STEERCONTROL: WORLD MODEL-GUIDED ADAPTATION OF FROZEN POLICIES TO CHANGING DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unexpected changes in dynamics are often unavoidable for intelligent agents operating in real-world settings. Policies trained under fixed dynamics can suffer substantial performance losses after deployment when the same action produces a different state transition. Many existing adaptation methods address such failures by training on anticipated variations or by relying on deployment rewards, reference trajectories, or parameter updates. We address this problem using a learned dynamics model that requires no supervision beyond what is available in a standard reinforcement learning (RL) setting. SteerControl pairs a frozen policy with a frozen one-step dynamics model, both trained only under nominal dynamics. It uses the difference between predicted and observed transitions to estimate a bounded correction to the policy's action mean without requiring fault labels, external supervision, deployment rewards, or weight updates. Across twelve continuous-control environments with varied actuator faults and physical perturbations, SteerControl recovers return lost by the faulted base policy under constant and switching perturbations. These results suggest that a nominal dynamics model can help an existing model-free policy maintain more of its performance when faced with unseen shifts in action effects during deployment.

1 INTRODUCTION

In practical deployments, robots should remain effective even when their response to control inputs changes after deployment. An actuator may weaken, a payload may alter the system's inertia, or contact conditions may change friction (Peng et al., 2018; Kumar et al., 2021a). In each case, the task and sensor observations may remain familiar, but the same action can produce a different next state. A policy trained under fixed dynamics can therefore lose performance even in states similar to those encountered during training. If this shift goes undetected, the controller may continue to act on outdated assumptions and compound control errors. Recovering during operation is important because collecting new training data, obtaining reward feedback, or replacing an already validated controller may be costly or impractical. A deployed controller should therefore perform well under nominal dynamics and recover useful behavior when the effects of its actions change.

Existing approaches address this problem at different stages. Dynamics randomization and robust or adversarial RL expose a policy to a specified family of changes during training so that it can be deployed without further updates (Peng et al., 2018; Pinto et al., 2017; Mankowitz et al., 2020). Universal policies, meta-RL, and context-conditioned methods are also trained across families of tasks or dynamics. At inference, they use recent experience to adapt to an unknown operating condition, either by updating learned parameters or by inferring physical parameters or latent context. Their adaptation mechanisms are therefore learned in advance from related tasks or dynamics (Yu et al., 2017; Nagabandi et al., 2019; Kumar et al., 2021a). Other methods use data collected during deployment to update policy representations, adapt a dynamics model for planning, identify simulator parameters, or learn a corrective policy (Hansen et al., 2021; Chua et al., 2018; Zhang et al., 2025; Johannink et al., 2019). These methods make different trade-offs, but they typically require at least one of the following: anticipated variation during training, multi-dynamics data, an explicit physical parameterization, deployment rewards or a known task objective, repeated trials, or changes to learned parameters. This raises a more restrictive question: can a controller trained only under nominal dynamics use observed state transitions to compensate for a dynamics change not represented during training while keeping all learned weights fixed?

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

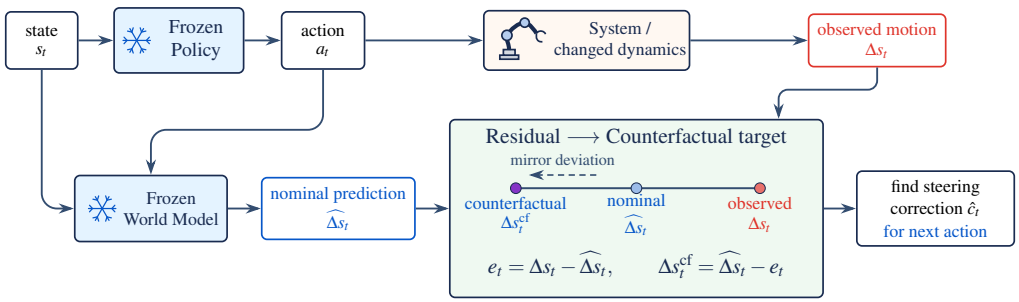


Figure 1: Overview of SteerControl. The frozen dynamics model predicts the nominal state change for the executed action. The difference between this prediction and the observed change defines a residual, which is used to construct a counterfactual target. A correction estimated from this target steers the policy on subsequent steps without updating either set of model weights.

In recent years, world models have attracted renewed interest in both control (Ha & Schmidhuber, 2018; Hafner et al., 2025) and self-supervised representation learning (Assran et al., 2023). On demanding continuous-control benchmarks, model-based RL can reduce the need for environment interactions, but model errors may compound over long rollouts and bias decisions (Chua et al., 2018; Lambert et al., 2020). On the other hand, model-free approaches such as PPO and SAC have often achieved superior performance on these benchmarks and remain widely used, although they often require many environment interactions (Schulman et al., 2017; Haarnoja et al., 2018). Learned dynamic models can nevertheless accelerate the learning of model-free approaches. Providing simulated or imagined experience for additional updates is a well-established strategy for improving their interaction efficiency (Sutton, 1991; Janner et al., 2019; Hafner et al., 2020). We consider a complementary role after training: learned dynamics models can reveal shifts in the transition distribution during deployment and guide strong model-free controllers that lack an explicit mechanism for detecting such shifts.

Based on this view, we introduce SteerControl, a test-time compensation method that pairs a frozen model-free policy with a learned one-step dynamics model, both trained under nominal dynamics and kept fixed at test time. After each transition, the model predicts the nominal state change for the executed action, and the difference between this prediction and the observed change defines a prediction residual. SteerControl uses the frozen model to estimate a bounded correction intended to counteract this residual and applies it to subsequent policy decisions, as illustrated in Figure 1. The method uses only observed transitions: it requires no deployment reward, fault label, dedicated fault sensor, or estimate of the changed physical parameters, and it updates neither policy nor model weights. SteerControl targets shifts in $p(s' \mid s, a)$ for which the encountered states remain sufficiently close to the nominal training distribution and the changed action effects can be compensated through the available actions.

We evaluate SteerControl in twelve simulated continuous-control environments under various types of actuator, mechanical, and environmental perturbations absent from training. The evaluation covers stationary faults and abrupt within-episode switches, with a fault-magnitude sweep and diagnostic tests of temporal averaging. SteerControl recovers a notable share of the fault-induced return loss in many stationary and switching settings and has the highest recovered-gain point estimate among the tested online controllers in most matched comparisons. SteerControl uses a simple yet effective online rule with a small number of hyperparameters and no prescribed fault model. The results show how a frozen nominal dynamics model can turn observed transition discrepancies into local guidance for corrective actions, allowing a frozen policy to compensate for previously unseen changes in action effects.

2 Method: SteerControl

2.1 Problem Formulation and Assumptions

We consider a continuous-control system whose nominal operation is described by the discounted Markov decision process (MDP) $\mathcal{M}_{\text{nom}} = (\mathcal{S}, \mathcal{A}, P_{\text{nom}}, r, \rho_0, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P_{nom} is the nominal transition kernel, r is the reward function, ρ_0 is the initial-state

distribution, and γ is the discount factor. A policy is trained only under the nominal transition kernel $P_{\text{nom}}(s' \mid s, a)$. At deployment, the state and action spaces and the task remain unchanged, but transitions follow an unknown time-indexed kernel,

$$s_{t+1} \sim P_t(\cdot \mid s_t, a_t). \tag{1}$$

The deployment dynamics may be nonstationary: $P_{t+k}(\cdot \mid s, a)$ need not equal $P_t(\cdot \mid s, a)$ within an episode for some $k > 0$. We assume, however, that successive transition kernels remain close over subsequent steps, allowing us to estimate and apply a local correction, rather than changing independently at every step. An unknown perturbation that remains fixed throughout an episode is a special case, for which P_t is identical at every step.

We focus on changes in the conditional transition dynamics rather than changes in the observed state distribution. Let $p_{\text{nom}}(s)$ denote the state marginal induced by nominal rollouts and let $p_t(s)$ denote the marginal induced by the deployment process at step t. We assume that $p_t(s)$ remains close to $p_{\text{nom}}(s)$ over the relevant states, while $P_t(\cdot \mid s, a)$ may differ from $P_{\text{nom}}(\cdot \mid s, a)$. This keeps the policy and the dynamics model within a familiar state region while allowing the effect of an action to change.

This separation is central to the frozen design. Because neural networks can be unreliable on out-of-distribution inputs, the state-marginal assumption keeps policy inputs concentrated in a state region represented during training. The dynamics model receives both state and action, so its use also requires coverage of the actions considered during steering. Random-action augmentation during nominal model training broadens this action coverage, while the correction bound limits action extrapolation during deployment. We assume that these conditions keep the deployed state-action inputs within the represented region even when $P_t(\cdot \mid s, a)$ differs from the nominal transition kernel.

After executing a_t, SteerControl observes only the transition (s_t, a_t, s_{t+1}). It does not receive the task reward, a fault label, or external supervision. The method uses these transitions first to estimate the local dynamics discrepancy and then to compute a temporary correction to later policy decisions. It does not update any learned parameter, and setting the correction to zero recovers the original policy. As with any action-based correction, recovery is limited to changes whose effects can be influenced through the available actions.

2.2 Nominal Policy and Dynamics Model

Let π_θ be a policy trained to maximize expected discounted return in $\mathcal{M}_{\text{nom}}$. SteerControl does not depend on a specific policy architecture or training algorithm, and the policy need not be differentiable. In our experiments, π_θ is a Gaussian model-free policy, and $\mu_\theta(s)$ denotes the location of its action distribution before the fixed action transformation. The correction can also be applied to the final action, so an accessible policy mean is not required. The policy parameters remain fixed during deployment.

We also collect a nominal transition dataset $\mathcal{D}_{\text{nom}} = \{(s_i, a_i, s'_i)\}_{i=1}^N$ and train a one-step dynamics model $f_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{d_s}$. For a transition (s, a, s'), let $\Delta s = s' - s$, with $\Delta s_i = s'_i - s_i$ for sample i. The model predicts the expected state change under nominal dynamics,

$$f_\phi(s, a) \approx \mathbb{E}_{s' \sim P_{\text{nom}}(\cdot \mid s, a)}[\Delta s]. \tag{2}$$

We fit the model by supervised regression,

$$\phi^\star \in \arg\min_\phi \frac{1}{N} \sum_{i=1}^{N} \| f_\phi(s_i, a_i) - \Delta s_i \|_2^2 . \tag{3}$$

Half of the nominal transitions in $\mathcal{D}_{\text{nom}}$ are generated using randomly sampled valid actions, and the remainder are policy-generated. The random-action samples broaden action coverage at nominally encountered states without including any shifted-dynamics data. Together with the state-marginal assumption and the correction bound, this augmentation supports the state-action coverage assumption above and reduces the likelihood of encountering out-of-distribution state-action combinations during deployment.

The dynamics model can be trained simultaneously with the policy or trained offline after policy learning. Once training is complete, both the policy and dynamics model are frozen. The model

is not expected to predict the changed kernel P_t. It provides a one-step nominal reference within its intended input regime, and the difference between this reference and the observed transition provides the adaptation signal. Using an ensemble of dynamics models can reduce the risk of biased estimates in regions with limited data support or due to overfitting.

In our implementation, normalization is applied only within the dynamics model; the base policy receives states in their original coordinates. Each state component s_j is standardized as $(s_j - \mu_{s,j})/\max(\sigma_{s,j}, 0.01)$ using statistics computed only from the dynamics-model training split. Each action component is divided by its corresponding action-space range rather than standardized using empirical action statistics. This preprocessing reduces scale imbalance across the input dimensions and supports stable nominal-model training and reliable action steering procedure. For notational simplicity, we do not introduce separate variables for the normalized states and actions used by f_ϕ.

2.3 Residual-Driven Policy Steering

For an observed transition, let $\Delta s_t = s_{t+1} - s_t$ be the observed state change. We denote the predicted state change by $\widehat{\Delta s}_t = f_\phi(s_t, a_t)$ and the next state by $\hat{s}_{t+1} = s_t + \widehat{\Delta s}_t$. The prediction residual is

$$e_t = s_{t+1} - \hat{s}_{t+1} = \Delta s_t - \widehat{\Delta s}_t, \tag{4}$$

where a_t is the action that was actually executed. The residual measures the local difference between the observed action effect and its nominal prediction. It contains transition noise and nominal-model error, while a persistent component can indicate a change in the transition dynamics. Because the prediction is conditioned on the action actually executed, the residual remains well defined whether that action is unsteered or includes a correction estimated at an earlier step.

SteerControl treats e_t as a local estimate of the dynamics discrepancy and defines the counterfactual next-state target

$$s_{t+1}^{\mathrm{cf}} = \hat{s}_{t+1} - e_t = s_t + f_\phi(s_t, a_t) - e_t. \tag{5}$$

Its compensatory interpretation uses a local additive approximation: the discrepancy represented by e_t is assumed to remain approximately constant over nearby candidate actions. Under this approximation, an action whose nominal-model prediction equals s_{t+1}^{cf} is expected to produce $s_{t+1}^{\mathrm{cf}} + e_t = \hat{s}_{t+1}$ under the deployment dynamics.

To find such an action, let $c \in \mathbb{R}^{d_a}$ be a candidate correction. We add c to the policy mean,

$$\widetilde{\mu}_t(c) = \mu_\theta(s_t) + c. \tag{6}$$

Here, $\mu_\theta(s_t)$ is the base-policy mean at step t. Let $a_t(c)$ denote the action obtained from the mean in Equation 6 while keeping the remaining policy outputs, sampled noise realization, and action transformation fixed. We constrain c to $\mathcal{C} = \{c \in \mathbb{R}^{d_a} : -c_{\max} \preceq c \preceq c_{\max}\}$, where $c_{\max}$ is chosen so that $a_t(c)$ remains within the action region covered by $\mathcal{D}_{\mathrm{nom}}$. We then choose the correction whose nominal prediction is closest to the counterfactual target:

$$\hat{c}_t = \arg\min_{c \in \mathcal{C}} \left\| s_t + f_\phi(s_t, a_t(c)) - s_{t+1}^{\mathrm{cf}} \right\|_2^2. \tag{7}$$

During this optimization, s_{t+1}^{cf} is treated as a fixed target, and gradients are not propagated through its construction. Gradients pass through $a_t(c)$ and the frozen model $f_\phi(s_t, a_t(c))$ only to c; neither θ nor ϕ is updated. The objective seeks a control change that cancels the observed residual without estimating the physical parameters that caused the shift or using the task reward.

Because e_t becomes available only after s_{t+1} is observed, the resulting $\hat{c}_t$ cannot affect a_t and is applied starting at step $t+1$. This one-step transfer assumes that the unknown discrepancy is locally stationary in time, so that the residual observed at step t remains informative at the next step.

The residual uses the action actually executed at step t, denoted by a_t. The observed next state s_{t+1} and the nominal prediction $\hat{s}_{t+1} = s_t + f_\phi(s_t, a_t)$ therefore correspond to the same initial state s_t and action a_t. This remains true whether a_t is an uncorrected policy action, includes a correction from an earlier step, or is selected independently of the policy, for example by random sampling.

For a Gaussian policy, applying a correction c changes $\mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$ to $\mathcal{N}(\mu_\theta(s) + c, \Sigma_\theta(s))$, while leaving the covariance, the subsequent action transformation, and all policy weights fixed.

These distributions refer to the policy before any nonlinear action transformation. We approximately solve Equation 7 using projected gradient descent, projecting c onto $\mathcal{C}$ after each update. The objective is a nonlinear least-squares problem, so a projected or damped Gauss–Newton method can also be used.

Although we use mean steering as the default, the correction can be placed elsewhere in the frozen policy. It may be added to an intermediate embedding and passed through the remaining policy layers, or it may be added directly to the final action. Applying it to the final action does not require gradients through the policy. The same residual objective applies after redefining the correction-to-action map, the dimension of c, and its feasible set.

2.4 Online Adaptation Procedure

At the first step, the policy acts without an estimated correction. After observing the first transition, SteerControl computes $\hat{c}_1$ and initializes the smoothed correction as $\bar{c}_1 = \hat{c}_1$. For each subsequent step $t \geq 2$, the policy uses $\bar{c}_{t-1}$ to produce a_t. After observing s_{t+1}, SteerControl computes e_t and $\hat{c}_t$. Transition noise and nominal-model errors can make individual estimates fluctuate across steps. Averaging these estimates can reduce the variance of the applied correction. We therefore update it recursively,

$$\bar{c}_t = (1-\alpha)\bar{c}_{t-1} + \alpha\hat{c}_t, \qquad \alpha \in (0,1], \qquad t \geq 2, \tag{8}$$

Here α is the weight assigned to the newest estimate. A larger value tracks changes more quickly, while a smaller value provides stronger smoothing. The only persistent adaptive state is $\bar{c}_t$, which is added to the policy mean to produce the next action. Algorithm 1 summarizes the deployment procedure.

Residuals from unsteered policy actions may be cleaner, but reserving such actions for measurement reduces the number of actions that receive compensation. In our experiments, the correction computed with Equation 7 remains effective when the residual comes from an already steered action. SteerControl therefore uses each observed transition both to estimate the current discrepancy and to update the correction, without dedicated unsteered measurement steps.

Algorithm 1: SteerControl online adaptation

Require: s_1; frozen π_θ and f_ϕ, $\mathcal{C}$, α, horizon T
$\bar{c}_0 \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Draw a_t from π_θ with mean $\mu_\theta(s_t) + \bar{c}_{t-1}$
  Execute a_t and observe s_{t+1}
  $e_t \leftarrow (s_{t+1} - s_t) - f_\phi(s_t, a_t)$
  $s^{\mathrm{cf}}_{t+1} \leftarrow s_t + f_\phi(s_t, a_t) - e_t$
  $\hat{c}_t \leftarrow \arg\min_{c \in \mathcal{C}} \| s_t + f_\phi(s_t, a_t(c)) - s^{\mathrm{cf}}_{t+1} \|_2^2$
  if $t = 1$ **then**
    $\bar{c}_t \leftarrow \hat{c}_t$
  else
    $\bar{c}_t \leftarrow (1-\alpha)\bar{c}_{t-1} + \alpha\hat{c}_t$
  end if
end for

3 Experiments

3.1 Experimental Setup

We evaluate SteerControl in 30 stationary and 12 switching fault settings across twelve continuous-control environments: six locomotion tasks (Ant, HalfCheetah, Hopper, Humanoid, Swimmer, and Walker2d), four robotic-arm tasks (FetchReach, PandaReach, Pusher, and Reacher), Pendulum, and LunarLander. FetchReach and PandaReach policies receive a desired goal in their observations; Pusher and Reacher also have episode-specific targets, but these are included in ordinary flat observations. The locomotion tasks, Pusher, Reacher, Pendulum, and LunarLander belong to the Gym/Gymnasium benchmark family (Brockman et al., 2016; Towers et al., 2024); the locomotion tasks, Pusher, and Reacher use MuJoCo physics (Todorov et al., 2012). FetchReach originates from the Fetch robotics suite (Plappert et al., 2018), while PandaReach comes from panda-gym (Gallouédec et al., 2021).

For each environment, all methods use the same Soft Actor-Critic (SAC) policy trained under nominal dynamics (Haarnoja et al., 2018); FetchReach and PandaReach also use hindsight experience replay (Andrychowicz et al., 2017). We fit a five-member dynamics ensemble to nominal transitions collected using an equal mix of frozen-policy and uniformly sampled actions. SteerControl uses the ensemble-mean prediction as f_ϕ. Training configurations are given in Appendix B.

We design the perturbation experiments around SteerControl's central assumption: the transition kernel may change while the distribution of visited states remains close to that seen during nominal training. Agent-centric faults are generally more closely aligned with this assumption, but we also include environment-based perturbations. The experiments cover changes in actuator bias, gain, and action direction; engine power, body mass, and joint damping; and external force, gravity, friction, and drag. Because the environments have different physical properties, a single perturbation type may not apply to all of them. We therefore evaluate at least one perturbation in each environment.

Base-Clean and Base-Perturbed denote the frozen policy without and with the fault. The closest prior work applicable to our setting is the online residual-control framework of Jayasinghe et al. (2026), from which we select three controllers for comparison. Online LMS learns a linear action correction from tracking error. CMAC learns a linear correction on fixed nonlinear features. Cerebellar uses local corrective pathways with fast and slow updates and regulates correction strength from recent reward trends. The trajectory-tracking versions use a stored clean reference. In FetchReach, PandaReach, and Pusher, Online LMS, CMAC, and trajectory-tracking Cerebellar reuse one reference across episodes even when the goal changes. The Cerebellar search also allows goal-directed error based on the current goal for FetchReach and PandaReach. The residual baselines use task-space tracking in FetchReach, PandaReach, and LunarLander, marked † in the tables.

SteerControl uses a default EMA weight of 0.2 to average corrections across environment steps. Rather than estimating $\hat{c}_t$ with a single large gradient update and applying it directly, it uses five projected gradient steps with a default step size of 0.2, then incorporates the estimate into the running average. We keep the number of gradient steps fixed at five across experiments. Online LMS and CMAC also use a fixed configuration across settings.

A Cerebellar configuration shared across settings typically recovered less than 10% of the fault-induced return gap. We therefore evaluated approximately 30 additional Cerebellar configurations per setting and selected the best-performing one using calibration seeds disjoint from final evaluation. The search allowed Cerebellar to retain learned memory across episodes when favored by calibration. For stationary faults, this option assumes that the fault persists across episodes. SteerControl resets its correction at each episode boundary and uses only transitions from the current episode for adaptation, without reference trajectories or reward feedback.

For each setting, every method is evaluated on the same five disjoint, held-out seed batches with eight parallel environment streams per batch. These runs yield at least 40 complete episodes per method. The seed batches evaluate one fixed trained policy per environment, not independent policy-training runs. We report mean episodic return R and its standard error (SE) over the collected episodes. To compare tasks with different return scales, we also report recovered gain and its SE, where the gain for method m is

$$G_m = 100\,\frac{R_m - R_{\mathrm{pert}}}{R_{\mathrm{clean}} - R_{\mathrm{pert}}}. \tag{9}$$

Thus, $G_m = 0$ matches Base-Perturbed and $G_m = 100$ restores the measured clean return. Cerebellar's gain uses separately rerun, seed-matched controls, as described with the calibration and evaluation details in Appendix C.

3.2 Stationary Faults

Table 1 compares one stationary fault in each environment. In these settings, the deployment transition kernel P_t does not change within an episode. SteerControl has the highest recovered-gain point estimate in ten of the twelve matched settings. Averaged across these twelve settings, its recovered gain is 65.1%, compared with 50.9% for Cerebellar, 9.2% for Online LMS, and 9.1% for CMAC. Cerebellar has a slightly higher point estimate in PandaReach and Pendulum; uncertainty is substantial in Humanoid. Across the 30 stationary fault settings reported in Appendix D, SteerControl has the highest recovered-gain point estimate in 24. Reacher and Pusher show notable recovered gains. For these environments, we apply steering directly to the final action, which provided better recovery than steering the policy mean.

To test sensitivity to fault magnitude, Figure 2 varies a constant actuator bias from 0.1 to 0.7 of the actuator half-range in Ant, HalfCheetah, Pusher, and Reacher. SteerControl improves on the faulted base policy at every plotted severity, but the fraction of lost return recovered falls at the largest bias in three of the four tasks. For example, at severity 0.7 the gains are 12%, 14%, and 16%

Table 1: Stationary faults, one per environment. Trq. bias = torque bias (an additive actuator offset in units of half the action range); Act. gain = actuator gain; Act. rot. = action rotation; Engn. pwr. = engine power. Entries are estimate$_{\mathrm{SE}}$. † marks a task-space residual baseline. The displayed Base-Clean and Base-Perturbed returns are from the original runs; Cerebellar gain uses separately rerun clean and faulted controls on the same held-out seeds.

Environment	Perturbation setting	Return		Recovered gain (%)			
		Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Ant	Act. gain $\times 0.4$	$5550.9_{130.8}$	$2806.4_{13.6}$	$+5.5_{0.7}$	$+5.8_{0.7}$	$+18.9_{0.9}$	$\mathbf{+37.1_{0.9}}$
FetchReach	Trq. bias $+0.7$	$-1.67_{0.04}$	$-26.9_{1.0}$	$+3.5_{5.5}^{\dagger}$	$+3.5_{5.5}^{\dagger}$	$+83.5_{4.0}^{\dagger}$	$\mathbf{+86.7_{4.0}}$
HalfCheetah	Trq. bias $+0.4$	$11030.6_{19.0}$	$5859.7_{37.8}$	$+2.1_{1.5}$	$+2.6_{1.4}$	$+9.2_{1.5}$	$\mathbf{+45.0_{1.0}}$
Hopper	Act. gain $\times 0.8$	$3060.9_{1.6}$	$1795.2_{41.6}$	$+6.6_{4.4}$	$+11.3_{4.2}$	$+33.6_{5.5}$	$\mathbf{+46.1_{3.3}}$
Humanoid	Knee gain $\times 0.25$	$5205.6_{0.4}$	$3763.2_{312.9}$	$-34.1_{29.7}$	$-21.5_{30.3}$	$+8.8_{32.3}$	$\mathbf{+53.8_{28.3}}$
LunarLander	Engn. pwr. $\times 0.8$	$281.5_{0.7}$	$154.6_{4.9}$	$+47.8_{5.3}^{\dagger}$	$+33.4_{5.5}^{\dagger}$	$+56.7_{5.2}^{\dagger}$	$\mathbf{+65.1_{5.1}}$
PandaReach	Trq. bias $+0.6$	$-1.69_{0.01}$	$-19.7_{0.5}$	$+26.7_{3.2}^{\dagger}$	$+26.7_{3.2}^{\dagger}$	$\mathbf{+87.7_{2.6}}^{\dagger}$	$+85.5_{2.6}$
Pendulum	Gravity $\times 1.2$	$-142.9_{6.5}$	$-305.6_{15.1}$	$+20.5_{11.9}$	$+31.3_{11.3}$	$\mathbf{+63.8_{10.5}}$	$+61.8_{10.8}$
Pusher	Trq. bias $+0.4$	$-24.7_{0.2}$	$-73.4_{0.5}$	$+15.6_{1.6}$	$+7.2_{1.6}$	$+55.9_{1.7}$	$\mathbf{+84.5_{1.4}}$
Reacher	Trq. bias $+0.4$	$-3.54_{0.05}$	$-18.5_{0.1}$	$+34.1_{1.0}$	$+34.0_{1.0}$	$+86.2_{1.0}$	$\mathbf{+93.5_{0.8}}$
Swimmer	Act. rot. 52°	$346.5_{0.2}$	$204.7_{0.5}$	$+2.5_{0.5}$	$+2.4_{0.5}$	$+55.8_{0.5}$	$\mathbf{+69.6_{0.4}}$
Walker2d	Gravity $\times 1.3$	$6476.2_{8.7}$	$4907.3_{32.6}$	$-21.1_{5.1}$	$-26.8_{5.4}$	$+50.7_{2.6}$	$\mathbf{+52.3_{2.3}}$
Average recovered gain				$+9.1$	$+9.2$	$+50.9$	$\mathbf{+65.1}$

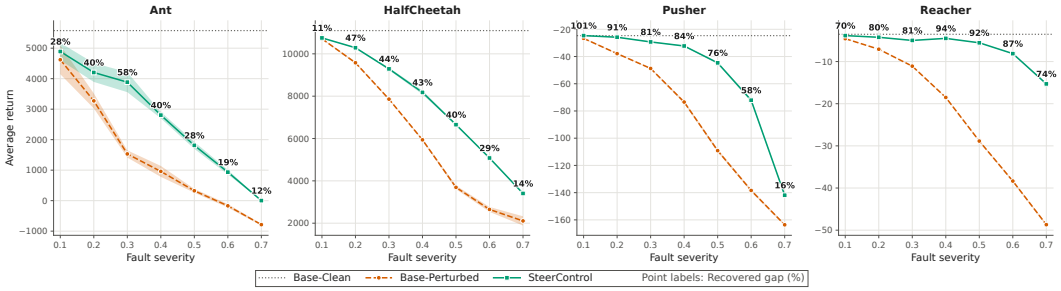


Figure 2: Mean episodic return under a constant additive actuator bias in four tasks. Severity is the bias as a fraction of the actuator half-range. Shaded bands show one SE across five matched seed-batch means from eight streams each. Labels above the solid curve give recovered gain.

in Ant, HalfCheetah, and Pusher, respectively. The sweep therefore shows recovery over multiple magnitudes while also making the remaining loss under severe faults visible. At larger perturbations, the local additive approximation may become less accurate, and the correction bounds that keep actions within their nominally covered range may limit the compensation needed for recovery.

3.3 Nonstationary Faults

We next compare SteerControl and the baselines when the deployment transition kernel P_t changes over time. Table 2 uses the same environments, fault types, and severities as Table 1. To make the nonstationary setting challenging, we alternate signed faults between equal and opposite values and one-sided faults between the damaged and nominal systems. Except for PandaReach, the switch interval is the number of control steps between square-wave switches, with equal time in each phase. FetchReach's switch clock continues across episode resets. Since the default low EMA weight can slow adaptation when faults change rapidly, we selected SteerControl's EMA weight from $\{0.2, 0.5, 0.8\}$ using calibration seeds for these experiments. Switch intervals were also selected using calibration seeds; PandaReach retains its stationary correction settings.

SteerControl has positive recovered-gain point estimates in all twelve matched switching settings and the largest estimate in nine. Averaged across these twelve settings, its recovered gain is 43.8%, compared with 30.1% for Cerebellar, 7.9% for CMAC, and 4.2% for Online LMS. The gains include 57.9% in Pusher, 51.0% in Reacher, and 55.7% in Swimmer despite repeated changes. Cerebellar has the largest estimate in FetchReach and Walker2d, and CMAC does so in LunarLander.

Table 2: Nonstationary faults use the same perturbation settings as Table 1. The fault switches on a square wave. Switch interval is the number of steps between consecutive switches (half the wave's period). PandaReach episodes average only 2.7 steps, leaving no room for a meaningful within-episode switch; its bias reverses at every episode boundary (1 ep.). Avg. ep. length is the average fault-free episode length in steps.

	Switch	Avg. ep.	Return		Recovered gain (%)			
Environment	interval	length	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Ant	125	973	$5550.9_{130.8}$	$4126.8_{70.2}$	$+8.3_{5.1}$	$-0.3_{7.5}$	$+7.7_{2.0}$	$\mathbf{+19.7_{9.4}}$
FetchReach	2	50	$-1.67_{0.04}$	$-12.0_{0.2}$	$-1.8_{3.1}{}^{\dagger}$	$-1.8_{3.1}{}^{\dagger}$	$\mathbf{+54.9_{3.5}}{}^{\dagger}$	$+53.1_{3.0}$
HalfCheetah	20	1000	$11030.6_{19.0}$	$6182.8_{57.0}$	$-2.4_{1.5}$	$-4.2_{1.6}$	$+1.8_{1.3}$	$\mathbf{+15.2_{1.5}}$
Hopper	250	1000	$3060.9_{1.6}$	$2610.8_{7.3}$	$-2.3_{2.8}$	$-0.0_{3.3}$	$-10.6_{10.4}$	$\mathbf{+16.9_{1.8}}$
Humanoid	125	1000	$5205.6_{0.2}$	$4153.9_{92.5}$	$+1.2_{12.5}$	$-12.2_{12.6}$	$-7.8_{13.3}$	$\mathbf{+41.4_{12.2}}$
LunarLander	100	179	$281.5_{0.7}$	$213.2_{4.7}$	$\mathbf{+51.5_{8.5}}$	$+43.7_{8.8}$	$+35.5_{9.0}{}^{\dagger}$	$+48.1_{8.9}$
PandaReach	1 ep.	2.7	$-1.69_{0.01}$	$-13.6_{0.4}$	$+19.8_{3.7}{}^{\dagger}$	$+16.4_{3.8}{}^{\dagger}$	$+80.3_{3.0}{}^{\dagger}$	$\mathbf{+83.4_{2.9}}$
Pendulum	50	200	$-142.9_{6.5}$	$-222.6_{8.5}$	$+0.9_{15.0}$	$+1.0_{15.0}$	$+62.3_{13.9}$	$\mathbf{+71.4_{14.0}}$
Pusher	12	100	$-24.7_{0.2}$	$-74.4_{0.2}$	$+1.2_{0.7}$	$+5.0_{0.7}$	$+54.7_{0.9}$	$\mathbf{+57.9_{0.6}}$
Reacher	10	50	$-3.54_{0.05}$	$-20.0_{0.1}$	$-0.6_{0.6}$	$+0.2_{0.6}$	$+29.3_{0.8}$	$\mathbf{+51.0_{0.5}}$
Swimmer	10	1000	$346.5_{0.2}$	$89.1_{2.8}$	$+5.8_{1.6}$	$+4.4_{1.4}$	$+36.5_{1.1}$	$\mathbf{+55.7_{1.1}}$
Walker2d	50	1000	$6476.2_{8.7}$	$5677.4_{15.5}$	$+12.6_{3.3}$	$-1.8_{4.5}$	$\mathbf{+16.4_{2.7}}$	$+11.9_{4.2}$
Average recovered gain					+7.9	+4.2	+30.1	**+43.8**

Because the correction estimate $\hat{c}_t$ is computed independently at each step, SteerControl can adapt to nonstationary perturbations. However, because the update is causal, $\hat{c}_t$ is available only after observing s_{t+1} and can first be applied at step $t+1$. This one-step delay requires dynamics shifts to be smooth between consecutive steps. Figure 3 examines recovery speed under different EMA weights after a single mid-episode sign reversal of actuator bias in Pusher and Reacher. Reward drops at the reversal, but SteerControl rapidly returns toward its pre-switch level, showing adaptation between two unseen faults within one episode. Larger EMA weights recover sooner, as expected; smaller weights retain more of the correction for the preceding bias. These trajectories illustrate the tracking-versus-smoothing trade-off in Equation 8.

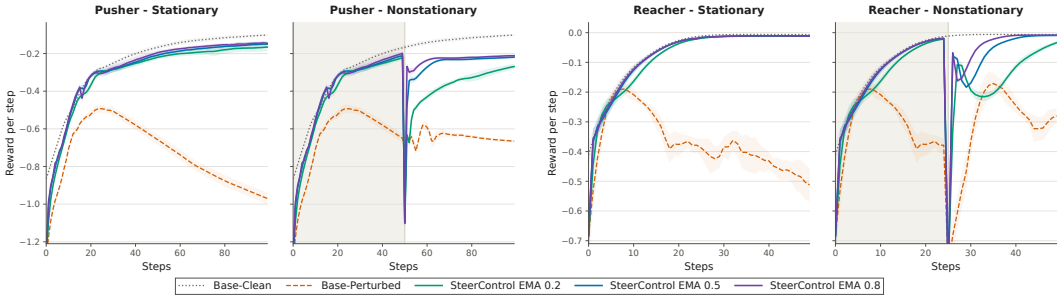


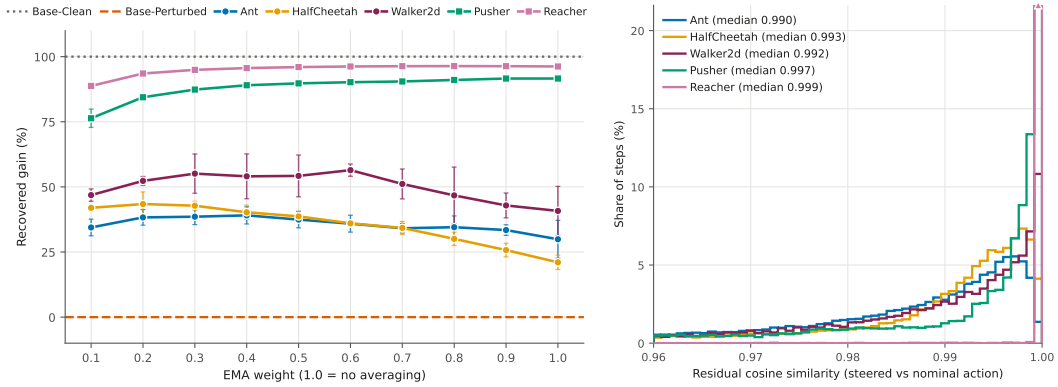
Figure 3: Mean reward per step under a constant actuator bias and a single sign reversal halfway through the episode, at step 50 of 100 in Pusher and step 25 of 50 in Reacher. Bands are pointwise 95% confidence intervals over ten matched seed-batch means.

3.4 Diagnostic Analysis

The optional moving average of corrections in Equation 8 is intended to reduce variation in residual-based correction estimates caused by dynamics-model error. Its effectiveness depends on the physical properties of the environment. In long-horizon environments with small changes between steps, lower EMA weights may reduce noise, whereas environments with rapid fluctuations can make lagged updates less effective. Figure 4a shows SteerControl's performance across five environments as the EMA weight varies from 0.1 to 1.0 in increments of 0.1 for stationary faults, using the same Base-Clean and Base-Perturbed controls at every weight. Each setting uses five matched seed batches of eight environment streams, yielding 40 episodes per weight in Ant, HalfCheetah, and Walker2d, and 400 in Pusher and Reacher. Although SteerControl remains effective across the

tested weights, larger weights improve recovery in Pusher and Reacher, whereas HalfCheetah loses recovered gain as α approaches one; Ant and Walker2d favor intermediate weights.

Figure 4b examines the local additive approximation behind the counterfactual target. At each state along a steered rollout, we execute the frozen policy's nominal action in a matched copy of the faulted simulator and compare its residual with that of the steered action. Across the five tasks, median residual cosine similarities range from 0.990 to 0.999. This directional agreement suggests that residual direction depends little on the executed action in these experiments.



(a) EMA weight sweep.

(b) Residual cosine similarity histogram.

Figure 4: Left: recovered gain as the EMA weight varies in five environments; a weight of one uses each estimate without averaging, and error bars are 95% confidence intervals across five seed-batch means. Right: histogram of cosine similarity between residuals from nominal and steered actions at matched faulted simulator states. Only similarities at least 0.96 are plotted, with each task normalized within that range.

4 Conclusion

We introduced SteerControl, which uses a frozen one-step dynamics model to locally guide a frozen model-free policy in response to unseen transition deviations during deployment. Both components are trained only under nominal dynamics, and online compensation requires no deployment rewards, reference trajectories, or updates to learned weights. Across twelve simulated continuous-control environments, SteerControl recovers a substantial share of the fault-induced return loss in many stationary and switching settings and achieves the highest recovered-gain point estimate among the evaluated residual controllers in most comparisons. Experiments with abrupt within-episode fault reversals further demonstrate fast recovery as action effects change during operation. Such recovery is valuable because deep RL policies often struggle to generalize to dynamics outside their training distribution (Packer et al., 2018). One interesting methodological aspect of SteerControl is that it keeps both the policy and dynamics model operating on inputs close to their training distributions, while their composition can handle out-of-distribution transition dynamics.

The correction mechanism is simple and effective, but its applicability depends on familiar state–action coverage, locally persistent discrepancies, and sufficient control authority. Recovery is particularly strong in several agent-centric perturbation settings and more limited for environment-based perturbations. Such limitations may reflect departures from nominal coverage, action-dependent discrepancies, or limits on feasible compensation; the present experiments do not isolate their relative contributions. Future work should examine these factors and extend compensation to dynamics changes that also shift the distribution of visited states. Broadening the dynamics model's coverage and accounting for its uncertainty when computing corrections are promising directions for maintaining useful guidance when the controller encounters states beyond its nominal training experience.

AI USE STATEMENT

Generative AI tools were used to assist with identifying relevant literature and related work, and to aid in polishing the writing of the manuscript. All references, summaries, and claims arising from AI-assisted literature discovery were independently verified by the authors against the original sources. The authors reviewed all AI-assisted content and take responsibility for the final content of the work.

REFERENCES

Tuo An, Jindou Jia, Gen Li, Jingliang Li, Chuhao Zhou, Pengfei Liu, Bofan Lyu, Jiaqi Bai, Xinying Guo, Geng Li, and Jianfei Yang. Feedback world model enables precise guidance of diffusion policy, 2026. URL https://arxiv.org/abs/2605.15705.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

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

Philip J. Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 619–629. PMLR, 2021. URL https://proceedings.mlr.press/v139/ball21a.html.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv preprint arXiv:1606.01540, 2016. URL https://arxiv.org/abs/1606.01540.

Wen-Hua Chen, Jun Yang, Lei Guo, and Shihua Li. Disturbance-observer-based control and related methods—an overview. *IEEE Transactions on Industrial Electronics*, 63(2):1083–1095, 2016. doi: 10.1109/TIE.2015.2478397. URL https://doi.org/10.1109/TIE.2015.2478397.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, and Tie-Yan Liu. An adaptive deep RL method for non-stationary environments with piecewise stable context. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e67a667df97d44fa43ffe8d179b44dd8-Abstract-Conference.html.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f859b94bda53149c7d7f5958c-Abstract.html.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289. PMLR, 2019. URL https://proceedings.mlr.press/v97/cobbe19a.html.

Maximilian Du and Shuran Song. DynaGuide: Steering diffusion polices with active dynamic guidance. In *Advances in Neural Information Processing Systems*, volume 38, pp. 44192–44221. Curran Associates, Inc., 2025. doi: 10.52202/085713-1473. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3ec6c6fc9065aa57785eb05dffe7c3db-Abstract-Conference.html.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021. URL https://arxiv.org/abs/2106.13687.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pašukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021. URL https://mlanthology.org/iclr/2021/hansen2021iclr-selfsupervised/.

James Harrison, Animesh Garg, Boris Ivanovic, Yuke Zhu, Silvio Savarese, Li Fei-Fei, and Marco Pavone. AdaPT: Zero-shot adaptive policy transfer for stochastic dynamical systems. In *Robotics Research*, volume 10 of *Springer Proceedings in Advanced Robotics*, pp. 437–453. Springer, 2020. doi: 10.1007/978-3-030-28619-4_34. URL https://doi.org/10.1007/978-3-030-28619-4_34.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Nethmi Jayasinghe, Diana Gontero, Spencer T. Brown, Vinod K. Sangwan, Mark C. Hersam, and Amit Ranjan Trivedi. Cerebellar-inspired residual control for fault recovery: From inference-time adaptation to structural consolidation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=3vNaSvdEAc.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation*, pp. 6023–6029. IEEE, 2019. doi: 10.1109/ICRA.2019.8794127. URL https://doi.org/10.1109/ICRA.2019.8794127.

Tor A. Johansen and Thor I. Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013. doi: 10.1016/j.automatica.2013.01.035. URL https://doi.org/10.1016/j.automatica.2013.01.035.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems XVII*, 2021a. doi: 10.15607/RSS.2021.XVII.011. URL https://doi.org/10.15607/RSS.2021.XVII.011.

Visak C. V. Kumar, Sehoon Ha, and C. Karen Liu. Error-aware policy learning: Zero-shot generalization in partially observable dynamic environments. In *Robotics: Science and Systems XVII*, 2021b. doi: 10.15607/RSS.2021.XVII.065. URL https://doi.org/10.15607/RSS.2021.XVII.065.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 2020. URL https://proceedings.mlr.press/v120/lambert20a.html.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5757–5766. PMLR, 2020. URL https://proceedings.mlr.press/v119/lee20g.html.

Daniel J. Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. In *International Conference on Learning Representations*, 2020. URL https://mlanthology.org/iclr/2020/mankowitz2020iclr-robust/.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://iclr.cc/virtual/2019/poster/1073.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning, 2018. URL https://arxiv.org/abs/1810.12282.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation*, pp. 3803–3810. IEEE, 2018. doi: 10.1109/ICRA.2018.8460528. URL https://doi.org/10.1109/ICRA.2018.8460528.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826. PMLR, 2017. URL https://proceedings.mlr.press/v70/pinto17a.html.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. arXiv preprint arXiv:1802.09464, 2018. URL https://arxiv.org/abs/1802.09464.

Roberta Raileanu, Max Goldstein, Arthur Szlam, and Rob Fergus. Fast adaptation to new environments via policy-dynamics value functions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7920–7931. PMLR, 2020. URL https://proceedings.mlr.press/v119/raileanu20a.html.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SyWvgP5el.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 348–358, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract.html.

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

Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block contextual MDPs for continual learning. In *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 608–623. PMLR, 2022. URL https://proceedings.mlr.press/v168/sodhani22a.html.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032, 2024. URL https://arxiv.org/abs/2407.17032.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28496–28510, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ef8446f35513a8d6aa2308357a268a7e-Abstract.html.

Shoukai Xu, Zihao Lian, Mingkui Tan, Liu Liu, Zhong Zhang, and Peilin Zhao. Test-time adapted reinforcement learning with action entropy regularization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70020–70031. PMLR, 2025. URL https://proceedings.mlr.press/v267/xu25ar.html.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems XIII*, 2017. doi: 10.15607/RSS.2017.XIII.048. URL https://doi.org/10.15607/RSS.2017.XIII.048.

Xilun Zhang, Shiqi Liu, Peide Huang, William Jongwon Han, Yiqi Lyu, Mengdi Xu, and Ding Zhao. Dynamics as prompts: In-context learning for sim-to-real system identifications. *IEEE Robotics and Automation Letters*, 10(4):3190–3197, 2025. doi: 10.1109/LRA.2025.3540391. URL https://doi.org/10.1109/LRA.2025.3540391.

A Related Work

Prior evaluations of reinforcement-learning generalization differ in the type of variation held out during training. CoinRun uses unseen procedurally generated levels and shows substantial overfitting to a fixed set of training instances, with broader level diversity reducing the train–test gap (Cobbe et al., 2019). Packer et al. instead vary physical parameters and separate tests within the randomized training range from tests on disjoint, more extreme ranges (Packer et al., 2018). They find extrapolation to the latter ranges generally more difficult, with no consistent advantage for the specialized robust or adaptive methods. Together, these benchmarks motivate treating dynamics generalization separately from changes in layout or observation. We compare related methods along three dimensions. We ask whether they train on varied dynamics, what information they use at deployment, and whether and how they adapt online, for example by updating policy or model weights, inferring a context, or applying a temporary action correction.

Training-time approaches either optimize a policy for robustness or learn how to adapt from later experience. Dynamics randomization, robust optimization, and adversarial training incorporate anticipated variation before deployment (Peng et al., 2018; Rajeswaran et al., 2017; Pinto et al., 2017; Mankowitz et al., 2020). This robustness can come at the cost of nominal performance. Other methods learn to adjust behavior using information inferred from recent experience. UP-OSI conditions its policy on estimated physical parameters, whereas RMA and CaDM infer latent context (Yu et al., 2017; Kumar et al., 2021a; Lee et al., 2020). Meta-RL takes a broader approach by training agents across task distributions to adapt from limited experience (Duan et al., 2016; Finn et al., 2017; Rakelly et al., 2019). Related methods select or condition behavior using transition-derived signals (Raileanu et al., 2020; Kumar et al., 2021b; Ball et al., 2021). These mechanisms can adapt rapidly at deployment, but their responses are learned in advance and remain tied to the training distribution.

A separate line of work adapts using interaction data collected in the current environment. Model-based controllers such as meta-adaptive control and PETS adapt learned dynamics models from recent transitions and plan with a specified reward or cost (Nagabandi et al., 2019; Chua et al., 2018). Reward-free test-time methods such as PAD and TARL instead update selected policy components without task rewards (Hansen et al., 2021; Xu et al., 2025). For changes within an episode, ZeUS and SeCBAD infer the current latent regime from recent experience; their inference systems and policies are trained across changing regimes (Sodhani et al., 2022; Chen et al., 2022). Continual RL addresses persistent learning across longer task streams, where retention, transfer, and forgetting are central concerns (Rolnick et al., 2019; Wołczyk et al., 2021). Overall, these methods use online interaction data either to update learned components or to infer context for an adaptation mechanism learned in advance across varied conditions.

Another adaptation paradigm retains a base controller and adds a corrective mechanism. Disturbance-observer control estimates aggregate model error or external disturbance and adds a compensating input, while control allocation maps desired effects to feasible actuator commands under actuator constraints or failures (Chen et al., 2016; Johansen & Fossen, 2013). These formulations generally assume an explicit model of the controlled system, disturbance channel, or actuator effectiveness. Residual RL also composes a correction with a fixed nominal command, but obtains that correction by reward-based policy learning on the target task (Johannink et al., 2019). AdaPT retains a source policy and computes target commands through tube MPC, using a source-simulator reference trajectory and bounds on source–target mismatch (Harrison et al., 2020). CAPTURE keeps its learned networks fixed but estimates simulator parameters from differences between real and simulated trajectories. Its task policy is trained over the same parameter family (Zhang et al., 2025).

DynaGuide differentiates through a learned long-horizon dynamics model to steer a frozen diffusion policy toward externally supplied images of desired outcomes or away from supplied images of undesired outcomes (Du & Song, 2025). Another related approach feeds online prediction errors back into subsequent latent predictions and uses the corrected predictions to guide a frozen diffusion policy toward latent states from expert demonstrations (An et al., 2026). Both use auxiliary predictive models for inference-time action guidance, but DynaGuide is guided by the supplied outcome images, whereas the second method relies on expert references and focuses on test-time shifts caused by changes in the initial state rather than transition dynamics. Cerebellar-inspired residual control also adds bounded residual actions to a frozen SAC policy for fault recovery, but updates

residual-controller weights from tracking error against a phase-aligned nominal trajectory and uses reward or progress feedback to regulate adaptation (Jayasinghe et al., 2026).

Within this literature, SteerControl is closest to auxiliary correction methods because it retains a nominal controller and computes a temporary correction online. It differs from disturbance-observer control and control allocation because it does not require an explicit model of the controlled system, disturbance channel, or actuator effectiveness. Unlike residual RL, it does not learn the correction from target-task rewards. Unlike adaptive model-based control and AdaPT, it does not replan using a specified reward or cost. It also avoids explicit physical-parameter identification, deployment-time weight updates, and adaptation mechanisms trained across varied tasks or dynamics. SteerControl instead uses nominal-only, fixed learned components to compensate for previously unseen transition-dynamics changes without a target outcome or reference trajectory: discrepancies between predictions from its one-step dynamics model and observed transitions guide temporary corrections to the model-free policy.

B Training Details

Figure 5 shows the twelve environments used in the evaluation. Table 3 gives the policy and dynamics-model architectures, training budgets, and data-collection settings. Figure 6 reports return during nominal policy training; the number of environment steps differs by task. Figure 7 reports training and validation error for standardized one-step state changes. We use validation error on three of the sixteen nominal data-collection streams to select the training epoch. The validation set contains no faulted transitions.

Table 3: Nominal policy and world-model training configurations. Ranges and alternatives indicate environment-specific choices.

Category	Parameter	Value
SAC policy	Actor and critic	Two hidden layers of 256 units each
	Training budget	60,000–6,000,000 environment steps, by environment
	Learning starts	1,000: FetchReach, PandaReach, Pendulum; 5,000: Reacher; 10,000: Ant, HalfCheetah, Hopper, Humanoid, LunarLander, Pusher, Swimmer, Walker2d
	Learning rate	3×10^{-4} generally; 10^{-3} for FetchReach, PandaReach, Pendulum; 7.3×10^{-4} for LunarLander
	Batch size	256
	Discount γ	0.99 generally; 0.95 for FetchReach/PandaReach, 0.98 for Pendulum, 0.9999 for Swimmer
	Target update τ	0.005 generally; 0.05 for FetchReach/PandaReach, 0.02 for Pendulum, 0.01 for LunarLander
World model	Ensemble and network	Five MLPs; three 256-unit SiLU hidden layers per member
	Prediction target	One-step observation change, $\Delta s = s_{t+1} - s_t$
	Clean data collection	16 environment streams; 1,000–24,000 steps per stream, by environment
	Action mix	50% frozen-policy actions and 50% uniform random actions
	Scaling	Normalize observations and actions; standardize Δs
	Optimizer	Adam; learning rate 10^{-3}; batch size 256
	Epoch budget	100–200, by environment
	Model selection	Hold out 3 of 16 environment streams; retain best validation epoch

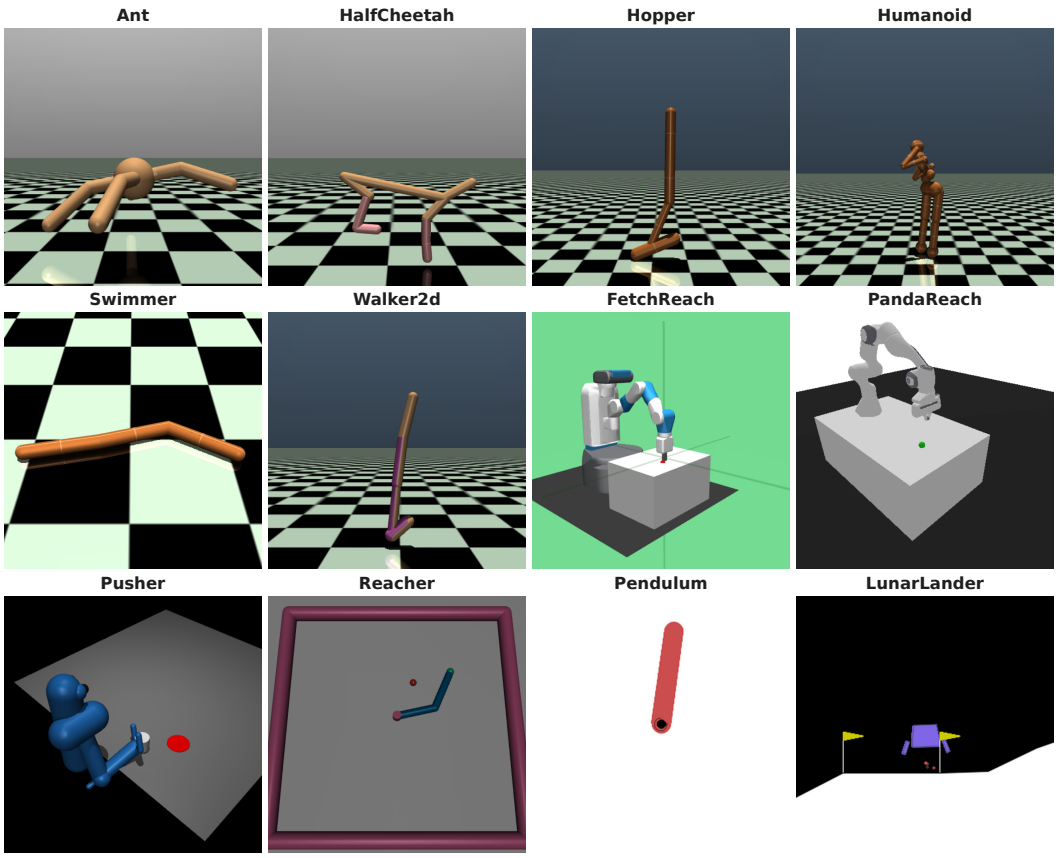


Figure 5: Snapshots of the twelve continuous-control environments used in the experiments.

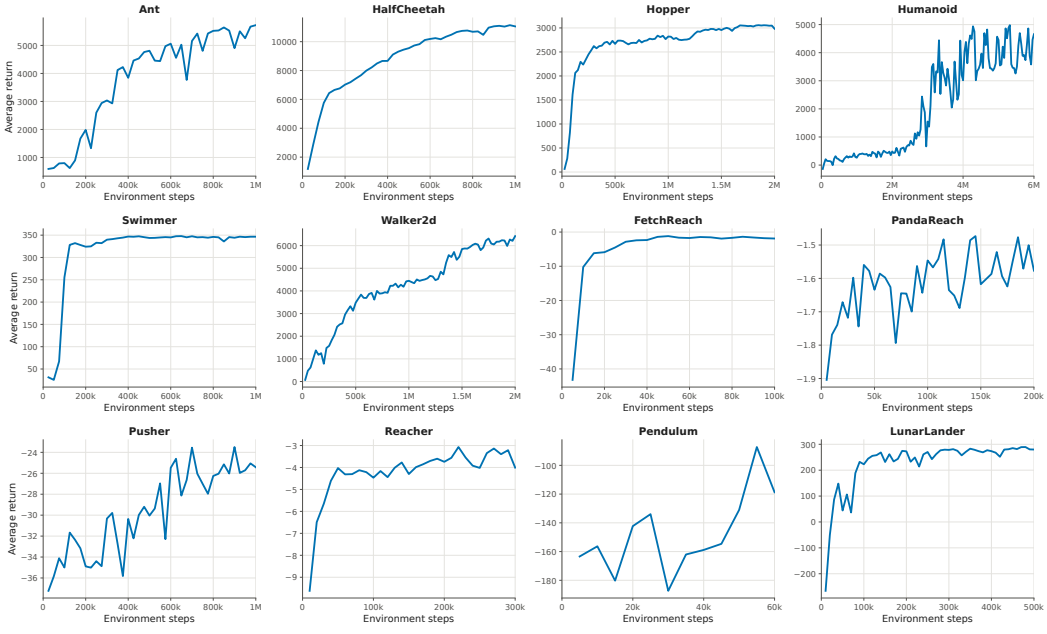


Figure 6: Average return during nominal SAC policy training as a function of environment steps. Axes are scaled separately for each task.

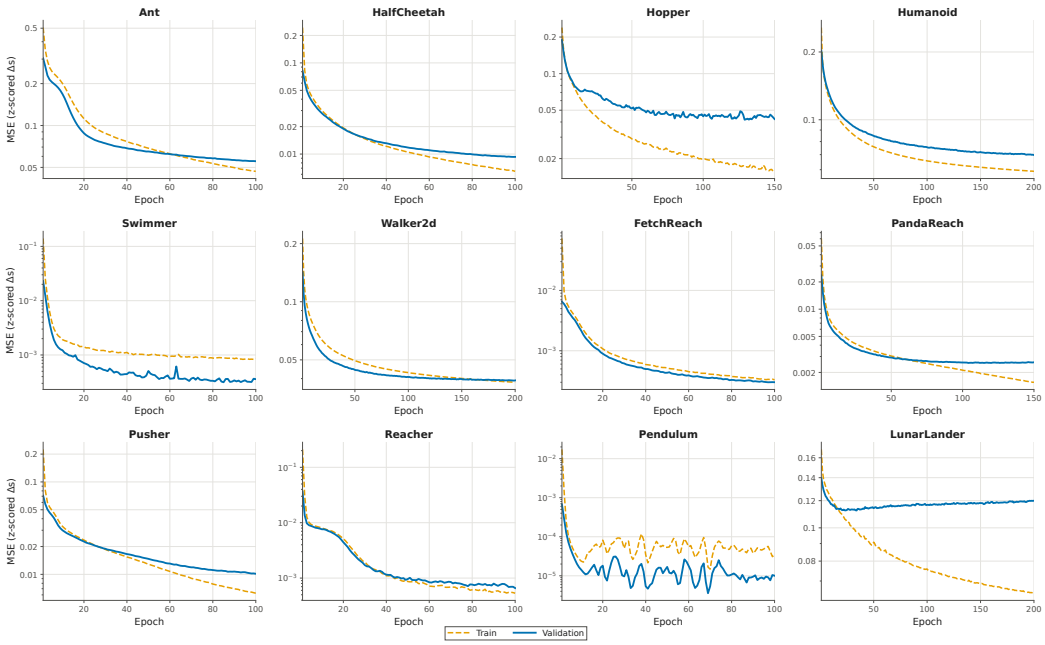


Figure 7: Training of the dynamics model. Curves show the training and validation mean squared error for one-step predictions of standardized state changes, averaged over the five ensemble members. The horizontal axis gives training epochs, and each task has its own vertical scale.

C Calibration and Evaluation Details

Table 4 lists SteerControl's default correction parameters and calibrated ranges. Online LMS and CMAC each use one fixed configuration across settings. Cerebellar's setting-specific search varies learning rate, correction gain and clipping, reference lookup, direction and authority gates, and feature traces. It also tests whether to retain corrective memory across episodes and, for FetchReach and PandaReach, whether to use goal-directed error instead of a stored clean trajectory.

Shorter tasks produce multiple complete episodes per environment stream. In switching conditions, all methods use the same fault schedule within a held-out seed batch. We rerun Base-Clean and Base-Perturbed on Cerebellar's held-out evaluation seeds and use these arms for its recovered gain and paired comparisons. The summary tables display the original Base-Clean and Base-Perturbed runs, so the gain computed from those printed means need not match Cerebellar's reported gain.

Table 4: SteerControl correction configuration details.

Parameter	Value
Injection site	Action, pre-squash mean μ (default), or first hidden layer h_1
Inner optimizer	Five normalized-gradient descent steps per update
Inner step size	$[0.1,\ 0.4]$; default 0.2
EMA weight α	$[0.1,\ 1]$; default 0.2

D Detailed Stationary Results

Tables 5–16 give episodic return and recovered gain for all 30 retained stationary fault settings. Entries show estimate$_{\mathrm{SE}}$; recovered gain is expressed as a fraction in these tables. Boldface marks the highest displayed point estimate among the four adaptive methods for each metric. Cerebellar gain uses its separately rerun matched controls, and † denotes task-space tracking in a residual-control baseline.

Table 5: Reported stationary settings for Ant. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Actuator gain $\times 0.4$						
Return	$5550.9_{130.8}$	$2806.4_{13.6}$	$2957.5_{13.6}$	$2965.9_{13.1}$	$3338.3_{20.4}$	$\mathbf{3823.3_{21.8}}$
Recovered gain	1	0	$+0.055_{0.007}$	$+0.058_{0.007}$	$+0.189_{0.009}$	$\mathbf{+0.371_{0.009}}$
Ground friction $\times 0.05$						
Return	$5550.9_{130.8}$	$4386.8_{74.5}$	$4457.9_{14.9}$	$4478.4_{20.2}$	$4481.1_{14.1}$	$\mathbf{4574.3_{16.4}}$
Recovered gain	1	0	$+0.061_{0.065}$	$+0.079_{0.066}$	$0.082_{0.018}$	$\mathbf{+0.161_{0.066}}$
One-sided actuator gain $\times 0.25$						
Return	$5550.9_{130.8}$	$2561.1_{14.5}$	$2646.5_{16.0}$	$2616.4_{17.4}$	$2605.7_{55.8}$	$\mathbf{2897.4_{7.8}}$
Recovered gain	1	0	$+0.029_{0.007}$	$+0.019_{0.008}$	$0.015_{0.019}$	$\mathbf{+0.113_{0.006}}$

Table 6: Reported stationary settings for FetchReach. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC†	Online LMS†	Cerebellar†	SteerControl
Torque bias $+0.7$						
Return	$-1.67_{0.04}$	$-26.9_{1.0}$	$-26.0_{1.0}$	$-26.0_{1.0}$	$-5.81_{0.16}$	$\mathbf{-5.02_{0.14}}$
Recovered gain	1	0	$+0.035_{0.055}$	$+0.035_{0.055}$	$+0.835_{0.040}$	$\mathbf{+0.867_{0.040}}$
Action rotation 60°						
Return	$-1.67_{0.04}$	$-12.8_{0.2}$	$-13.1_{0.2}$	$-13.5_{0.2}$	$-12.8_{0.2}$	$\mathbf{-8.33_{0.27}}$
Recovered gain	1	0	$-0.027_{0.031}$	$-0.064_{0.032}$	$+0.000_{0.031}$	$\mathbf{+0.400_{0.033}}$

Table 7: Reported stationary settings for HalfCheetah. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Torque bias $+0.4$						
Return	$11030.6_{19.0}$	$5859.7_{37.8}$	$5970.7_{69.6}$	$5992.0_{64.5}$	$6339.6_{67.8}$	$\mathbf{8184.5_{33.6}}$
Recovered gain	1	0	$+0.021_{0.015}$	$+0.026_{0.014}$	$+0.092_{0.015}$	$\mathbf{+0.450_{0.010}}$
Joint damping $\times 1.5$						
Return	$11030.6_{19.0}$	$6209.9_{11.9}$	$6526.6_{22.5}$	$6448.9_{24.4}$	$\mathbf{7029.4_{24.3}}$	$6779.8_{40.0}$
Recovered gain	1	0	$+0.066_{0.005}$	$+0.050_{0.006}$	$\mathbf{+0.170_{0.005}}$	$+0.118_{0.009}$

Table 8: Reported stationary settings for Hopper. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Actuator gain $\times 0.8$						
Return	$3060.9_{1.6}$	$1795.2_{41.6}$	$1878.9_{36.5}$	$1938.7_{32.7}$	$2136.4_{48.0}$	$\mathbf{2379.3_{6.4}}$
Recovered gain	1	0	$+0.066_{0.044}$	$+0.113_{0.042}$	$+0.336_{0.055}$	$\mathbf{+0.461_{0.033}}$
Thigh actuator gain $\times 0.15$						
Return	$3060.9_{1.6}$	$604.0_{90.6}$	$643.0_{94.2}$	$605.3_{94.3}$	$916.0_{100.9}$	$\mathbf{1173.3_{84.8}}$
Recovered gain	1	0	$+0.016_{0.053}$	$+0.001_{0.053}$	$+0.162_{0.051}$	$\mathbf{+0.232_{0.050}}$
Horizontal body force $0.1\,g$						
Return	$3060.9_{1.6}$	$2705.2_{9.5}$	$2615.0_{58.6}$	$2611.3_{56.8}$	$2762.8_{2.5}$	$\mathbf{2767.6_{1.2}}$
Recovered gain	1	0	$-0.254_{0.167}$	$-0.264_{0.162}$	$\mathbf{+0.193_{0.056}}$	$+0.175_{0.027}$
One-sided actuator gain $\times 0.6$						
Return	$3060.9_{1.6}$	$849.0_{48.4}$	$955.6_{57.2}$	$919.1_{51.1}$	$\mathbf{1108.6_{38.6}}$	$1087.5_{60.4}$
Recovered gain	1	0	$+0.048_{0.034}$	$+0.032_{0.032}$	$\mathbf{+0.117_{0.030}}$	$+0.108_{0.035}$

Table 9: Reported stationary settings for Humanoid. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Right-knee actuator gain $\times 0.25$						
Return	$5205.6_{0.4}$	$3763.2_{312.9}$	$3271.6_{292.9}$	$3453.1_{304.7}$	$4034.6_{286.3}$	$\mathbf{4538.9_{262.2}}$
Recovered gain	1	0	$-0.341_{0.297}$	$-0.215_{0.303}$	$+0.088_{0.323}$	$\mathbf{+0.538_{0.283}}$

Table 10: Reported stationary settings for LunarLander. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC†	Online LMS†	Cerebellar†	SteerControl
Engine power $\times 0.8$						
Return	$281.5_{0.7}$	$154.6_{4.9}$	$215.3_{4.6}$	$197.1_{5.0}$	$226.8_{4.5}$	$\mathbf{237.2_{4.2}}$
Recovered gain	1	0	$+0.478_{0.053}$	$+0.334_{0.055}$	$+0.567_{0.052}$	$\mathbf{+0.651_{0.051}}$
Actuator gain $\times 0.6$						
Return	$281.5_{0.7}$	$182.0_{5.1}$	$201.5_{5.2}$	$200.8_{5.0}$	$215.0_{5.0}$	$\mathbf{219.3_{4.8}}$
Recovered gain	1	0	$+0.196_{0.073}$	$+0.189_{0.072}$	$+0.326_{0.073}$	$\mathbf{+0.375_{0.071}}$

Table 11: Reported stationary settings for PandaReach. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC†	Online LMS†	Cerebellar†	SteerControl
Torque bias $+0.6$						
Return	$-1.69_{0.01}$	$-19.7_{0.5}$	$-14.8_{0.3}$	$-14.9_{0.3}$	$\mathbf{-4.03_{0.03}}$	$-4.30_{0.03}$
Recovered gain	1	0	$+0.267_{0.032}$	$+0.267_{0.032}$	$\mathbf{+0.877_{0.026}}$	$+0.855_{0.026}$
Action rotation 75°						
Return	$-1.68_{0.01}$	$-23.3_{0.4}$	$-23.0_{0.4}$	$-23.0_{0.4}$	$-12.37_{0.05}$	$\mathbf{-11.1_{0.1}}$
Recovered gain	1	0	$+0.014_{0.026}$	$+0.014_{0.025}$	$+0.506_{0.018}$	$\mathbf{+0.566_{0.019}}$

Table 12: Reported stationary settings for Pendulum. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Torque bias $+0.8$						
Return	$-142.9_{6.5}$	$-607.2_{6.4}$	$-601.5_{6.5}$	$-602.4_{6.5}$	$-391.1_{8.1}$	$\mathbf{-225.5_{9.7}}$
Recovered gain	1	0	$+0.012_{0.020}$	$+0.010_{0.020}$	$+0.465_{0.022}$	$\mathbf{+0.822_{0.025}}$
Gravity $\times 1.2$						
Return	$-142.9_{6.5}$	$-305.6_{15.1}$	$-272.1_{12.1}$	$-254.6_{10.6}$	$\mathbf{-201.8_{8.1}}$	$-205.1_{9.1}$
Recovered gain	1	0	$+0.205_{0.119}$	$+0.313_{0.113}$	$\mathbf{+0.638_{0.105}}$	$+0.618_{0.108}$

Table 13: Reported stationary settings for Pusher. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Torque bias $+0.4$						
Return	$-24.7_{0.2}$	$-73.4_{0.5}$	$-65.8_{0.6}$	$-69.9_{0.6}$	$-46.2_{0.6}$	$\mathbf{-32.2_{0.4}}$
Recovered gain	1	0	$+0.156_{0.016}$	$+0.072_{0.016}$	$+0.559_{0.017}$	$\mathbf{+0.845_{0.014}}$
Fingertip payload 12.6 kg						
Return	$-24.7_{0.2}$	$-58.8_{0.2}$	$-59.7_{0.2}$	$-59.7_{0.2}$	$-58.7_{0.3}$	$\mathbf{-52.5_{0.2}}$
Recovered gain	1	0	$-0.027_{0.009}$	$-0.026_{0.009}$	$+0.002_{0.010}$	$\mathbf{+0.186_{0.009}}$
Viscous medium, viscosity 25						
Return	$-24.7_{0.2}$	$-74.3_{0.3}$	$-78.9_{0.4}$	$-78.9_{0.4}$	$-70.7_{0.4}$	$\mathbf{-66.1_{0.3}}$
Recovered gain	1	0	$-0.093_{0.010}$	$-0.093_{0.010}$	$+0.072_{0.010}$	$\mathbf{+0.165_{0.010}}$
Joint damping $\times 9$						
Return	$-24.7_{0.2}$	$-57.8_{0.5}$	$-58.4_{0.5}$	$-58.3_{0.5}$	$-57.7_{0.5}$	$\mathbf{-52.5_{0.3}}$
Recovered gain	1	0	$-0.019_{0.020}$	$-0.017_{0.020}$	$+0.002_{0.020}$	$\mathbf{+0.159_{0.017}}$
Shoulder-pan actuator reversed, $\times -0.15$						
Return	$-24.7_{0.2}$	$-42.1_{0.3}$	$-42.2_{0.3}$	$-42.2_{0.3}$	$-41.9_{0.3}$	$\mathbf{-39.4_{0.3}}$
Recovered gain	1	0	$-0.010_{0.023}$	$-0.008_{0.023}$	$+0.009_{0.023}$	$\mathbf{+0.156_{0.023}}$

Table 14: Reported stationary settings for Reacher. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Torque bias $+0.4$						
Return	$-3.54_{0.05}$	$-18.5_{0.1}$	$-13.4_{0.1}$	$-13.4_{0.1}$	$-5.61_{0.11}$	$\mathbf{-4.51_{0.05}}$
Recovered gain	1	0	$+0.341_{0.010}$	$+0.340_{0.010}$	$+0.862_{0.010}$	$\mathbf{+0.935_{0.008}}$

Table 15: Reported stationary settings for Swimmer. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Action rotation 52°						
Return	$346.5_{0.2}$	$204.7_{0.5}$	$208.3_{0.5}$	$208.0_{0.5}$	$283.8_{0.5}$	$\mathbf{303.5_{0.2}}$
Recovered gain	1	0	$+0.025_{0.005}$	$+0.024_{0.005}$	$+0.558_{0.005}$	$\mathbf{+0.696_{0.004}}$
Body mass $\times 2.6$						
Return	$346.5_{0.2}$	$134.0_{0.8}$	$146.2_{0.9}$	$142.7_{0.9}$	$184.8_{1.1}$	$\mathbf{210.1_{1.2}}$
Recovered gain	1	0	$+0.057_{0.006}$	$+0.041_{0.006}$	$+0.242_{0.006}$	$\mathbf{+0.358_{0.007}}$
Torque bias $+0.8$						
Return	$346.5_{0.2}$	$138.0_{0.5}$	$131.4_{0.7}$	$137.5_{0.6}$	$\mathbf{187.6_{0.5}}$	$182.2_{0.4}$
Recovered gain	1	0	$-0.031_{0.004}$	$-0.002_{0.004}$	$\mathbf{+0.238_{0.004}}$	$+0.212_{0.003}$

Table 16: Reported stationary settings for Walker2d. Entries are estimate$_{\mathrm{SE}}$. Cerebellar uses separately rerun clean and faulted controls for its recovered gain.

Metric	Base-Clean	Base-Perturbed	CMAC	Online LMS	Cerebellar	SteerControl
Gravity $\times 1.3$						
Return	$6476.2_{8.7}$	$4907.3_{32.6}$	$4575.6_{72.6}$	$4486.1_{78.9}$	$5703.0_{23.5}$	$\mathbf{5727.9_{16.6}}$
Recovered gain	1	0	$-0.211_{0.051}$	$-0.268_{0.054}$	$+0.507_{0.026}$	$\mathbf{+0.523_{0.023}}$
Joint damping $\times 5$						
Return	$6477.2_{7.5}$	$5495.3_{8.2}$	$5521.1_{16.0}$	$5425.8_{16.1}$	$5609.7_{9.5}$	$\mathbf{5840.4_{25.6}}$
Recovered gain	1	0	$+0.026_{0.018}$	$-0.071_{0.018}$	$+0.111_{0.013}$	$\mathbf{+0.351_{0.027}}$
Body mass $\times 1.3$						
Return	$6476.2_{8.7}$	$4699.4_{11.7}$	$4822.4_{26.2}$	$4749.1_{24.4}$	$4939.8_{19.4}$	$\mathbf{5097.1_{28.0}}$
Recovered gain	1	0	$+0.069_{0.016}$	$+0.028_{0.015}$	$+0.135_{0.013}$	$\mathbf{+0.224_{0.017}}$