeeds

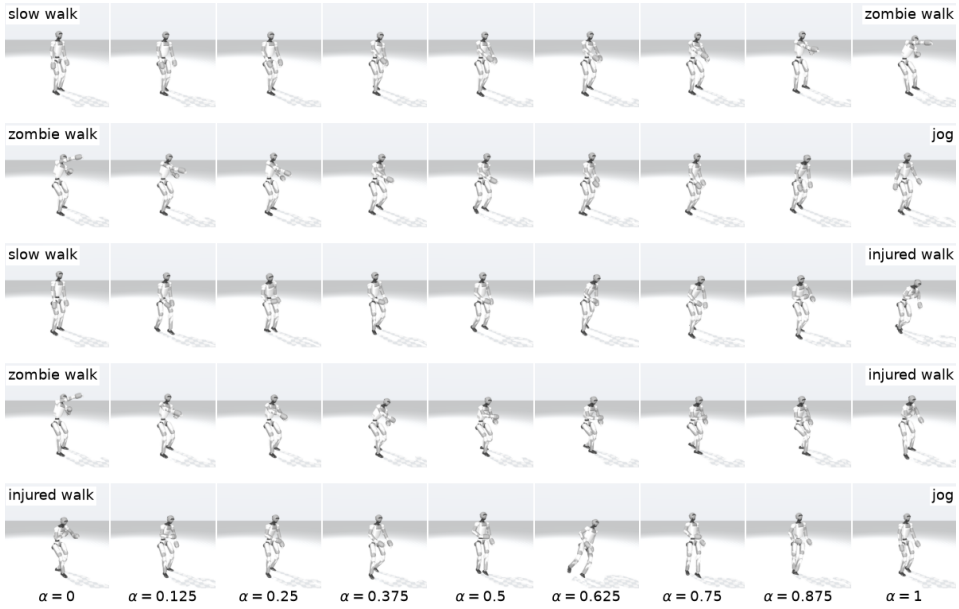


Figure 6: ProtoMotions coefficient interpolations from top to bottom: slow walking to zombie walking, zombie walking to jogging, slow walking to injured walking, zombie walking to injured walking, and injured walking to jogging. The first 2 paths are evaluated quantitatively; the remaining rows provide qualitative examples.

on 96.4% of interior-policy trials, whereas zombie-walk–jog succeeds on only 39.3% and its interior policies deviate from the intended behavioral interpolation. This result indicates that straight paths between distant coefficients can pass through regions of the learned span that the training data represent poorly; a linear policy space therefore does not guarantee stable semantic interpolation.

One advantage of the linear policy representation is that we can analyze its effective dimension and action covariance. We stack the policy Jacobians over sampled states and compute the effective rank of their singular spectrum, which estimates how many basis directions materially change the policy on a given dataset. The same Jacobians map coefficient covariance into action covariance, whose off-diagonal terms reveal actuator coupling, and propagate uncertainty in the inferred coefficients through the closed-loop dynamics. Monte Carlo coefficient samples closely match the predicted terminal uncertainty, and nominal 95% trajectory ellipses attain $94.70 \pm 0.75\%$ empirical coverage across 36 seed–task evaluations. The fixed-state action covariance is exact under the linear coefficient model, while the closed-loop trajectory covariance remains a first-order approximation.

5 Conclusion

In this paper, we showed that function encoders learn low-dimensional spaces of feedback policies directly from expert demonstrations. Across G1 humanoid locomotion, Go1 quadruped locomotion, planar-quadrotor control, and linear-quadratic control, the learned representation for each system supports sparse policy identification, objective-driven synthesis, interpolation between related behaviors, and closed-loop analysis, with identification fast enough for real-time use. Compared with the baselines, FE generally recovers expert behavior more faithfully from limited demonstrations and synthesizes new policies by optimizing only their coefficients; the same policy space also exposes effective rank, conditioning, and closed-loop uncertainty. Our experiments also show that a single linear span provides only a local approximation to the manifold of useful policies, and performance degrades when the desired behavior lies outside the coverage of the demonstrations. Future work will develop multiple local spaces or nonlinear atlases, expand the learned basis without disrupting existing policies, and test identification, synthesis, interpolation, and uncertainty analysis on physical robots under sensor noise, actuator mismatch, and real-world distribution shift.

REPRODUCIBILITY STATEMENT

The appendix documents the system definitions, datasets, model architectures, training procedures, evaluation protocols, and computational resources used in our experiments. We will release the code and trained models with the public release of the paper.

REFERENCES

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning, 2018. URL https://arxiv.org/abs/1606.05312.

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

Bones Studio. BONES-SEED: Skeletal Everyday Embodied Dataset. https://bones.studio/datasets/seed, 2025. Motion Data by Bones Studio.

Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *ICLR*, 2018.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1–2): 5–23, 2016. doi: 10.1007/s10472-015-9463-9.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning, 2017. URL https://arxiv.org/abs/1703.07326.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017.

Jean-Baptiste Gaya, Laure Soulier, and Ludovic Denoyer. Learning a subspace of policies for online adaptation in reinforcement learning, 2022. URL https://arxiv.org/abs/2110.05169.

Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu. Building a subspace of policies for scalable continual learning, 2023. URL https://arxiv.org/abs/2211.10445.

James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. *arXiv preprint arXiv:1807.08912*, 2018.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb.

A.J. Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 02 2013. doi: 10.1162/NECO_a_00393.

Tyler Ingebrand, Adam J. Thorpe, and Ufuk Topcu. Function encoders: A principled approach to transfer learning in hilbert spaces. *ICML*, 2025a.

Tyler Ingebrand, Amy Zhang, and Ufuk Topcu. Zero-shot reinforcement learning via function encoders, 2025b. URL https://arxiv.org/abs/2401.17173.

Hassan Iqbal, Xingjian Li, Tyler Ingebrand, Adam Thorpe, Krishna Kumar, Ufuk Topcu, and Ján Drgoňa. Zero-shot function encoder-based differentiable predictive control, 2026.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84 (1–2):171–203, 2011. doi: 10.1007/s10994-010-5223-6.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *IJCAI*, 2022.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control, 2024. URL https://arxiv.org/abs/2310.04582.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning, 2018. URL https://arxiv.org/abs/1803.07055.

Shie Mannor, Reuven Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. *Proceedings, Twentieth International Conference on Machine Learning*, 2, 07 2003.

Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process global optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 270–277. IEEE, May 2016. doi: 10.1109/icra.2016.7487144. URL http://dx.doi.org/10.1109/ICRA.2016.7487144.

Conor McGartoll. g1-rl-locomotion. Hugging Face model repository, 2025. URL https://huggingface.co/cmcgartoll/g1-rl-locomotion. Revision a8756173f3fae6f78fb37a8cf993bacd06833428.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 2018.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):1–17, 2022. ISSN 1557-7368. doi: 10.1145/3528223.3530110. URL http://dx.doi.org/10.1145/3528223.3530110.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 12685–12724, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/370645ae400005f888e418f46b594539-Paper-Conference.pdf.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019. URL https://arxiv.org/abs/1903.08254.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. URL https://arxiv.org/abs/1703.03864.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. ICML, 2015.

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

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. *ICML*, 2021.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning, 2022. URL https://arxiv.org/abs/2210.11653.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. Calm: Conditional adversarial latent models for directable virtual characters. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23, pp. 1–9. ACM, 2023. doi: 10.1145/3588432.3591541. URL http://dx.doi.org/10.1145/3588432.3591541.

Chen Tessler, Yifeng Jiang, Xue Bin Peng, Erwin Coumans, Yi Shi, Haotian Zhang, Davis Rempe, Gal Chechik, and Sanja Fidler. ProtoMotions3: An Open-source Framework for Humanoid Simulation and Control. https://github.com/NVLabs/ProtoMotions/, 2025.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models, 2025. URL https://arxiv.org/abs/2504.11054.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *IJCAI*, 2018.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *NeurIPS*, 2021.

Xu Wang et al. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024.

Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11217–11227. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wortsman21a.html.

Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 5(49):eabb2174, 2020. doi: 10.1126/scirobotics.abb2174.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground, 2025. URL https://arxiv.org/abs/2502.08844.

Ruihan Zhao, Tyler Ingebrand, Sandeep Chinchali, and Ufuk Topcu. Mos-vla: A vision-language-action model with one-shot skill adaptation, 2025.

AI USE STATEMENT

We used generative AI tools to assist with prose editing, drafting and debugging experiment and analysis code, and formatting figures and tables. The authors designed the methods and experimental protocols, ran the experiments, interpreted the results, and verified all AI-assisted text and code against the underlying implementation and saved outputs. The authors take full responsibility for the content of the paper.

A COEFFICIENT SENSITIVITIES, ADJOINTS, AND UNCERTAINTY

We assume that F, G_θ, ℓ_τ, and Φ_τ are continuously differentiable. We fix a realization of x_0 and evaluate every derivative along the closed-loop trajectory $(x_c(t), \pi_c(x_c(t)))$. We substitute (2) into the system dynamics and differentiate the closed-loop dynamics to obtain the sensitivity $S(t) := D_c x_c(t) \in \mathbb{R}^{n_x \times k}$, which obeys

$$\dot{S}(t) = (F_x + F_u D_x \pi_c) S(t) + F_u G_\theta(x_c(t)), \qquad S(0) = 0. \tag{6}$$

Here F_x and F_u are the dynamics Jacobians. We differentiate the cost to obtain

$$\nabla_c J_{F,\tau}(\pi_c) = \mathbb{E}_{x_0 \sim \rho_{0,\tau}} \Bigg[S(T_\tau)^\top \Phi_x(x_c(T_\tau)) + \int_0^{T_\tau} \{ S(t)^\top \ell_x + [D_x \pi_c \, S(t) + G_\theta(x_c(t))]^\top \ell_u \} \, dt \Bigg]. \tag{7}$$

In (7), the expectation averages the pathwise derivatives over the task-specific initial-state distribution.

The forward calculation propagates an $n_x \times k$ sensitivity matrix. We can compute the same gradient with the adjoint $p(t) \in \mathbb{R}^{n_x}$:

$$-\dot{p}(t) = (F_x + F_u D_x \pi_c)^\top p(t) + \ell_x + (D_x \pi_c)^\top \ell_u, \qquad p(T_\tau) = \Phi_x(x_c(T_\tau)). \tag{8}$$

We eliminate S from (7) to obtain

$$\nabla_c J_{F,\tau}(\pi_c) = \mathbb{E}_{x_0 \sim \rho_{0,\tau}} [\int_0^{T_\tau} G_\theta(x_c(t))^\top (\ell_u + F_u^\top p(t)) \, dt]. \tag{9}$$

The adjoint produces an action-space descent signal. The learned basis maps this signal into the k coefficient directions.

A.1 COEFFICIENT UNCERTAINTY FROM POLICY IDENTIFICATION

We use the linear coefficient model to quantify local identification uncertainty. For a context set $\mathcal{C} = \{(x_r, u_r)\}_{r=1}^{n_\mathcal{C}}$, define $A_\mathcal{C} = [G_\theta(x_1)^\top, \ldots, G_\theta(x_{n_\mathcal{C}})^\top]^\top$ and $u_\mathcal{C} = [u_1^\top, \ldots, u_{n_\mathcal{C}}^\top]^\top$. Under independent action noise with variance σ^2, we compute

$$\Gamma_\mathcal{C} = A_\mathcal{C}^\top A_\mathcal{C} + \lambda_{\mathrm{id}} I, \qquad \widehat{c}_\mathcal{C} = \Gamma_\mathcal{C}^{-1} A_\mathcal{C}^\top u_\mathcal{C}, \qquad P_\mathcal{C} = \sigma^2 \Gamma_\mathcal{C}^{-1} A_\mathcal{C}^\top A_\mathcal{C} \Gamma_\mathcal{C}^{-1}. \tag{10}$$

We interpret $P_\mathcal{C}$ as the covariance of the estimated coefficients under this noise model. As $\lambda_{\mathrm{id}} \to 0$, it reduces to $\sigma^2 (A_\mathcal{C}^\top A_\mathcal{C})^{-1}$ when the design has full column rank. We use small singular values of $A_\mathcal{C}$ to identify coefficient directions that the demonstration excites weakly. The inferred policy remains most uncertain along these directions.

A.2 CONTROL AND TRAJECTORY KERNELS

We let $P \in \mathbb{R}^{k \times k}$ denote a positive-semidefinite coefficient covariance. For policy identification, we set $P = P_\mathcal{C}$. For geometric analysis with a positive-definite coefficient metric M, we set $P = M^{-1}$. At fixed states, we define the matrix-valued policy kernel

$$K_\pi(x, x') = G_\theta(x) P G_\theta(x')^\top \in \mathbb{R}^{n_u \times n_u}. \tag{11}$$

If c has mean $\widehat{c}$ and covariance P, then $K_\pi(x, x') = \mathrm{Cov}(\pi_c(x), \pi_c(x'))$. We read its diagonal blocks as the control covariance at 1 state and its off-diagonal blocks as the coupling between control inputs at different states.

Along a closed-loop trajectory, coefficient perturbations change both the action and the visited state. We capture both effects with the total control sensitivity

$$H(t) := D_c[\pi_c(x_c(t))] = D_x \pi_c(x_c(t)) S(t) + G_\theta(x_c(t)) \in \mathbb{R}^{n_u \times k}. \tag{12}$$

We then compute the closed-loop control kernel

$$K_u(t,t') = H(t)PH(t')^\top \in \mathbb{R}^{n_u \times n_u}. \tag{13}$$

The direct term $G_\theta(x_c(t))$ measures the action change at a fixed state. The term $D_x\pi_c(x_c(t))S(t)$ measures the additional action change caused by the perturbed state. We compute the state kernel from the trajectory sensitivity:

$$K_x(t,t') = S(t)PS(t')^\top \in \mathbb{R}^{n_x \times n_x}. \tag{14}$$

We also compute the state–control cross-kernel $K_{xu}(t,t') = S(t)PH(t')^\top$. Together, these kernels propagate the same coefficient uncertainty through the policy and closed-loop dynamics.

For a small zero-mean perturbation δc with covariance P, we linearize

$$x_{c+\delta c}(t) \approx x_c(t) + S(t)\delta c, \qquad u_{c+\delta c}(t) \approx \pi_c(x_c(t)) + H(t)\delta c. \tag{15}$$

We use $K_x(t,t)$ and $K_u(t,t)$ as first-order state and control covariance tubes. The cross-time blocks $K_x(t,t')$ and $K_u(t,t')$ retain temporal correlations. Every kernel has rank at most k, which identifies the joint state or actuator directions represented locally by the learned coordinates.

The exact fixed-state policy covariance for the linear policy model is given in (11). We approximate the closed-loop control and trajectory covariances to first order because x_c depends nonlinearly on c. These kernels describe coefficient uncertainty only. Process noise, model uncertainty, and policy-space approximation error require additional terms.

B Additional Results

Here, we present additional experimental details and results for the policy function encoder and its baselines. In Section B.2, we show policy identification, coefficient geometry, interpolation, and data-scaling results for command-conditioned and motion-conditioned G1 policy families. In Section B.3, we show how sparse identification, basis capacity, and learned policy structure affect Go1 control. In Section B.4, we show identification, closed-loop synthesis, coefficient-geometry, and kernel results in a controlled planar-quadrotor setting. In Section B.5, we compare the recovered feedback law with the exact Riccati solution in a linear-quadratic system.

We present the baseline and ablation contracts in Section B.6. We compare against learned-feature ridge-head meta-learning in Section B.7. We compare the FE with a learned subspace of complete policy-network parameters in Section B.8. In Section B.9, we compare the FE with static and dynamic PULSE-style latent-policy representations.

We will release the code online with the public release of the paper.

B.1 Experimental details

We ran the experiments on a Linux workstation with 2 Intel Xeon Gold 6226R processors (32 physical cores and 64 hardware threads), 256 GB (251 GiB) of system memory, and 8 NVIDIA RTX A5000 GPUs with 24 GiB of memory per GPU. Each training or evaluation job used a single GPU; we did not use model parallelism, data parallelism, or computation distributed across GPUs. We used the multiple GPUs to run independent jobs concurrently, while each individual run used the compute available on 1 GPU.

B.2 G1 experiments

The Unitree G1 is a 29-actuator humanoid controlled at $50\,\mathrm{Hz}$. We evaluate 2 policy families on this system. The first teacher is a command-conditioned locomotion policy whose command $d = (v_x, v_y, \omega)$ specifies longitudinal velocity, lateral velocity, and yaw rate. Fixing d for an entire rollout defines 1 command-specific expert policy. The second teacher is a reference-conditioned motion tracker whose reference specifies a full-body motion and phase. Fixing 1 reference motion defines 1 behavior-specific expert policy.

Table 2 summarizes the 2 G1 training datasets before we describe each data source and its associated experiments.

Table 2: G1 training datasets.

Policy family	Tasks	Trajectories/task	Total trajectories	Valid pairs
Command-conditioned	48	12	576	172,144
Motion-conditioned	7	2,560	17,920	5,372,786

The motion-conditioned count per task comprises 2,048 ordinary and 512 recovery trajectories.

Table 3: G1 commands used for offline FE training.

Family	Training commands
Standing	$(0, 0, 0)$, repeated for 8 independent task instances
Forward	$v_x \in \{0.832, 0.247, 0.671, 0.531, 0.674, 0.228, 0.236, 0.324\}$
Backward	$v_x \in \{-0.162, -0.534, -0.375, -0.159, -0.312, -0.584, -0.467, -0.643\}$
Lateral	$v_y \in \{0.211, -0.126, 0.389, 0.275, -0.299, 0.318, -0.293, 0.163\}$
Turning	$\omega \in \{0.614, 0.713, 0.394, 0.224, -0.577, -0.640, 0.559, 0.563\}$
Curved	$(v_x, \omega) \in \{(0.340, -0.161), (0.451, -0.688), (0.771, -0.625), (0.782, 0.439),$ $(0.333, -0.206), (0.349, 0.178), (0.219, -0.113), (0.652, -0.067)\}$

Each tuple gives the nonzero components of $d = (v_x, v_y, \omega)$ in $\mathrm{m\,s^{-1}}$, $\mathrm{m\,s^{-1}}$, and $\mathrm{rad\,s^{-1}}$, respectively.

Task-conditioned PPO teacher dataset. We initialize the command-conditioned teacher from the publicly released Brax PPO checkpoint of McGartoll (2025), trained for locomotion on the 29-actuator Unitree G1 in the MuJoCo Playground `G1JoystickFlatTerrain` environment (Zakka et al., 2025). The teacher maps its observation and the fixed planar command to 29 normalized joint actions. The FE input is the 100-dimensional physical observation obtained after removing the 3 command entries, and the regression target is the teacher action.

For command-conditioned locomotion, we define 8 task instances from each of 6 families: standing, forward walking, backward walking, lateral walking, in-place turning, and curved walking. The 8 standing instances use the same zero command; the other 40 tasks use distinct fixed commands. Table 3 gives the exact training-command support. The support contains single-axis longitudinal, lateral, and turning commands and joint longitudinal–turning commands, but no command combines lateral motion with longitudinal motion or turning. We vary initial states, observation noise, and external pushes using the environment's default reset and perturbation processes.

For every training command, we collect 12 independently seeded trajectories of at most 300 steps: 8 for training, 2 for validation, and 2 for testing. The resulting dataset contains 576 trajectories and 172,144 valid state–action pairs after masking states following termination. For evaluation at the 48 represented commands, the 2 test trajectories exchange context and query roles, and each inferred policy is executed from fresh resets. We separately collect 2 300-step demonstrations for each of 2 held-out commands that combine longitudinal, lateral, and yaw motion.

ProtoMotions dataset. Table 4 records the 7 motion references used in the second G1 experiment. Each reference defines 1 behavior-level function for episodic FE training; context and query samples may come from different rollouts but share 1 coefficient inferred from the context set.

The 14,336 ordinary trajectories use joint-position, joint-velocity, root-position, root-orientation, root-velocity, and root-angular-velocity noise magnitudes of 0.01, 0.04, 0.004, 0.01, 0.03, and 0.05, respectively. The 3,584 recovery trajectories increase these values to 0.04, 0.15, 0.015, 0.035, 0.08, and 0.15 and initialize the robot from a different randomly phased reference motion while leaving the target behavior unchanged. The teacher completes between 99.69% and 100% of the requested recovery-augmented trajectories in each behavior, and we exclude invalid frames after termination before training.

We initialize the motion-conditioned teacher from the official inference checkpoint released with the ProtoMotions motion-tracking framework (Tessler et al., 2025). We use the ProtoMotions-compatible BONES-SEED motion assets (Bones Studio, 2025). We use a reference-conditioned PPO motion tracker with adversarial motion-prior regularization. In contrast to the joystick experiment, we specify its behaviors with full reference motions rather than a 3-dimensional velocity command. For each selected motion b, fixing its reference trajectory and phase evolution defines a behavior-conditioned controller instance $\pi_b^{\mathrm{E}}(x_t, h_t) = \pi^{\mathrm{E}}(x_t, h_t, r_b(t))$, where h_t denotes the teacher's processed-action history and $r_b(t)$ denotes the motion-specific reference target. The 7

Table 4: G1 ProtoMotions reference motions.

Behavior	Frames	Duration (s)	Source motion
Idle standing	490	16.30	`idle_loop_002__A026_M.motion`
Slow walking	354	11.77	`359_walking_slow10_poses_keypoints_retargeted.npy`
Curved walking	241	8.00	`walk_arc_cw_loop_R_002__A217_M.motion`
Left turn	313	10.40	`348_turn_left02_poses_keypoints_retargeted.npy`
Zombie walking	261	8.67	`zombie_walk_ff_loop_315_003__A135.motion`
Injured-torso walking	204	6.77	`injured_torso_walk_ff_loop_270_R_002__A231.motion`
Low-energy curved jog	325	10.80	`low_energy_jog_arc_cw_loop_003__A155_M.motion`

We select these references from the ProtoMotions-compatible BONES-SEED assets. Frame counts and durations describe the source clips before randomized-phase 6-second rollout collection.

controller instances share the weights of 1 tracking network but receive separate reference clips with different joint, root, and contact trajectories.

We select 7 G1-compatible reference motions: idle standing, slow walking, curved walking, a left turn, zombie walking, injured-torso walking, and a low-energy curved jog. The first 4 provide stable locomotion anchors with changes in speed and direction, while the latter 3 introduce pronounced differences in posture, arm configuration, torso orientation, and gait style. The source clips contain between 204 and 490 motion frames and span 6.77–$16.30\,\mathrm{s}$; collection starts at randomized reference phases and uses a common 6-second rollout horizon. During offline collection, the teacher receives its reference targets, motion timing, and processed-action history, but the FE receives only the 64-dimensional proprioceptive state consisting of 29 joint positions, 29 joint velocities, local base angular velocity, and projected gravity. We use the teacher's 29-dimensional joint action as the regression target and exclude reference pose, phase, task label, future target, and teacher action history from the learner input.

The final dataset contains 17,920 trajectories, with 2,560 trajectories and approximately 7.68×10^5 state–action samples per behavior. Of these, 2,048 trajectories per behavior begin from a randomly phased state of the corresponding motion with joint and root perturbations, while 512 recovery trajectories per behavior retain the desired reference but initialize the physical robot from a randomly selected different motion with larger perturbations. Every trajectory contains at most 300 control steps at $50\,\mathrm{Hz}$, and the resulting dataset contains 5,372,786 valid state–action pairs with a valid-frame fraction of 0.9994. We partition complete trajectories into 4,297,586 training samples and 1,075,200 validation samples, preventing states from the same rollout from appearing in both partitions.

Training. For command-conditioned locomotion, we standardize states and actions using the training split and train

$$\pi_c(x) = \mathrm{clip}(\mu_u + s_u \odot G_\theta(\hat{x})c, -1, 1)\,, \qquad \hat{x} = (x - \mu_x)/s_x, \tag{16}$$

where $c \in \mathbb{R}^8$ and a 2-hidden-layer network with 256 SiLU units per layer produces $G_\theta(\hat{x}) \in \mathbb{R}^{29\times 8}$. Each optimization step samples 8 command-policy episodes. Within each episode, 256 context points from 4 trajectories determine c by ridge regression with regularization 10^{-3}, and 256 points from 4 disjoint query trajectories define the outer action-reconstruction loss. We optimize only the shared basis for 250,000 Adam updates with initial learning rate 10^{-4}, a 10% linear warmup, and cosine decay. We train 3 seeds and evaluate the saved checkpoints on the fixed validation and test trajectory splits.

We fit an 8-basis policy FE, $u(x) = G_\theta(x)c_b$, where a shared 2-hidden-layer network with 256 SiLU units per layer produces $G_\theta(x) \in \mathbb{R}^{29\times 8}$ and $c_b \in \mathbb{R}^8$ identifies 1 of the 7 motion-defined behaviors. For each behavior and optimization step, we sample 256 context points from 4 trajectories, infer c_b using (4) with regularization 10^{-3}, and minimize action error on 256 points drawn from 4 disjoint query trajectories. We optimize only the shared basis through these differentiable least-squares solves for 250,000 Adam steps with initial learning rate 4×10^{-3}, a 10% warmup, and cosine decay to 2×10^{-4}. We evaluate both G1 policy families through identification and interpolation. We additionally vary command-policy capacity and motion-dataset coverage, analyze the learned command-policy geometry and kernel, and test zero-demonstration synthesis for the motion-conditioned family.

Identification. We test policy identification from expert state–action demonstrations at commands represented during offline training, a common robotics setting in which an observed trajectory supplies examples of the feedback behavior to be recovered. We provide a deterministic, temporally sparse context of 64 valid samples from each evaluation trajectory rather than drawing random or i.i.d. samples. The mean expert-trajectory completion fraction is 0.9976 over the requested 300-step horizon. We select 64 indices $t_1 < \cdots < t_{64}$ that span 1 complete 300-step trajectory, include both endpoints, and lie 4 or 5 control steps apart. Samples within this trajectory remain temporally correlated. We reserve the second, independently seeded trajectory exclusively for action prediction, so the context and query states come from different rollouts.

For task j, each normalized context state produces $G_\theta(\hat{x}_{t_\ell}) \in \mathbb{R}^{29\times 8}$ and each expert action produces the standardized target $\hat{u}^{\mathrm{E}}_{t_\ell} = (u^{\mathrm{E}}_{t_\ell} - \mu_u)/s_u \in \mathbb{R}^{29}$. We concatenate the 64 basis evaluations along their action dimension to form the design matrix $A_j \in \mathbb{R}^{1856\times 8}$, and concatenate the corresponding transformed expert actions to form $b_j \in \mathbb{R}^{1856}$. We then estimate the 8-dimensional task coefficient using (4) with ridge parameter 10^{-3}. Each solve therefore uses 1,856 scalar action equations to identify 8 unknowns. For each trained checkpoint, we reverse the roles of the 2 test trajectories to obtain 96 cross-trajectory identification trials, or 288 trials across 3 training seeds, and then execute every inferred policy from a third, independently seeded reset. Across seeds, the FE attains a held-out-trajectory action RMSE of 0.0427 ± 0.0006, a mean closed-loop survival fraction of 0.9916 ± 0.0003, and a $(97.6 \pm 0.6)\%$ full-horizon success rate over 300 control steps.

The inferred coefficient supplies the task identity at execution time; the FE receives the current physical state but not the command descriptor. The identified policies reproduce the demonstrated behavior from fresh resets, although turning is attenuated and lateral policies exhibit some longitudinal drift. We therefore report action reconstruction together with closed-loop survival and command tracking because pointwise action error does not fully characterize rollout stability.

Held-out commands. We next test policy identification for 2 commands excluded from offline training, $d_1 = (0.740, -0.134, -0.104)$ and $d_2 = (0.675, -0.013, -0.310)$. Both commands lie within the marginal range of every training-command coordinate, but neither type of simultaneous longitudinal, lateral, and yaw motion occurs in the finite training set. For each command, we collect 2 independent 300-step expert trajectories and perform identification in both directions: 1 trajectory supplies 64 uniformly spaced context samples for coefficient estimation, while the other supplies 300 held-out samples for action evaluation, after which their roles are reversed. This protocol produces 4 identified coefficients per checkpoint, or 12 across 3 training seeds. The held-out-trajectory action RMSE is 0.0438 ± 0.0008 across seeds. Each identified coefficient is then executed once from a separate fresh reset for at most 300 control steps. All 12 closed-loop rollouts complete the requested horizon, yielding a mean survival fraction of 100%. The corresponding planar-velocity RMSE is $0.230 \pm 0.007\,\mathrm{m\,s^{-1}}$, and the yaw-rate RMSE is $0.410 \pm 0.017\,\mathrm{rad\,s^{-1}}$. The policies are therefore stable on these evaluation resets, while their nonzero tracking errors show that successful execution does not imply exact command reproduction.

ProtoMotions behaviors. On trajectories held out by complete rollout, the trained FE attains action RMSE 0.0205 relative to the corresponding reference-conditioned teacher behaviors. We evaluate the 7 frozen endpoint policies for $20\,\mathrm{s}$ from 4 reset scenarios each; they complete 27 of 28 trials and attain mean survival fraction 0.972. Using context and query points from disjoint trajectories, physical-action RMSE decreases from 0.0349 with 4 context pairs to 0.0194 with 128 pairs and 0.0193 with 256 pairs. For reference, the motion-tracking teacher completes all 98 trials in a separate $4\,\mathrm{s}$ audit spanning every source-pose–target-motion pair at 2 initial phases, although the different evaluation horizon precludes interpreting these percentages as a matched head-to-head comparison.

Interpolation. We next test whether interpolation between learned coefficients produces intermediate policies without additional demonstrations or online optimization. For each endpoint pair A, B, we execute $c(\alpha) = (1-\alpha)c_A + \alpha c_B$ at $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. We examine 4 paths: forward-to-turn, forward-to-lateral, turn-to-curved, and backward-to-curved locomotion. At each interpolation weight, the original command-conditioned expert receives the analogous interpolated command $d(\alpha) = (1-\alpha)d_A + \alpha d_B$ and serves only as a behavioral reference; its trajectory is not used to estimate the FE coefficient. The 3 interior weights on each path are absent from the finite training set. We evaluate each interior policy from 8 independently seeded resets, producing $4 \times 3 \times 8 = 96$ FE rollouts and 96 matched expert rollouts, each with a maximum duration of 300

control steps. We report both full-horizon success, defined as the fraction of rollouts completing all 300 steps, and mean survival fraction, defined as the average proportion of the requested horizon completed before termination. The interpolated FE policies complete all 300 steps in 95 of 96 interior rollouts (99.0%) and attain a mean survival fraction of 0.9988. The matched expert policies also complete 95 of 96 interior rollouts, with mean survival fraction 0.9998. Full-horizon FE success is 95.8% for forward-to-turn and 100% for forward-to-lateral, turn-to-curved, and backward-to-curved interpolation. Figure 7(a) shows the success rate at each interpolation weight. Each marker summarizes 8 FE rollouts. We include the 2 stored endpoint coefficients at $\alpha = 0$ and $\alpha = 1$ to show the complete path, although only the 3 interior weights test unseen interpolations. Dashed curves show the matched expert policies. We see full FE success at 11 of the 12 interior path–weight combinations. The remaining point is 87.5% at $\alpha = 0.5$ on the forward-to-turn path. All 24 backward-to-curved interior rollouts complete the horizon.

We separately measure how closely each interpolated FE policy matches the behavior of the expert under the analogous interpolated command. For each matched reset, we compute the Euclidean difference between the rollout-mean behavior vectors $(\bar{v}_x, \bar{v}_y, \bar{\omega})$ and then average the 8 differences. Figure 7(b) reports this quantity. We see a mean paired difference below 0.10 at 10 of the 12 interior path–weight combinations. The largest value is 0.116 at $\alpha = 0.25$ on the forward-to-turn path. The backward-to-curved path has a difference of 0.092 at $\alpha = 0.25$ while all 8 rollouts complete the horizon. These 2 measurements therefore describe distinct properties: success records whether the robot remains active for the full horizon, while the paired behavior difference records how closely its mean velocity and yaw rate match the expert. Because this norm combines linear velocity and yaw rate in their native units, we use it only as a descriptive comparison within this figure.

ProtoMotions behaviors. We next examine whether straight lines in the learned coefficient space induce meaningful paths between the more distinct motion-defined behaviors. For the slow-walk–zombie-walk and zombie-walk–low-energy-jog pairs, we evaluate 9 policies $c(\alpha) = (1 - \alpha)c_A + \alpha c_B$, with $\alpha \in \{0, 0.125, \ldots, 1\}$, for 20 s from both endpoint pose distributions and 4 reset scenarios per pose distribution. Across the 7 interior weights, the slow-walk–zombie-walk path completes 54 of 56 trials (96.4%). Including the 2 endpoint weights gives 69 of 72 completed trials (95.8%), and every policy through $\alpha = 0.75$ completes all 8 trials. The realized behavior descriptor changes from being closer to slow walking at $\alpha = 0$ to being closer to zombie walking at $\alpha = 1$. In contrast, the zombie-walk–jog path completes 22 of 56 interior trials (39.3%), or 32 of 72 trials (44.4%) when we include the 2 endpoints, and its interior policies are simultaneously less stable and farther from both endpoint descriptors and their linearly interpolated target. These results indicate that the FE learns locally connected regions of useful feedback policies, but that its raw coefficient space is not globally convex in closed-loop behavior. Finding paths that follow the policy-induced metric or kernel, rather than Euclidean line segments in coefficient coordinates, is left for future work. We show representative rollout poses for both paths in Figure 6 of the main text. Figure 7(c) reports the corresponding 20 s success rates, and Figure 7(d) reports normalized error from the linearly interpolated behavior descriptor.

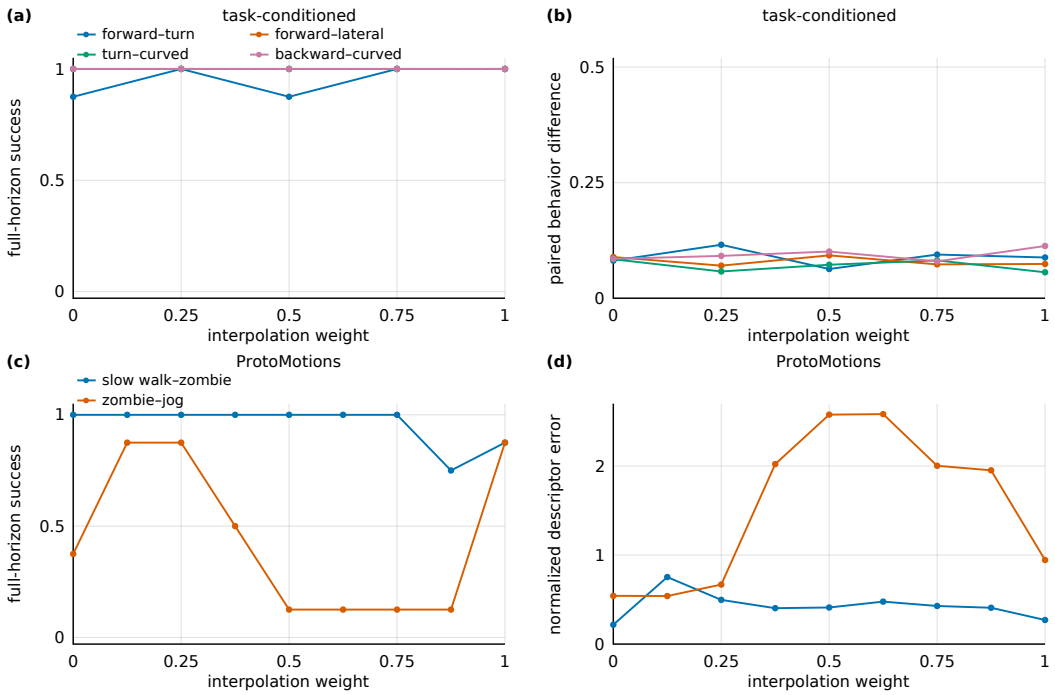


Figure 7: (a) Full-horizon success for 4 task-conditioned paths; each marker averages 8 300-step rollouts, solid curves show FE policies, and dashed curves show matched experts. (b) Mean Euclidean difference between the FE and expert rollout-mean vectors $(\bar{v}_x, \bar{v}_y, \bar{\omega})$ over the same 8 resets. (c) Full-horizon success for 2 ProtoMotions paths; each marker averages 8 20 s rollouts split across the 2 endpoint pose distributions. (d) Root-mean-square error from the linearly interpolated 7-dimensional behavior descriptor after normalizing each coordinate by its endpoint variation.

Ablations. For the task-conditioned capacity ablation, we independently train joystick-policy FEs with $k \in \{2, 4, 8, 12, 16\}$ while holding fixed the 48-command dataset, 2-layer width-256 SiLU basis, optimizer, and 250,000-update training budget. We use a separate 14-command held-out evaluation bank, rather than the 2 simultaneous commands used in the headline identification experiment. For each capacity, we infer a coefficient from 64 state–action pairs drawn from 1 of 2 trajectories for each of the 14 held-out commands, evaluate action prediction on the other trajectory, and execute the identified policy from 4 fresh resets per command. The offline action MSE decreases from 2.043×10^{-3} at $k = 2$ to 1.324×10^{-3} at $k = 16$. Mean held-out action RMSE is 0.0501, 0.0455, 0.0431, 0.0425, and 0.0430 for $k = 2$, 4, 8, 12, and 16, respectively. The improvement therefore begins to plateau at $k = 8$. All 5 models complete all 56 fresh-reset rollouts, so survival does not distinguish capacity. Tracking remains mildly capacity dependent: $k = 8$ has the lowest yaw-rate RMSE ($0.369\,\mathrm{rad\,s^{-1}}$), while the differences among $k = 8$, 12, and 16 are small. We retain $k = 8$ as the compact operating point because its held-out action RMSE is within 1.5% of the minimum while using fewer online coefficients. The effective-rank diagnostic below provides a complementary geometric justification. Table 5 compiles the 4 capacity criteria.

We separately ablate the ProtoMotions FE architecture under a fixed 50,000-update budget. Every model uses 8 bases, episodic context–query ridge training, and the same recovery-augmented 7-behavior dataset containing 2,560 trajectories per behavior. Table 6 reports mean and sample standard deviation over 3 training seeds. Each closed-loop entry averages the 7 endpoint policies and 4 common 20 s reset scenarios per endpoint, giving 28 rollouts per seed and 84 per architecture.

At the chosen width and depth, SiLU gives the lowest action errors and the greatest mean closed-loop success among the tested activations. Increasing width or depth improves both action reconstruction and long-horizon robustness under the fixed update budget: the width-512 and 4-layer models complete all 84 closed-loop trials. We use the 2×256 SiLU model as the main cross-system config-

Table 5: Matched task-conditioned G1 capacity sweep.

Bases k	Offline MSE ($\times 10^{-3}$)	Held-out RMSE	Survival	Yaw RMSE
2	2.043	0.0501	1.000	0.386
4	1.615	0.0455	1.000	0.375
8	1.380	0.0431	1.000	**0.369**
12	1.338	**0.0425**	1.000	0.372
16	**1.324**	0.0430	1.000	0.375
Teacher	—	—	1.000	—

We independently train each FE on the same 48 task instances for 250,000 updates. Held-out metrics average the separate 14-command capacity bank; survival averages 56 fresh-reset scenarios. Each capacity uses 1 training seed. Bold marks only the best value in each nonconstant metric; $k = 8$ is selected as the compact operating point in the text. Yaw RMSE is in $\mathrm{rad\,s^{-1}}$.

Table 6: G1 ProtoMotions architecture ablation.

Architecture	Held-out RMSE ↓	Identification RMSE ↓	Survival ↑	Success ↑
ReLU, 2×256	0.02128 ± 0.00012	0.02131 ± 0.00010	0.887 ± 0.049	0.774 ± 0.074
Tanh, 2×256	0.02244 ± 0.00018	0.02239 ± 0.00023	0.806 ± 0.049	0.679 ± 0.071
GELU, 2×256	0.02140 ± 0.00044	0.02136 ± 0.00043	0.957 ± 0.024	0.881 ± 0.021
SiLU, 2×256	0.02093 ± 0.00012	0.02099 ± 0.00019	0.965 ± 0.025	0.929 ± 0.036
SiLU, 2×128	0.02574 ± 0.00016	0.02565 ± 0.00021	0.621 ± 0.024	0.345 ± 0.041
SiLU, 2×512	$\mathbf{0.01743\pm0.00019}$	$\mathbf{0.01769\pm0.00022}$	$\mathbf{1.000\pm0.000}$	$\mathbf{1.000\pm0.000}$
SiLU, 3×256	0.01957 ± 0.00011	0.01980 ± 0.00007	0.981 ± 0.017	0.964 ± 0.036
SiLU, 4×256	0.01908 ± 0.00016	0.01946 ± 0.00032	$\mathbf{1.000\pm0.000}$	$\mathbf{1.000\pm0.000}$

Errors are physical-action RMSE. Survival is the mean fraction of the requested 20 s horizon completed; success is the fraction of rollouts completing the full horizon. All configurations use the same data, split, coefficient dimension, optimizer schedule, and evaluation resets.

uration, while this ablation shows that additional policy-basis capacity can improve the G1 endpoint policies.

We next analyze the task-conditioned coefficient geometry and kernel. A direct Euclidean projection of the raw coefficients is not invariant to the FE parameterization: an invertible change of basis can rescale or mix the columns of G_θ while leaving every represented policy unchanged. We therefore measure coefficient displacements by their induced first-order change in policy output. For the standardized linear action map used by the FE, $\pi_c(x) = \mu_u + s_u \odot G_\theta(\hat{x})c$ before physical clipping, the coefficient Jacobian is $J_c(x) = \mathrm{diag}(s_u)G_\theta(\hat{x})$ away from an active clip boundary. We define the policy-induced metric

$$M = \mathbb{E}_{(x,c)\sim\rho}\left[J_c(x)^\top J_c(x)\right], \qquad \|\delta c\|_M^2 = \delta c^\top M \delta c, \tag{17}$$

where ρ is the empirical distribution of states and corresponding task coefficients. We estimate M from 14,400 valid states obtained from 1 fresh 300-step expert rollout for each of the 48 training task instances, using the learned coefficient associated with that task at every state. The transformed coordinate $\widetilde{c} = M^{1/2}c$ has ordinary Euclidean distance $\|\delta\widetilde{c}\|_2$ equal to the local policy distance $\|\delta c\|_M$, so PCA of $\widetilde{c}$ describes variation in policy outputs rather than arbitrary variation in the coefficient gauge.

We compute this metric-aware projection for the 48 learned task coefficients and the 2 coefficients identified from unseen joint-command demonstrations. Across 3 training seeds, this global occupancy-weighted metric has participation rank 5.58 ± 0.54, which measures the effective number of contributing eigenvalue directions as $(\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$. Thus, roughly 6 combinations of the 8 coefficient directions dominate action-space variation under the sampled occupancy distribution. The first principal component explains $68.6 \pm 0.3\%$ of metric-aware policy variation and is strongly correlated with requested longitudinal velocity ($|r| = 0.9812 \pm 0.0002$); lateral velocity is most strongly associated with the third component ($|r| = 0.742 \pm 0.018$), and yaw rate with the fourth ($|r| = 0.921 \pm 0.010$), where component signs are arbitrary. The first 2 components explain $84.09 \pm 0.27\%$ of policy variation and the first 4 explain $95.25 \pm 0.18\%$. Forward walking, standing, and backward walking policies are ordered primarily along the dominant longitudinal direction, while turning and lateral behavior occupy distinct directions that become clearer in later components.

Policy-kernel diagnostics. We additionally test the rank of the policy tangent and its induced matrix-valued kernel on 12 family-balanced, 300-step teacher trajectories. Unlike the global metric above, this diagnostic forms a separate local coefficient Gram matrix for each sampled task trajec-

Table 7: G1 data-scaling ablation.

Trajectories per behavior	Held-out RMSE ↓	Identification RMSE ↓	Survival fraction ↑	Full-horizon success ↑
8	0.0370	0.0366	8.7%	0.0%
32	0.0220	0.0225	43.6%	17.9%
128	0.0165	0.0165	81.4%	71.4%
512	0.0144	0.0141	87.5%	82.1%
2,048	**0.0139**	**0.0136**	**91.5%**	**89.3%**

Each model uses 8 bases, 250,000 updates, and 1 training seed. Errors are physical-action RMSE. Closed-loop values average 7 behaviors and 4 common reset scenarios per behavior over a 20 s horizon. Arrows indicate the preferred direction.

tory. Although $J_c(x)$ has numerical rank 8 at every sampled state, the singular spectrum of the stacked tangent has effective rank 5.60 ± 0.37: approximately 6, 7, and 8 directions contain 90%, 95%, and 99% of its squared singular-value energy, respectively. These task-local coefficient metrics are positive definite with mean condition number 1.75 ± 0.13, so the empirical kernel retains algebraic rank 8, but its variation is strongly concentrated in a lower-dimensional subspace. Moreover, $54.4 \pm 2.8\%$ of the same-state kernel's squared energy lies in off-diagonal actuator entries, showing that the learned representation couples joint actions rather than reducing to independent scalar kernels. Together with the capacity ablation, this spectrum supports 8 bases as a compact operating point: roughly 6 directions capture most local policy sensitivity, while the remaining directions retain weaker variations that improve the representation of the task family. Figure 8 shows the corresponding action covariance, excitation spectra, and kernel spectra.

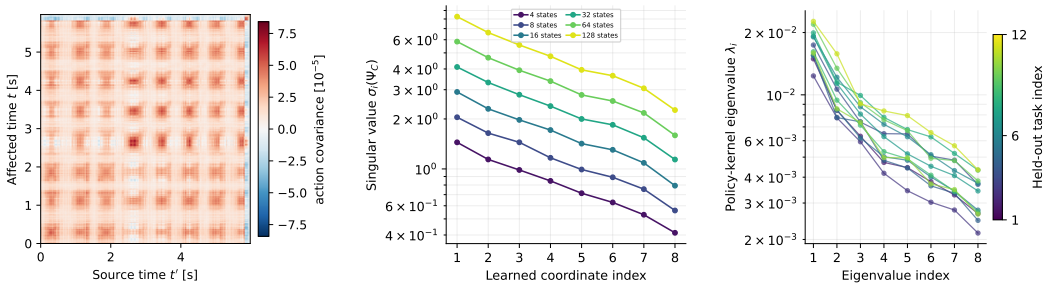


Figure 8: From left to right: same-state action covariance, coefficient-direction excitation, and induced kernel spectrum for the selected 8-basis G1 joystick FE. The middle-panel legend maps color to context size; the right-panel colorbar maps color to held-out task index.

We also evaluate ProtoMotions data scaling. To isolate the effect of offline dataset coverage, we retrain the standard 8-basis FE on nested subsets of the same-motion, perturbed-reset dataset while holding the architecture, validation split, batch size, optimizer, and 250,000-update training budget fixed; this ablation excludes the separate cross-motion recovery trajectories used by the final model. Table 7 shows that increasing offline trajectory coverage consistently improves both policy identification and closed-loop performance. Although held-out action error begins to saturate beyond 128–512 trajectories per behavior, 20 s full-horizon success continues to increase from 71.4% at 128 trajectories to 89.3% at 2,048 trajectories, demonstrating that small improvements in pointwise imitation error can correspond to substantial gains in long-horizon robustness.

Synthesis. We also evaluate zero-demonstration synthesis for a new target behavior under 1 fixed CEM rollout budget and common reset seeds. Raw coefficient arithmetic attains validation loss 70.29, compared with 1.93 for the nearest stored coefficient. CEM reduces the loss to 2.77 from a cold start, 20.31 from the raw arithmetic coefficient, and 2.91 from its metric-reduced coordinates. Thus, coefficient arithmetic is not a reliable policy-composition operator, although it can define an initialization for task-aware search.

Table 8: Go1 FE/FOMAML identification.

Method	Action RMSE ↓	Survival ↑	v_{xy} RMSE ↓	Yaw RMSE ↓	Time (s) ↓
Policy FE (ours)	$\mathbf{0.0521 \pm 0.0008}$	$\mathbf{0.978 \pm 0.028}$	$\mathbf{0.083 \pm 0.015}$	$\mathbf{0.103 \pm 0.015}$	$\mathbf{0.0677 \pm 0.0006}$
FOMAML	0.0621 ± 0.0018	0.878 ± 0.125	0.490 ± 0.074	0.355 ± 0.080	0.2961 ± 0.0629

Mean $\pm$ standard deviation over 4 matched seeds. Both methods adapt using demonstration action loss. Arrows indicate the preferred direction; bold entries mark the better mean.

B.3 GO1 LOCOMOTION

The experiments use the Go1 model, flat-terrain joystick environment, PPO implementation, and baseline optimization settings released with MuJoCo Playground (Schulman et al., 2017; Zakka et al., 2025). We do not import an external Go1 teacher checkpoint. Starting from randomly initialized weights, we train a task-conditioned teacher in 3 PPO stages with 512 parallel training environments and 32 evaluation environments per stage, for 16,384,000 simulated environment steps in total. The teacher receives the 6-dimensional descriptor $d = (v_x, v_y, \omega, h, r, p)$, which specifies longitudinal and lateral velocity, yaw rate, base height, roll, and pitch, together with the environment observation, and outputs 12 normalized joint actions. We retain standing, crouching, forward walking, in-place turning, curved walking, crouched walking, and leaning after auditing each family for survival and task tracking; backward and lateral walking remain upright but fail the prescribed velocity-tracking gates, so we exclude them.

Dataset. We construct 2 versions of this 7-family problem. In the joint-teacher setting, one task-conditioned PPO policy represents $\pi^{\mathrm{E}}(x, d)$, and fixing d produces 1 member of the policy family. In the independent-teacher setting, we train 7 separately parameterized PPO policies, one for each behavior family, with each expert covering a small range of speeds, heights, or turning rates within its family. Each dataset contains 28 policy functions, comprising 4 descriptors from each family, and multiple independently seeded and perturbed trajectories for every function. Both datasets contain 896 trajectories: 32 trajectories per policy, partitioned into 24 training, 4 validation, and 4 test trajectories.

Training. We distill each dataset into the task-blind policy family $\pi_c(x) = G_\theta(x)c$, where the FE receives only the 37-dimensional physical state and never receives d. During episodic FE training, several trajectories from 1 policy provide context for a differentiable ridge estimate of c, disjoint trajectories provide a query loss, and we back-propagate the query action error through the coefficient solve to update G_θ. The joint-teacher FE uses an 8-dimensional coefficient, a 2-hidden-layer width-256 basis, and 250,000 Adam updates with linear warmup followed by cosine decay. The independent-expert FE uses the same representation, optimizer schedule, trajectory split, and update budget to isolate the teacher construction rather than dataset size, FE architecture, or optimization budget.

Identification. After training, we test policy distillation by estimating a coefficient for each held-out policy from a limited state–action demonstration using (4). We measure action reconstruction on a disjoint expert trajectory and then hold $\hat{c}$ fixed during closed-loop execution from fresh resets. This protocol separately tests fitting the observed demonstration, predicting actions on a different expert occupancy, and controlling the robot from new initial conditions. Across 4 matched seeds, the standardized 8-basis joint-teacher FE attains held-out action RMSE 0.0521 ± 0.0008 and mean horizon survival 0.978 ± 0.028. On the trajectory-level validation split used during training, the joint-teacher and independent-expert FEs attain action RMSE 0.06979 ± 0.00011 and 0.06169 ± 0.00006, respectively. We compare sparse ridge identification with task-blind pooled behavior cloning and FOMAML; Table 15 records the offline and online information available to each method and marks comparisons that require complete stored policies rather than sampled trajectories. The FE and FOMAML receive the same expert context and both adapt by minimizing state–action regression error; neither method observes the locomotion reward during adaptation. Table 8 reports the 4-seed comparison. Relative to FOMAML, the FE reduces action RMSE, adapts $4.4\times$ faster, and more accurately reproduces the requested linear velocity and yaw rate. The FE also achieves a 10.1 percentage-point higher mean survival fraction in this evaluation.

The FE selects an 8-dimensional coefficient by least squares, whereas FOMAML adapts all parameters of 1 meta-trained network. In this matched evaluation, the FE more faithfully reproduces the demonstrated motion and remains upright more often. We also use the task-conditioned teacher, pooled behavior-cloned policy, and nearest stored expert as closed-loop references. We also evaluate a stored-policy simplex that selects convex weights over complete offline policy heads, matching the policy-mixture structure used by Gaya et al. (2022). We call this a Gaya-compatible simplex control rather than an exact reproduction—it uses stored fitted policies, whereas Gaya jointly learns full-network parameter anchors by reinforcement learning under uniformly sampled simplex coordinates and a squared-cosine anti-collapse penalty.

We further test identification by holding the number of context samples fixed and drawing them either from 1 trajectory or across several reset occupancies. Multi-trajectory context improves the condition number of the least-squares design but does not uniformly improve action error or survival, showing that numerical conditioning and closed-loop policy quality are related but distinct. In a disjoint-trajectory context sweep, mean held-out action RMSE decreases from 0.0745 with 4 state–action pairs to 0.0594 with 128 pairs, with most of the improvement obtained by 16–32 pairs. To distinguish an ill-determined coefficient estimate from a requested behavior outside the learned policy family, we also measure the normalized projection residual $r_{\mathrm{proj}} = \min_c \|u_C - G_\theta(x_C)c\|_2 / \|u_C\|_2$ on the identification context. With 8 context pairs, its task-level Spearman correlation with subsequent failure is 0.330 (95% bootstrap CI $[-0.075, 0.637]$) and its failure-detection AUROC is 0.701. Held-out action error is more predictive in this evaluation, with correlation 0.665 and AUROC 0.939, but requires a disjoint expert trajectory. The projection residual is available from the identification context alone and provides a suggestive, rather than decisive, pre-execution diagnostic. We also remove curved walking entirely from offline FE training and attempt to identify it from a new demonstration. With curved walking included during training, the identified policy attains action RMSE 0.0786, survival fraction 0.974, planar-velocity RMSE $0.124\,\mathrm{m\,s^{-1}}$, and yaw-rate RMSE $0.085\,\mathrm{rad\,s^{-1}}$. Removing the family increases action RMSE to 0.2728 and reduces survival to 0.320; planar-velocity and yaw-rate RMSE increase to $0.622\,\mathrm{m\,s^{-1}}$ and $0.484\,\mathrm{rad\,s^{-1}}$, respectively. Online least squares therefore selects a policy within the learned span; behavior families omitted from offline training remain poorly represented.

Ablations and representation analysis. We independently train standard FEs with $k \in \{2, 4, 8, 16\}$ basis functions. Offline action MSE decreases from 0.003633 at $k = 2$ to 0.003142 at $k = 16$, whereas held-out identification RMSE is 0.0625, 0.0602, 0.0596, and 0.0614 and closed-loop survival is 0.710, 0.852, 0.911, and 0.859, respectively. Reconstruction continues to improve beyond $k = 8$, while held-out identification and closed-loop control do not; we therefore use 8 bases for the primary Go1 analysis. We additionally train the basis with a Matryoshka objective that requires the prefixes $k = 2, 4, 8, 16$ to reconstruct the policy family. The ordered prefixes attain identification RMSE 0.0626, 0.0611, 0.0608, and 0.0610 and progressively higher survival 0.816, 0.860, 0.883, and 0.907, respectively. The complete prefix nearly matches the conventionally trained 8-basis FE. A pooled-policy residual FE attains identification RMSE 0.0603 and survival 0.884, compared with 0.0596 and 0.911 for the matched standard FE. These controlled, 1-seed ablations use the same independent-expert dataset, trajectory split, width-256 2-layer SiLU architecture, 250,000-update budget, context, and rollout fixture.

Table 9 consolidates the structural and training-coverage ablations under their respective evaluation scopes.

Coefficient geometry and policy kernel. We also analyze the learned representation in both coefficient and policy space. A low-dimensional PCA projection of the fitted coefficients reveals recognizable family structure and provides a common view of interpolation, coefficient addition, and online search trajectories. Since raw coefficient coordinates change under an invertible reparameterization of the basis, we also form the local policy tangent $J_c(x) = \partial \pi_c(x)/\partial c$, the occupancy metric $M = \mathbb{E}_{x\sim\rho}[J_c(x)^\top J_c(x)] + \lambda I$, and the induced matrix-valued kernel $K(x, x') = J_c(x) M^{-1} J_c(x')^\top$. For the 8-basis FE, the tangent has local numerical rank 8 and mean effective rank 5.65, while 32.3% of the same-state kernel energy lies in off-diagonal actuator entries. Tangent-subspace similarity decreases from states within one trajectory to different rollouts, tasks, and behavior families, indicating that the FE learns a state-dependent local action space whose

Table 9: Go1 policy-FE ablations.

Ablation	Variant	Active bases	Action RMSE ↓	Survival ↑
Structural ablations, averaged over all 28 policy functions				
Capacity	Standard	2	0.06248	0.7103
Capacity	Standard	4	0.06020	0.8515
Capacity	Standard	8	**0.05958**	**0.9113**
Capacity	Standard	16	0.06135	0.8592
Ordered prefix	Matryoshka	2	0.06261	0.8164
Ordered prefix	Matryoshka	4	0.06107	0.8597
Ordered prefix	Matryoshka	8	0.06082	0.8828
Ordered prefix	Matryoshka	16	0.06100	0.9074
Parameterization	Residual	8	0.06033	0.8840
Training-family coverage, evaluated on curved walking only				
Coverage	Curved walking included	8	**0.07865**	**0.9738**
Coverage	Curved walking excluded	8	0.27284	0.3197

The structural panel uses one training seed and averages over 28 held-out policy identifications, each executed from four fresh resets. The coverage panel averages the four curved-walking policies under the same context and rollout protocol. Action RMSE is measured on a trajectory disjoint from the identification context; survival is the mean completed fraction of the 1,000-step horizon.

geometry follows the organization of the demonstrated behaviors. Figure 9 shows these Go1 kernel diagnostics.

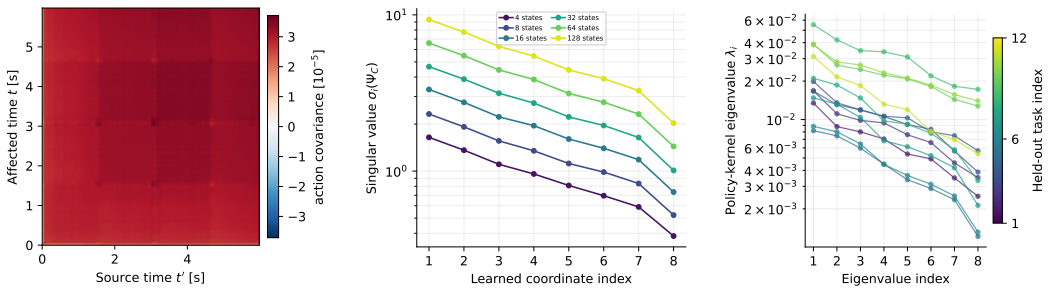


Figure 9: From left to right: same-state action covariance, coefficient-direction excitation, and induced kernel spectrum for the selected 8-basis Go1 FE. The middle-panel legend maps color to context size; the right-panel colorbar maps color to held-out task index.

Interpolation. We next evaluate whether this geometry supports interpolation without a new demonstration or online optimization. For endpoint coefficients c_A and c_B, we execute $c(\alpha) = (1-\alpha)c_A + \alpha c_B$ along forward-walk–crouched-walk and forward-walk–turn paths and compare with both the endpoint policies and a coefficient identified from an analogous target demonstration. The interpolated policies produce recognizable changes in speed, posture, and turning. Under the standardized joint-teacher FE, the posture path survives the full horizon at every interpolation point. The steering path survives 0.918 of the horizon on average, although its affine midpoint reaches only 0.605. Identifying the midpoint coefficient from 32 state–action pairs restores full-horizon survival. The corresponding held-out action RMSE is 0.0786 along the affine steering path and 0.0549 after demonstration-based identification. The steering result remains nonmonotonic between stable endpoints, indicating useful local connectivity without global convexity of the stable-policy set. Constant metric whitening cannot change an affine segment after mapping it back to coefficient space; constructing a genuinely different path requires a state-dependent geodesic optimizer and is left for future work.

Coefficient algebra. We also test coefficient analogies such as $c_{\text{walk}} - c_{\text{stand}} + c_{\text{crouch}}$. We occasionally see the resulting coefficient produce a stable behavior that reflects the intended combination. These successes are not reliable, however, because our training objective does not explicitly enforce semantic vector arithmetic; the same algebra therefore does not hold in general. Such structure would nevertheless be useful and is natural to investigate in this representation: the learned policy space is linear and supports vector addition and subtraction by construction. We view principled

objectives for making those operations behaviorally consistent as a promising direction for future work.

Synthesis. Finally, we test demonstration-free synthesis by optimizing c for a new crouched-turn objective. We evaluate the 8-basis independent-expert FE on a crouched-turn target of $(v_x, \dot{\psi}, h) = (0.45\,\mathrm{m\,s^{-1}}, 0.30\,\mathrm{rad\,s^{-1}}, 0.25\,\mathrm{m})$. We run CEM and CMA-ES from multiple ranked-bank and composition initializations and evaluate the selected candidates on 64 common disturbed 1,000-step resets. The best stored coefficient survives the full horizon in every rollout, with linear-velocity RMSE $0.0545\,\mathrm{m\,s^{-1}}$ and yaw-rate RMSE $0.2968\,\mathrm{rad\,s^{-1}}$. The selected CEM candidate attains survival fraction 0.965, 90.6% full-horizon success, linear-velocity RMSE $0.1303\,\mathrm{m\,s^{-1}}$, and yaw-rate RMSE $0.1360\,\mathrm{rad\,s^{-1}}$. The selected CMA-ES candidate completes every rollout, with linear-velocity RMSE $0.1135\,\mathrm{m\,s^{-1}}$ and yaw-rate RMSE $0.0954\,\mathrm{rad\,s^{-1}}$. Thus coefficient search can improve combined posture and turning tracking relative to retrieval, although different candidates trade linear-speed and yaw accuracy.

B.4 Planar quadrotor

We define the quadrotor state as $x = (p_x, p_z, \phi, v_x, v_z, \omega) \in \mathbb{R}^6$, where (p_x, p_z) is its planar position, ϕ is its body angle, (v_x, v_z) is its linear velocity, and ω is its angular velocity. The control input $u = (u_L, u_R) \in \mathbb{R}^2$ specifies the left and right rotor thrusts. The equations of motion are

$$\begin{aligned} \dot{p}_x &= v_x, & \dot{p}_z &= v_z, & \dot{\phi} &= \omega, \\ \dot{v}_x &= -\frac{u_L + u_R}{m}\sin\phi - d v_x, & \dot{v}_z &= \frac{u_L + u_R}{m}\cos\phi - g - d v_z, & \dot{\omega} &= \frac{\ell(u_L - u_R)}{I} - d_\omega \omega. \end{aligned} \tag{18}$$

with $m = 1$, $I = 0.026$, $\ell = 0.22$, $d = 0.08$, $d_\omega = 0.03$, and $0 \le u_L, u_R \le 8$. We vary the horizontal and vertical goals and the position, attitude, and effort weights while holding this plant fixed. A validated geometric PD controller supplies an analytical feedback reference for each task: it first computes desired horizontal and vertical accelerations, converts them to total thrust and desired attitude, and then uses attitude feedback to divide that command between the 2 rotors.

Data collection. We sample 48 training, 8 validation, and 12 held-out goal–cost combinations and collect teacher state–action trajectories from independently randomized initial position, attitude, linear velocity, and angular velocity. This dataset contains 38,400 nominal training pairs and 76,800 additional pairs providing disturbed state coverage over 200-step rollouts with $\Delta t = 0.04\,\mathrm{s}$. We fit a standardized linear policy $\widehat{u} = G_\theta(\widehat{x})c$ with $G_\theta(\widehat{x}) \in \mathbb{R}^{2\times k}$. During execution, we de-normalize $\widehat{u}$ and clip the resulting rotor thrusts to their physical bounds.

Ablations. We compare standard, pooled-policy residual, and Matryoshka training for $k \in \{2, 4, 8, 16\}$ and select the model by fresh-reset closed-loop validation rather than action MSE. This criterion is consequential: although increasing capacity and imposing nested prefixes often reduce reconstruction error, they do not monotonically improve control, and an early model with action MSE near 10^{-3} has essentially zero fresh-reset success. We use a standard 8-basis FE with 2 hidden layers of width 256 and SiLU activations, trained on disturbed coverage with the episodic ridge objective. We retain the residual and Matryoshka variants only as ablations.

Policy identification. We test whether a short expert demonstration identifies a usable feedback policy. For a held-out task, we apply (4) to context pairs $C = \{(x_i, u_i)\}_{i=1}^n$ in standardized action coordinates; we then freeze $\hat{c}$ and evaluate it on disjoint expert states and in closed loop from fresh disturbed initial conditions. With 16 context states, the identified policies incur mean cost 1.012 times the teacher cost, with median ratio 1.000, and the 32-context evaluation succeeds on 100.0% of rollouts at mean cost ratio 1.020. Because every state contributes 2 scalar action equations, identifying $k = 8$ coefficients requires at least 4 states: the measured design ranks are 2, 4, and 8 for 1, 2, and 4 context states, respectively. We therefore reach full rank with 4 context states, but the resulting system is poorly conditioned: small changes in those samples can produce large changes in the identified coefficient. We see substantially better conditioning with 8 states. At 16 states, identification reaches a practical knee, and increasing the context from 32 to 128 states provides little additional benefit.

Table 10: Unified planar-quadrotor comparison.

Method	Action RMSE ↓	Success ↑	Cost/teacher ↓
Analytical teacher	0	0.644 ± 0.022	1.000
Policy FE (ours)	$\mathbf{0.000128 \pm 0.000011}$	0.651 ± 0.010	1.171 ± 0.260
Functional PCA†	0.00757 ± 0.00050	0.585 ± 0.058	1.951 ± 0.912
Random 8-D subspace	0.02327 ± 0.00668	0.326 ± 0.064	277.430 ± 207.394
Nearest expert†	0.00850 ± 0.00171	0.541 ± 0.044	1.325 ± 0.138
Pooled BC	0.24822 ± 0.01961	0.016 ± 0.014	22.692 ± 1.614
Adapted pooled BC	0.02727 ± 0.00226	0.329 ± 0.008	3.444 ± 0.762
FOMAML	0.04929 ± 0.00600	$\mathbf{0.665 \pm 0.093}$	1.064 ± 0.272
Stored-policy simplex†	0.00543 ± 0.00289	0.655 ± 0.018	$\mathbf{1.036 \pm 0.079}$

Mean $\pm$ standard deviation over 3 seeds. Each seed uses 512 offline tasks, 32 held-out tasks, 128 context pairs from 3 trajectories, 1 disjoint query trajectory, and 2,048 common closed-loop trials per method. We normalize cost by the analytical teacher. † marks methods given complete analytical policies rather than sampled trajectories.

Matched identification baselines. Matched baselines isolate whether this result follows merely from reducing online adaptation to 8 variables. Using the same 12 held-out tasks, expert contexts, disturbances, and evaluation resets, the 8-dimensional FE obtains action RMSE 0.000643 ± 0.000073, 100.0% success, and cost ratio 1.017. An equally sized random subspace around the same pooled policy fails every rollout and does not produce a usable controller. Functional PCA is the strongest comparison, achieving 99.5% success and cost ratio 1.001 using complete evaluations of all 48 analytical training policies rather than sampled trajectories. FOMAML achieves 34.4% success. Adapting a pooled behavior-cloned policy achieves 11.2%, and nearest-expert retrieval achieves 28.9%. A Gaya-style convex mixture of the stored policies achieves 51.6%. These comparisons show that low online dimension is not sufficient: the learned functional directions, and not merely the number of fitted coefficients, account for the FE's performance.

Large-scale goal-family comparison. We next repeat the comparison under a larger, goal-only task distribution to test whether this conclusion persists with substantially broader offline coverage. We sample 512 training tasks and 32 held-out tasks by varying the horizontal and vertical goals while holding the plant, cost, and controller construction fixed. We evaluate every analytical training policy at 768 states spanning position, attitude, linear velocity, and angular velocity, yielding 393,216 offline state–action pairs. We train a standard 8-basis FE for 50,000 episodic updates; each update estimates a task coefficient from 16, 32, 64, 96, 128, or 256 randomly selected pairs and updates the basis using prediction error on the remaining states. At evaluation, we give every adaptive method the same 128 context pairs sampled across 3 perturbed expert trajectories. We measure action error on a fourth, disjoint trajectory and execute each frozen policy from 64 common disturbed resets for each held-out task.

Table 10 compares the methods. Among the representations trained from sampled trajectories, the FE produces the lowest held-out action error. The random 8-dimensional subspace produces more than 180 times the FE action error and reaches 32.6% success, compared with $65.1 \pm 1.0\%$ for the FE. FOMAML reaches the highest success among these methods at 66.5% and the lowest normalized cost at 1.064. Its action error is more than 380 times the FE error. We therefore see different outcomes for task completion and reproduction of the demonstrated policy. Functional PCA, nearest-expert retrieval, and the stored-policy simplex use complete analytical training policies rather than sampled trajectories; the dagger in the table marks this additional information. The stored-policy simplex forms a convex mixture of 8 selected expert policies.

We also sweep the number of context pairs for 3 independently trained checkpoints. Increasing the context from 4 to 8 pairs reduces mean query-action RMSE from 0.0763 ± 0.0373 to 0.00174 ± 0.00029 and increases closed-loop success from $61.0 \pm 15.0\%$ to $99.85 \pm 0.17\%$. With 128 pairs, action RMSE falls further to 0.000219 ± 0.000007, and every rollout succeeds. We see that rank and conditioning reveal fragile fits when the context is small, but they do not guarantee closed-loop success. With 4 context pairs, query-action error has a task-level Spearman correlation of 0.50 with failure and a failure-detection AUROC of 0.73. With 128 pairs, the correlations between failure and the condition number, smallest Gram eigenvalue, projection residual, or query-action error are all near zero, and every bootstrap interval contains zero.

We repeat the FE/FOMAML comparison with matched SiLU architectures and physical-thrust training targets. Both models use 2 hidden layers of width 256. The FE outputs a 2×8 basis and adapts

Table 11: Planar-quadrotor FE/FOMAML identification.

Method	Thrust RMSE ↓	Success ↑	Mean task cost ↓	Seeds
Policy FE (ours)	$\mathbf{0.000203 \pm 0.000012}$	$\mathbf{0.651 \pm 0.010}$	$\mathbf{2.211 \pm 0.076}$	3
FOMAML	0.01439 ± 0.00186	0.633 ± 0.003	2.379 ± 0.066	3

Mean $\pm$ standard deviation over 3 matched seeds. Both methods use the same 512 training tasks, 32 held-out tasks, 128 context pairs, query trajectory, and closed-loop resets. Both adapt using demonstration action loss in physical thrust coordinates.

Table 12: Continual planar-quadrotor adaptation.

Method	Current-task RMSE ↓	Retained-task degradation ↓	Revisit ratio ↓
FE, current coordinate (ours)	0.0174 ± 0.0055	$7.85\text{e}{-2} \pm 7.18\text{e}{-2}$	5.884 ± 8.123
FE, cached RLS (ours)	$\mathbf{0.0121 \pm 0.0030}$	$\mathbf{1.94\text{e}{-6} \pm 3.50\text{e}{-6}}$	0.435 ± 0.311
FOMAML, reset	0.0648 ± 0.0111	$1.50\text{e}{-1} \pm 1.67\text{e}{-1}$	$\mathbf{0.094 \pm 0.053}$
FOMAML, continual	0.0815 ± 0.0099	$3.64\text{e}{-2} \pm 2.21\text{e}{-2}$	0.871 ± 0.574
FOMAML, cached	0.0827 ± 0.0090	$1.03\text{e}{-3} \pm 9.17\text{e}{-4}$	0.987 ± 0.724

Mean $\pm$ standard deviation across 6 fixtures after 16 observations for each target. Retained-task degradation is the increase in held-out action MSE relative to the best earlier estimate. Revisit ratio divides the final return-task MSE by the first-task MSE. Task identities are provided to cached methods.

only 8 coefficients, whereas FOMAML outputs the 2 rotor thrusts directly and adapts all 68,610 network parameters. We train both methods for 5,000 updates on the same 512-task datasets, select the FOMAML inner loop on validation tasks, and freeze every setting before held-out evaluation. Table 11 reports the resulting 3-seed comparison. The FE reduces held-out thrust error by more than an order of magnitude and also achieves slightly higher closed-loop success and lower task cost. This experiment evaluates demonstration-based identification; optimization using the online task objective is a separate synthesis setting.

Continual task adaptation. We next evaluate identification when the target changes over time. For the FE, we update the 8-dimensional coefficient with recursive ridge regression. Given a new normalized state–action pair $(\widehat{x}_t, u_t)$ and design matrix $\Psi_t = G_\theta(\widehat{x}_t)$, we use

$$\begin{aligned} K_t &= P_{t-1}\Psi_t^\top \left(I + \Psi_t P_{t-1}\Psi_t^\top\right)^{-1}, \\ c_t &= c_{t-1} + K_t\left(u_t - \Psi_t c_{t-1}\right), \\ P_t &= P_{t-1} - K_t\Psi_t P_{t-1}. \end{aligned} \tag{19}$$

When task identity is available, we retain a separate (c_τ, P_τ) for each task τ. Returning to a task restores its previous recursive state while leaving the shared basis fixed.

We compare 2 FE protocols with 3 FOMAML protocols. *FE, current coordinate* starts a new recursive estimate whenever the target changes, while *FE, cached RLS* restores the task-specific state. *FOMAML, reset* returns to the meta-trained initialization whenever the target changes, *FOMAML, continual* carries one adapted network through the complete sequence, and *FOMAML, cached* retains one adapted network per task. The labels A–D denote 4 distinct goal-reaching tasks, each with a different target horizontal and vertical position. The numerical targets vary across the 6 independently trained fixtures, but every fixture follows the same $A \to B \to C \to A \to D \to B \to A$ sequence. After each target change, we provide 1, 2, 4, 8, and 16 new expert state–action pairs. We then measure action error on later expert states that were not used for adaptation. We supply task boundaries and identities to the cached methods. We train the FOMAML initialization with Adam, but adapt it with the same first-order SGD inner update used during meta-training. The continual and cached variants retain adapted network weights, not Adam momentum, across targets.

Figure 10 shows adaptation throughout the sequence. Cached FE–RLS reaches current-task action RMSE 0.0121 ± 0.0030 after 16 observations, compared with 0.0648 ± 0.0111 for reset FOMAML and 0.0815 ± 0.0099 for continual FOMAML. Table 12 reports the complete comparison. Caching reduces the FE's retained-task degradation from 7.85×10^{-2} to 1.94×10^{-6} MSE. Continual FOMAML has retained-task degradation 3.64×10^{-2}. In this known-task experiment, the small recursive FE state therefore supports rapid updates and near-lossless resumption when an earlier task returns.

Synthesis. We next evaluate demonstration-free synthesis when we remove an entire behavior family from offline training. We test hover, cruise, aggressive transfer, and vertical-rate tracking.

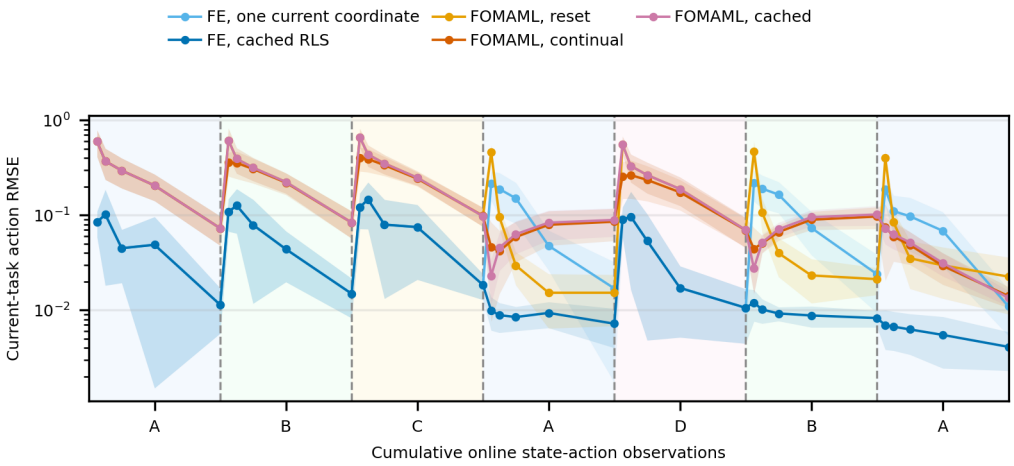


Figure 10: A–D denote 4 planar-quadrotor goal-reaching tasks with different target positions; repeated letters mark the return of an earlier target. Within each shaded block, points show action RMSE on unused expert states after 1, 2, 4, 8, and 16 new state–action pairs. Curves and bands show the mean and 1 standard deviation across 6 independently trained fixtures. Cached FE–RLS resumes its stored recursive estimate when tasks A and B return.

Table 13: Planar-quadrotor family synthesis.

Method	Evaluated family	Success (%), full / omitted	Teacher RMSE (m), full / omitted
Policy FE (ours)	Hover	100.0/100.0	0.242/0.233
Policy FE (ours)	Cruise	100.0/100.0	0.194/0.189
Policy FE (ours)	Aggressive transfer	91.1/100.0	0.394/0.452
Policy FE (ours)	Vertical rate	50.0/49.5	0.172/0.215
FOMAML	Hover	100.0/100.0	0.289/0.293
FOMAML	Cruise	100.0/100.0	0.283/0.277
FOMAML	Aggressive transfer	100.0/100.0	0.458/0.459
FOMAML	Vertical rate	50.0/50.0	0.258/0.257

No method receives a test-time demonstration. Full denotes training on all 4 behavior families; omitted removes the evaluated family. Success uses 192 perturbed closed-loop trials. Teacher RMSE is planar-position trajectory error in meters. Entries report full / omitted.

For each family, we compare adaptation from a model trained on all 4 families with adaptation from a matched model trained without the evaluated family. Both methods minimize the same family-specific task cost for 500 gradient steps and receive no expert trajectory at test time. The FE updates its 8 coefficients in a standardized two-layer, width-256 SiLU policy space, whereas FOMAML updates all 17,922 policy parameters. We evaluate each adapted controller on 12 held-out tasks from 16 perturbed initial conditions, giving 192 closed-loop trials per entry in Table 13.

Omitting a family has little effect on binary success for hover, cruise, or vertical-rate tracking. Aggressive-transfer success increases from 91.1% to 100.0% in the omitted-family condition, although its teacher-trajectory RMSE increases from 0.394 to 0.452 m. Vertical-rate RMSE similarly increases from 0.172 to 0.215 m. These results show that the omitted-family test is more sensitive in trajectory fidelity than in its binary success criterion.

Success and teacher likeness measure different outcomes. FE and FOMAML both reach 100% success on hover and cruise. FOMAML exceeds FE on aggressive transfer when that family appears in training (100.0% versus 91.1%), while both reach 100% when it is omitted. Their vertical-rate success differs by at most 0.5 percentage points. The FE nevertheless has lower teacher RMSE in every family and training condition. Thus, the eight-dimensional FE search more faithfully reproduces the reference trajectory, while full-network FOMAML adaptation provides a success advantage only for the trained-all aggressive-transfer condition under this budget. Neither method exceeds 50% success on vertical-rate tracking, so conclusions for that family remain limited.

Coefficient geometry and policy kernel. The geometry of the learned space enables diagnostic analysis of the learned policies. Let $J_c(x) = \partial \pi_c(x)/\partial c$ and define the occupancy-weighted metric $M = \mathbb{E}_{x\sim\rho}[J_c(x)^\top J_c(x)] + \lambda I$; the induced matrix-valued kernel is $K(x, x') = J_c(x)M^{-1}J_c(x')^\top \in \mathbb{R}^{2\times 2}$. Its rank is at most k, so the singular spectrum of the stacked context Jacobian reveals which coefficient directions a demonstration identifies, while its condition number distinguishes numerical identifiability from mere full rank. Across 3 training seeds, the sampled design has numerical rank 8 and energy effective rank 6.92 ± 0.28, while $7.60 \pm 1.36\%$ of the kernel's squared energy lies in cross-rotor entries, directly showing that the learned perturbations coordinate total thrust and torque rather than treating the rotors as independent scalar regressions. Figure 11 shows the corresponding rotor covariance and spectral diagnostics.

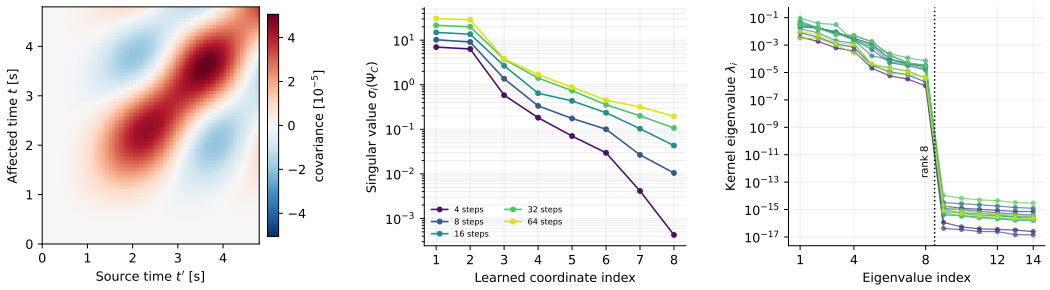


Figure 11: Kernel diagnostics for the standardized 8-basis, 2-layer, width-256 SiLU planar-quadrotor FE. From left to right: the closed-loop horizontal-position covariance across time for one representative held-out task, the mean excitation of each coefficient direction across 12 held-out tasks as the context grows, and the trajectory-kernel spectrum for those tasks. The model is identified from 64 context state–action pairs for the covariance and kernel panels. The spectrum drops to the shared numerical floor after the eighth eigenvalue, consistent with the kernel's rank-eight bound.

Identification uncertainty. The learned coefficient model also lets us propagate uncertainty from policy identification to actions and trajectories. With ridge design matrix A_C and estimated action-noise variance $\hat{\sigma}^2$, we use $\Sigma_c = \hat{\sigma}^2 G^{-1} A_C^\top A_C G^{-1}$, where $G = A_C^\top A_C + \lambda I$, and obtain the local cross-state action covariance $\mathrm{Cov}[u(x), u(x')] = J_c(x)\Sigma_c J_c(x')^\top$. Linearizing the closed-loop rollout then propagates Σ_c to a trajectory uncertainty tube. The sharp identifiability transition occurs between 4 and 8 context states: the mean context condition number falls from 2019.6 ± 1480.4 to 23.7 ± 4.4, and the mean terminal position error falls from $67.38 \pm 44.19\,\mathrm{m}$ to $0.0387 \pm 0.0014\,\mathrm{m}$. With 64 context states, the corresponding values are 11.8 ± 1.6 and $0.0373 \pm 0.0013\,\mathrm{m}$. Across 36 evaluations comprising 3 training seeds and 12 held-out tasks, Monte Carlo coefficient samples closely match the predicted terminal position standard deviations, and nominal 95% trajectory ellipses attain $94.70 \pm 0.75\%$ coverage. These results establish the kernel as a practical tool for experimental design, cross-actuator influence analysis, and first-order uncertainty propagation; they do not imply a globally calibrated uncertainty model outside the sampled occupancy.

Held-out composition. We finally train on 25 policies comprising hover and axis-aligned transfers, remove all diagonal transfers from the offline dataset, and test whether the learned space can compose horizontal and vertical motion into diagonal goal reaching. We evaluate 8 held-out diagonal tasks from 64 common disturbed resets per task, giving 512 closed-loop trials per method. Identification from 1 demonstration achieves 100.0% success and a 1.002 cost ratio, whereas nearest-axis retrieval achieves only 10.9% success and an 8.166 cost ratio. The zero-demonstration arithmetic candidate $c_{\mathrm{horizontal}} + c_{\mathrm{vertical}} - c_{\mathrm{hover}}$ attains 89.5% success and a 1.314 cost ratio. Removing half the context retains 100.0% success and a 1.003 cost ratio. Huber regression with 2 gross action outliers attains 87.5% success, but 1 of the 8 diagonal tasks fails catastrophically, producing a mean cost ratio of 1.31×10^6. Missing velocity information and dense action corruption reduce success to 20.3% and 4.1%, respectively. These failures show that a robust regression solver cannot recover information that is absent from the context.

B.5 LINEAR-QUADRATIC CONTROL

With $x = (q_1, q_2, \dot{q}_1, \dot{q}_2)$ and a scalar force applied to the first mass, the continuous-time system is $\dot{x} = A_c x + B_c u$, where

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & 1 & -0.18 & 0 \\ 1 & -2 & 0 & -0.18 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^\top. \tag{20}$$

We compute the exact zero-order-hold discretization (A, B) at $\Delta t = 0.05\,\mathrm{s}$ and define each task by $Q_\tau = \mathrm{diag}(q_{1,\tau}, q_{2,\tau}, q_{v,\tau}, q_{v,\tau})$ and scalar $R_\tau = r_\tau$. We sample $q_{1,\tau}, q_{2,\tau} \in [0.8, 5.0]$, $q_{v,\tau} \in [0.08, 0.50]$, and $r_\tau \in [0.05, 0.50]$. For every task, we solve the discrete algebraic Riccati equation and use the resulting optimal policy $u = -K_\tau x$ as the expert. We generate every LQR expert from these equations and do not import an external checkpoint. Let P_τ denote the converged Riccati value matrix and

$$K_\tau = \left(R_\tau + B^\top P_\tau B\right)^{-1} B^\top P_\tau A. \tag{21}$$

Although evaluation uses a finite rollout of $T = 150$ steps, we include the terminal value $x_T^\top P_\tau x_T$:

$$J_{\tau,T} = \sum_{t=0}^{T-1} \left(x_t^\top Q_\tau x_t + u_t^\top R_\tau u_t\right) + x_T^\top P_\tau x_T. \tag{22}$$

This finite computation equals the infinite-horizon cost when the Riccati controller is used, making $u_t = -K_\tau x_t$ the exact reference for every reported rollout horizon.

Data collection. For each of 3 seeds, we generate 16 task policies, use 12 for FE training, and reserve 4 complete policies for evaluation. For each task, we draw 256 context states and 256 disjoint query states from a zero-mean Gaussian with coordinate scales $(1.5, 1.5, 1, 1)$ and label every state with its exact LQR action. We train an 8-basis FE, $u = G_\theta(x)c_\tau$, where a 2-hidden-layer network with 64 tanh units per layer produces $G_\theta(x) \in \mathbb{R}^{1\times 8}$. At each of 800 Adam updates, we infer one coefficient vector per training task from its 256 context pairs using (4) with ridge regularization 10^{-3} and update the shared basis using action MSE on the disjoint query states.

Policy identification. For each held-out LQR task, we infer c_τ from $n \in \{2, 4, 8, 16, 32, 64\}$ state–action pairs, evaluate action error on 256 disjoint states, and fit the equivalent linear gain $\widehat{K}_\tau$ to the FE outputs. With 64 context pairs, the FE attains mean action RMSE 0.0648 ± 0.0153 and mean gain error $\|\widehat{K}_\tau - K_\tau\|_F = 0.0170 \pm 0.0065$ across 3 seeds. Executing $\widehat{K}_\tau$ and the exact Riccati gain from common random initial states gives a mean cost ratio of 1.000020 ± 0.000016. Sparse coefficient identification therefore recovers a feedback law whose infinite-horizon cost is within approximately 0.002% of exact LQR.

Policy synthesis. We use the same frozen 8-basis FE trained for identification and optimize only c_τ for each held-out quadratic objective. Synthesis receives the system dynamics and cost but no expert state–action pairs. We differentiate through the closed-loop rollout, optimize the 8 coefficients with Adam, and apply an L-BFGS refinement. Across 12 held-out objectives from 3 seeds, the synthesized policies attain a mean cost ratio of 1.000433 ± 0.000254 relative to exact LQR. Individual ratios range from 1.000075 to 1.001072. For comparison, projecting the unavailable exact controller into the FE span produces a mean ratio of 1.000572 ± 0.000373. Objective-driven coefficient optimization can slightly outperform action-space projection because it directly minimizes control cost rather than policy-reconstruction error.

Policy-kernel analysis. We also evaluate the policy kernel on 64 held-out states. With scalar actuation it is $K_\theta(x, x') = G_\theta(x)G_\theta(x')^\top$, rather than the matrix-valued action kernel used for the quadrotor and legged systems. Across all 3 seeds, its empirical numerical rank is 8, equal to the selected basis count. This verifies directly that the learned policy family uses all 8 available coefficient directions on the sampled state distribution.

Table 14: Coefficient-identification latency.

System	Bases k	CPU (ms)	GPU (ms)
Go1	2	0.358	0.804
Go1	4	0.369	0.751
Go1	8	0.401	0.759
Go1	16	0.486	0.779
G1	8	0.675	0.714

Warmed median wall-clock time for complete identification from 128 context pairs. All Go1 rows use matched standardized checkpoints and the same 10-warmup, 100-repeat timing protocol.

Table 15: Information available to each comparison.

Method	Traj.	Stored policies	Online demo.	Online variables	Online optimizer
Policy FE (ours)	yes	no	yes	k coefficients	ridge least squares
Learned-feature ridge	yes	no	yes	$n_u d$ head entries	ridge least squares
FOMAML	yes	no	yes	all parameters	action-MSE gradient descent
Pooled BC	yes	no	no	none	none
Adapted pooled BC	yes	no	yes	all parameters	action-MSE gradient descent
Functional PCA	no	yes	yes	k coefficients	ridge least squares
Nearest expert	no	yes	yes	policy index	context-error retrieval
Stored-policy simplex	no	yes	yes	simplex weights	constrained action regression
CEM synthesis	yes	no	no	k coefficients	task-return CEM

"Stored policies" denotes access to complete policy functions or evaluations beyond the sampled trajectory dataset.

Coefficient-identification latency. We measure online coefficient inference after training and loading the basis. For 128 context state–action pairs, we evaluate the basis, construct the regularized Gram system, and solve for the coefficient vector. We place the inputs on the timed device before starting the clock and report the median wall-clock time. The Go1 measurements use 10 warmup repetitions followed by 100 timed repetitions; the G1 measurements use 5 warmups and 31 timed repetitions. We synchronize the GPU around each repetition. The measurement excludes data collection, simulation, training, checkpoint loading, disk access, and host–device transfer. Table 14 reports the complete solve: G1 identification with 8 bases requires 0.675 ms on CPU and 0.714 ms on GPU, while the 8-basis Go1 FE requires 0.401 ms on CPU and 0.759 ms on GPU.

B.6 FE BASELINES AND ABLATIONS

Table 15 distinguishes methods trained from sampled trajectories from comparisons that require complete stored policies. FE identification, the learned-feature ridge head, and FOMAML receive the same online expert context but adapt different objects, while task-aware CEM receives the control objective instead of an expert demonstration.

B.7 LEARNED-FEATURE RIDGE-HEAD BASELINE

The closest generic meta-learning analogue of our identification procedure is a shared nonlinear feature extractor followed by a task-specific linear head fit from context data. This family includes the closed-form ridge inner loop of R2-D2 (Bertinetto et al., 2019), the differentiable convex inner loop of MetaOptNet (Lee et al., 2019), and the feature-reuse interpretation formalized by ANIL (Raghu et al., 2020). ALPaCA additionally places a learned Gaussian prior on the linear-head weights and performs Bayesian linear regression (Harrison et al., 2018), closely paralleling our coefficient uncertainty calculation.

We instantiate a bias-free learned-feature regression baseline for control as

$$\widehat{x} = (x - \mu_x)/s_x, \qquad \widehat{u}_\tau(x) = W_\tau h_\eta(\widehat{x}), \qquad u_\tau(x) = \mathrm{clip}(\mu_u + s_u \odot \widehat{u}_\tau(x)), \tag{23}$$

where $h_\eta(\widehat{x}) \in \mathbb{R}^d$ is shared across tasks and the context set determines $W_\tau \in \mathbb{R}^{n_u \times d}$ by differentiable ridge regression. Disjoint query trajectories train h_η. The implementation contains no task-specific intercept; μ_u is the shared training-set action mean. Unlike FE, which adapts one coefficient vector $c \in \mathbb{R}^k$ shared across all action rows, this baseline adapts $n_u d$ head entries independently. We match offline trajectories, trajectory-disjoint context/query splits, normalization, optimizer budgets, online context, and evaluation resets.

Table 16: Learned-feature ridge-head results. Values are means ± sample standard deviations across 4 Go1 seeds and 3 quadrotor seeds. Quadrotor action error is measured in standardized-action coordinates.

System	d	Adapted entries	Action RMSE ↓	Success/survival ↑
Go1	2	24	0.1398 ± 0.0010	$88.1 \pm 6.4\%$
Planar quadrotor	4	8	0.001305 ± 0.000225	$61.3 \pm 0.4\%$

We evaluate the baseline on Go1 and the planar quadrotor. Both use a 2-layer, width-256 SiLU feature network and ridge regularization 10^{-3}. On Go1, we train on the same 28 policy functions and 32 trajectories per policy used by FE, compare $d \in \{1, 2\}$ using validation error, and report $d = 2$ across 4 seeds. Training uses 64 context points per episode for 250,000 updates; evaluation uses 32 context pairs, one disjoint query trajectory, and 4 fresh-reset rollouts per policy. On the quadrotor, we use $d = 4$, which gives 8 adapted head entries and therefore matches the FE's 8 online coefficients. We train for 5,000 updates on 512 tasks and evaluate 32 held-out tasks from 128 context pairs and 64 common closed-loop scenarios per task across 3 seeds.

On Go1, FE reduces action RMSE from 0.1398 to 0.0521, reduces combined tracking RMSE from 0.442 to 0.133, and increases mean survival from 88.1% to 97.8%. On the planar quadrotor, the ridge head is more accurate than FOMAML in action space but remains less accurate than FE; its 61.3% success rate is also below FE's 65.1%. These comparisons test efficient demonstration-conditioned adaptation. Semantic arithmetic and objective-driven synthesis are not assumed capabilities of the ridge-head baseline.

B.8 Comparison with a learned subspace of policies

Gaya et al. (2022) learn a low-dimensional affine subspace of complete policy-network parameters for online adaptation. In the 2-anchor case, a scalar coordinate $\alpha \in [0, 1]$ defines $\theta(\alpha) = (1 - \alpha)\theta_0 + \alpha\theta_1$ and $\pi_\alpha^{\mathrm{Gaya}}(x) = \pi_{\theta(\alpha)}(x)$. We train the anchors jointly by reinforcement learning while sampling coordinates from the simplex and holding them fixed during each rollout. At test time, the method evaluates candidate coordinates from rollout returns and selects the coordinate with the largest observed reward.

This formulation is not directly equivalent to the standard policy function encoder considered in this work. Our standard FE represents $\pi_c(x) = G_\theta(x)c$ for $c \in \mathbb{R}^k$, where the FE combines learned bases in action-function space rather than network-parameter space. Moreover, the standard FE identifies c from state–action demonstrations by linear regression, whereas Gaya selects its coordinate from closed-loop returns. A direct comparison would therefore confound the location of interpolation, the dimension and constraints of the online coordinate, and the information available during adaptation.

To obtain a controlled comparison, we construct a constrained FE line that matches the dimensionality and simplex geometry of the 2-anchor Gaya model:

$$\pi_\alpha^{\text{FE-line}}(x) = (1 - \alpha)g_0(x) + \alpha g_1(x). \tag{24}$$

The Gaya and FE lines consequently use the same 2-dimensional simplex coordinate $(1 - \alpha, \alpha)$, but interpolate different objects: Gaya interpolates every policy-network parameter, whereas the FE line interpolates the 2 policies' action outputs. We introduce this constrained FE only for the matched comparison; it does not replace the standard, unrestricted FE used in our identification experiments.

We implement the Gaya policy line from the formulation of Gaya et al. (2022). Following that construction, we replace each affine policy layer by 2 parameter anchors, apply the same simplex coordinate to every layer, hold the sampled coordinate fixed throughout each rollout, condition the critic on the coordinate, optimize both anchors jointly with PPO, and use the squared parameter cosine as an anti-collapse penalty. We validate the implementation by testing endpoint equivalence and parameter interpolation, measuring anchor separation, plotting coordinate–return landscapes, and evaluating reward-only coordinate selection across 3 training seeds. The trained anchors attain a mean squared parameter cosine of 2.84×10^{-5}, an endpoint action RMS difference of 0.418, and a mean endpoint velocity separation of $0.678\,\mathrm{m\,s^{-1}}$. Under 50% actuator strength, reward-only coordinate selection improves upon the fixed policy in all 3 seeds by a mean return of 211.6. We

Table 17: Pendulum coordinate-selection returns.

Method	All configurations	Long pendulum	Weak actuator
Fixed PPO	-238.3	-483.4	-322.1
Gaya	-270.3	-428.5	-306.9
Policy FE line (ours)	-278.7	-450.1	-334.3

The aggregate row averages all 10 test configurations. The other rows show the 2 configurations where adaptation most clearly improves upon fixed PPO. Higher is better.

see that the anchors remain distinct and that reward-only coordinate selection improves the fixed policy. These checks validate the qualitative behavior required by the method, though our simulator, reward, and PPO implementation differ from those used by Gaya et al. (2022).

We use this implementation in 2 complementary comparisons. In the dataset-matched supervised comparison, we train the standard FE, constrained FE line, behavior-cloned Gaya line, and coordinate-conditioned MLP using the same fixed expert demonstrations. This experiment isolates policy representation, identification, and semantic interpolation without giving Gaya additional reinforcement-learning experience. In the matched closed-loop comparison, we train Gaya-RL and FE-RL from scratch using the same PPO implementation, interaction budget, reset schedule, action distribution, critic conditioning, and coordinate-sampling protocol. This second experiment evaluates reward-only system adaptation under Gaya's intended training conditions.

Pendulum. We evaluate the fixed PPO policy, Gaya policy line, and FE policy line on 10 pendulum configurations that vary mass, length, gravity, damping, and actuator strength. We train the FE to reproduce the 2 policies at the endpoints of the Gaya policy line. For each test configuration, we evaluate 11 equally spaced coordinates using the same adaptation rollouts for both methods. We then fix the selected coordinate and evaluate it on 32 new resets. Table 17 reports the resulting returns across all configurations and for the 2 most challenging perturbations.

We find that the FE approximately reproduces the adaptation behavior of the Gaya endpoints, but neither learned policy line consistently improves over the fixed policy on this system. Because the tested pendulum variations produce little useful separation among the policies, we treat this experiment as a diagnostic rather than the primary comparison.

HalfCheetah task-family experiment. We independently train HalfCheetah expert policies at target speeds $v \in \{0.5, 1.25, 2.0, 2.75, 3.5\}\,\mathrm{m\,s^{-1}}$. We train additional evaluation-only experts at the held-out midpoint speeds $v \in \{0.875, 1.625, 2.375, 3.125\}\,\mathrm{m\,s^{-1}}$. For each training speed, we collect 128 independently reset trajectories of 500 simulation steps. We train every supervised representation using this same fixed dataset.

We compare a 4-basis FE, a 2-basis FE line, a behavior-cloned Gaya parameter line, and a coordinate-conditioned MLP. For each held-out speed, we provide between 4 and 128 state–action context pairs. We identify the 4-basis FE coefficients by ridge regression and identify the 1-dimensional coordinates of the policy-line models by bounded numerical optimization. We then freeze the identified policy and measure action error on disjoint expert trajectories. With 4 context pairs, the 4-basis FE, Gaya line, 2-basis FE line, and conditioned MLP obtain mean held-out action RMSEs of 0.250, 0.277, 0.344, and 0.354, respectively. With 128 context pairs, the corresponding errors are 0.232, 0.271, 0.327, and 0.354. We therefore observe that the 4-basis FE identifies held-out policies more accurately from sparse demonstrations than the tested 1-dimensional policy lines. This is not a dimension-matched result: the 4-basis FE estimates 4 coefficients, while the policy-line methods estimate 1 coordinate. The 2-basis FE line supplies the dimension-matched FE comparison.

HalfCheetah closed-loop adaptation. We separately train Gaya-RL, FE-RL, a coordinate-conditioned MLP, and a fixed PPO policy using the same PPO implementation, interaction count, reset schedule, action distribution, and coordinate-sampling protocol. For Gaya-RL, we interpolate between 2 complete network-parameter anchors. For FE-RL, we interpolate between 2 action-space basis policies. For both representations, we use a diversity penalty during training to discourage coordinate collapse.

Table 18: HalfCheetah coordinate-selection returns.

Method	Nominal	Velocity masking	50% actuator strength
Fixed PPO	**1796.4**	542.0	42.6
Gaya	1431.6	673.1	**254.0**
Policy FE (ours)	1781.5	569.2	64.5
Conditioned MLP	1460.7	**916.2**	39.2

We select each adaptive policy using 8 common adaptation resets and evaluate it on 32 new matched resets. Higher is better; bold entries mark the best method in each condition.

We evaluate velocity masking, velocity noise, full-observation noise, one-step observation delay, 75% and 50% actuator strength, and 1 weakened joint. For each variation, we evaluate 11 coordinates on the same 8 adaptation resets, for a total of 88 online episodes per method. We select the coordinate with the highest adaptation return, fix it, and evaluate it on 32 new matched resets. Table 18 reports the nominal result and the 2 perturbations that produce the clearest adaptation gains.

We see that the fixed PPO policy remains strongest under nominal conditions, the conditioned MLP performs best under velocity masking, and Gaya performs best under 50% actuator strength. Under most milder perturbations, the fixed policy remains best or competitive with the adapted policies.

We also train variation-specific oracle policies. The oracle obtains 1362.2 under 50% actuator strength, compared with 254.0 for the best adapted policy. We therefore observe substantial remaining performance headroom in all tested low-dimensional adaptation spaces.

Diversity ablation. We repeat the closed-loop training with and without the diversity penalties. For Gaya, the squared cosine similarity between the 2 endpoint parameter vectors increases from approximately 2.8×10^{-5} with the penalty to 9.3×10^{-2} without it. For the FE, the corresponding action-space similarity increases from 1.1×10^{-3} to 8.7×10^{-1}. Thus, the penalties produce measurably different endpoint policies. However, the more diverse models do not uniformly obtain higher returns. We therefore find that endpoint diversity is necessary to create a nondegenerate adaptation coordinate, but does not by itself guarantee useful adaptation.

Summary. We find no single representation that performs best in every setting. Gaya produces the largest improvement under severe actuator loss, while the conditioned MLP performs best under velocity masking. The FE produces the lowest sparse-demonstration identification error among the tested representations and retains near-fixed-policy performance under nominal conditions. These results separate 2 adaptation settings: Gaya performs reward-only coordinate selection, whereas the FE directly supports least-squares policy identification from state–action demonstrations.

B.9 Comparison with PULSE-style latent policy representations

PULSE (Luo et al., 2024) learns a low-dimensional latent representation of humanoid motion by distilling a motion imitator into an encoder, action decoder, and state-conditioned prior. In the original formulation, PULSE conditions the imitator and encoder on a motion reference. The encoder observes the physical state and desired reference and produces the time-varying latent variable $q_\eta(z_t \mid x_t, g_t)$, where g_t denotes the reference-motion observation. The decoder maps the current physical state and latent variable to the action $u_t = D_\phi(x_t, z_t)$, while PULSE trains a prior $p_\rho(z_t \mid x_t)$ to approximate the encoder without observing the reference. After distillation, the decoder and prior define a reusable motion space in which a higher-level reinforcement-learning policy can select a new latent z_t at each time step.

This construction differs from the policy FE studied here in 2 important ways. First, the FE identifies 1 coefficient vector $c \in \mathbb{R}^k$ from an expert context and holds the policy $u_t = G_\theta(x_t)c$ fixed throughout the rollout. The coefficient identifies an entire feedback policy rather than a time-indexed sequence of motion commands. Second, PULSE provides a dynamic motion prior for hierarchical reinforcement learning rather than a method for identifying 1 fixed policy from a sparse state–action demonstration. A direct comparison between the standard FE and the original PULSE execution model would therefore change both the temporal bandwidth of the representation and the information available online.

Table 19: G1 FE and PULSE information contracts.

Method	Policy	Dimension	Schedule	Coordinate source
Policy FE (ours)	$G_\theta(x)c$	8	Fixed	Ridge regression
Static PULSE-style	$D_\phi(x, z)$	8	Fixed	Context encoder and refinement
Dynamic PULSE prior	$D_\phi(x, z_t)$	32	Per step	$p_\rho(z_t \mid x_t)$
Dynamic PULSE posterior	$D_\phi(x, z_t)$	32	Per step	$q_\eta(z_t \mid x_t, g_t)$

Method	Online context	Training distribution	Training target
Policy FE (ours)	64 state–action pairs	Fixed expert trajectories	Query actions
Static PULSE-style	64 state–action pairs	Fixed expert trajectories	Query actions
Dynamic PULSE prior	None	Student-visited states	Teacher actions
Dynamic PULSE posterior	Continuous reference g_t	Student-visited states	Teacher actions

Method	Training budget	Parameters	Matched FE comparison
Policy FE (ours)	225k updates	0.678M	—
Static PULSE-style	250k updates	1.422M	yes
Dynamic PULSE prior	10^7 transitions	13.627M	no
Dynamic PULSE posterior	10^7 transitions	13.627M	no

Only the FE and static PULSE-style baseline receive the same offline dataset and sparse expert context. Only the dynamic posterior receives a reference or phase at execution. The static model minimizes reconstruction plus KL; the dynamic model adds temporal regularization. The dynamic posterior is a privileged diagnostic rather than a matched competitor.

We consequently implement 2 complementary PULSE baselines. The first is a *static PULSE-style latent policy* constructed specifically for the matched comparison with the FE. The second is a *dynamic PULSE reproduction* that retains the principal temporal and conditional-prior structure of the original method. Table 19 summarizes their differences.

Static PULSE-style baseline. The static baseline replaces the linear FE decoder with a nonlinear latent policy. Given the context set $C_b = \{(x_i, u_i)\}_{i=1}^{n_C}$ from behavior b, a permutation-invariant context encoder computes

$$h_b = \frac{1}{|C_b|} \sum_{(x,u)\in C_b} e_\eta(x, u). \tag{25}$$

We compute the posterior parameters from this pooled representation:

$$q_\eta(z \mid C_b) = \mathcal{N}\big(\mu_\eta(h_b), \operatorname{diag} \sigma_\eta^2(h_b)\big)\,. \tag{26}$$

We compute the predicted action from the state and 1 sampled latent using the nonlinear decoder, $\widehat{u} = D_\phi(x, z)$. We minimize

$$\mathcal{L}_{\text{static}} = \mathbb{E}_{z\sim q_\eta(z|C_b)} \left[\frac{1}{|Q_b|} \sum_{(x,u)\in Q_b} \|u - D_\phi(x, z)\|_2^2 \right] + \beta D_{\text{KL}} \left(q_\eta(z \mid C_b) \,\|\, \mathcal{N}(0, I)\right), \tag{27}$$

where C_b and Q_b contain samples from disjoint trajectories of the same behavior.

We use an 8-dimensional latent, 3 hidden layers of width 512, SiLU activations, learning rate 10^{-4}, and a KL coefficient annealed to 10^{-2} over the first $50{,}000$ updates. Each update contains all 7 behaviors, with 4 context and 4 query trajectories per behavior and 256 state–action samples in each set. At evaluation, the context encoder initializes the latent from 64 held-out state–action pairs. We then perform 100 gradient steps on the latent alone and hold the resulting value fixed for the complete 20-second rollout.

The static baseline infers 1 rollout-level latent from an unordered expert context, a comparison-specific design that PULSE does not use.

Dynamic PULSE reproduction. We replace the published humanoid, motions, and simulator with the Unitree G1, 7 ProtoMotions references, and MuJoCo. We train through online distillation for 10 million student-visited transitions, corresponding to $4{,}874$ learner updates: 24 simulation workers execute the current student, and the ProtoMotions tracker labels the visited states.

We evaluate the same checkpoint in 2 modes. The *dynamic-prior* controller receives only the physical state; it uses no expert demonstration or online motion reference. The *dynamic-posterior* controller receives the teacher reference at every step, so we use it as a privileged diagnostic of the

Table 20: G1 FE and PULSE evaluation.

Method	Success rate	Descriptor error	Action RMSE	Action jerk
Teacher	99.86% $[99.20, 99.97]$	0.000	0.000	4.852
Policy FE (ours)	97.71% $[96.32, 98.59]$	$\mathbf{0.213}$	0.170	5.615
Static PULSE-style	$\mathbf{99.14}\%$ $[98.14, 99.61]$	0.253	0.277	$\mathbf{3.604}$
Dynamic PULSE prior	84.71% $[81.86, 87.19]$	0.288	0.257	7.276
Dynamic PULSE posterior	87.71% $[85.07, 89.94]$	0.282	$\mathbf{0.130}$	10.012

Each controller uses 700 paired 20-second rollouts. Success brackets are 95% binomial confidence intervals. Lower is better for descriptor error, action RMSE, and jerk. Bold marks the best learned method; the dynamic posterior is not information-matched to the static methods.

decoder rather than an information-matched FE comparison. We evaluate the distilled controller directly instead of training the downstream high-level latent PPO policy used in the published PULSE task experiments. We note that this is not an exact PULSE implementation, but it reproduces the model structure and online-distillation training procedure.

Training dataset. The matched FE and static PULSE-style models use the ProtoMotions dataset described above.

Before sampling, we divide the trajectories into $10{,}752$ training, $3{,}584$ validation, and $3{,}584$ test trajectories. We always draw context and query sets from different trajectories. Consequently, neither method can minimize the episodic objective by reproducing a particular stored trajectory.

We train the dynamic PULSE model on a different but behavior-matched data stream because online distillation is part of its algorithm. Its workers execute the current student on the same 7-behavior library, while the teacher supplies reference-conditioned actions at the states visited by the student. This difference gives the dynamic model recovery-state coverage that the fixed supervised dataset does not provide, but also prevents it from serving as a strictly dataset-matched comparison.

Evaluation protocol. For each of 7 behaviors, we evaluate 20 common random seeds under 5 initialization and disturbance conditions: ordinary perturbed resets, behavior-matched poses, mismatched poses, neutral standing, and mid-rollout pushes. Every controller is therefore evaluated on 700 paired 20-second rollouts. We hold the teacher, FE, and static PULSE coordinates fixed throughout each rollout. The 2 dynamic PULSE variants recompute their latent at every simulation step. We report survival or task success, behavior-descriptor error, teacher-action RMSE, and action jerk. The descriptor contains planar velocity, yaw rate, and root height. It is intentionally compact and should not be interpreted as a complete measure of posture or gait fidelity. Table 20 reports all 4 metrics for the matched static comparison and the 2 dynamic PULSE variants.

Matched static comparison. The static PULSE-style controller obtains a success rate 1.43 percentage points higher than the FE. However, this paired difference is not statistically significant after correction for the 7 tested outcomes ($p_{\mathrm{Holm}} = 0.290$). In contrast, the FE reduces descriptor error from 0.253 to 0.213 and teacher-action RMSE from 0.277 to 0.170. Both differences remain significant after Holm correction ($p_{\mathrm{Holm}} = 9 \times 10^{-4}$). Static PULSE produces substantially smoother actions, reducing the jerk metric from 5.615 to 3.604.

The static latent model also exhibits posterior collapse. At the selected checkpoint, its KL contribution is approximately 5×10^{-8}, and the mean inferred-coordinate norm on the test data is 1.3×10^{-4}. The decoder therefore learns to predict actions primarily from the physical state while largely ignoring the inferred behavior latent. Its high survival and low jerk should consequently be interpreted as the behavior of a conservative pooled state-feedback controller, not as evidence that it successfully identifies behavior-specific static coordinates. The limited diversity of our matched dataset is a likely contributor: with only 7 behaviors, substantially fewer than the broad motion collections used to train PULSE, the decoder can reduce its loss by learning a pooled state–action map instead of using the latent to separate behaviors. This baseline therefore tests the matched-data regime rather than the full-scale PULSE setting. The FE retains a behavior-dependent coordinate and more accurately reproduces the desired expert policy.

Dynamic comparison. The dynamic prior requires no expert context, but the current state alone does not always specify which of the 7 behaviors is desired. It obtains 84.71% success and an action

RMSE of 0.257. Supplying the desired reference continuously through the posterior improves these values to 87.71% and 0.130, respectively. The improvement confirms that the decoder can use its dynamic latent, but it also demonstrates that the instantaneous physical state does not generally determine the behavior.

The posterior obtains lower one-step teacher-action error than the FE, yet its closed-loop success is 10 percentage points lower and its action jerk is nearly twice as large. We see the same separation between action fidelity and closed-loop performance in the FOMAML comparison, where FOMAML reaches higher success despite larger action error. For the dynamic posterior, temporal variation in the inferred latent, student-state distribution shift, and recurrent action errors provide plausible explanations for the lower rollout success.

Neither dynamic result constitutes a matched replacement for FE identification. The prior receives no observation of the desired behavior, whereas the posterior receives a new desired-motion reference at every step. The FE provides the middle ground we target: it receives a finite expert demonstration once, identifies a fixed policy coordinate, and then executes without further expert information.

Summary. The matched comparison supports 2 conclusions. First, the linear coefficient structure of the FE does not prevent it from producing robust humanoid feedback policies: it reaches 97.71% success while more accurately preserving the requested expert behavior than the tested nonlinear static latent model. Second, the nonlinear static model does not automatically learn a useful policy coordinate; in this experiment, its latent collapses and the decoder behaves predominantly as a pooled state-feedback policy.

The dynamic experiment answers a different question. A PULSE-style time-varying latent can reduce action error when it is continuously conditioned on the desired motion, but this requires greater model capacity and additional online information and does not improve long-horizon robustness in the MuJoCo reproduction. We therefore use static PULSE as the matched baseline for policy identification and report dynamic PULSE separately as a structurally closer, but information-unmatched, reference.