LEARNING GENERATIVE MOTION MANIFOLDS FOR WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Parameterized Latent Motions for Model-Based Reinforcement Learning (PaLMR), a framework that learns generative motion manifolds as reusable action interfaces for world models. From task-independent motion experience, PaLMR learns behavior-aligned latent coordinates and a context-conditioned flow model for executing and interpolating motions. The learned interface remains fixed during model-based RL, providing shared coordinates for predicting motion outcomes and composing motion sequences. We establish conditions for predictive generalization and show how denser motion coverage tightens planning-error bounds, supported by controlled experiments demonstrating more accurate prediction and effective use of interpolated motions. Across navigation and robotic manipulation benchmarks, PaLMR outperforms primitive-action and generative-policy RL baselines in overall task performance and online sample efficiency.

1 INTRODUCTION

World models support model-based reinforcement learning by predicting the consequences of actions and evaluating future behavior (Hansen et al., 2024). Their effectiveness depends on how the predicted state and the actions driving its evolution are represented. Skills and action chunks make temporally extended behavior a unit of decision rather than individual motor commands (Co-Reyes et al., 2018; Pertsch et al., 2021; Shi et al., 2023; Li et al., 2025). A latent action space provides coordinates for these behaviors, serving both as an input to the world model and as the domain of planning. Its organization therefore shapes two connected problems: learning how requested behaviors affect the environment, and searching for behavior sequences that achieve a task.

For this action space to support flexible decisions, it should offer executable motions between observed behaviors. In Figure 1 (top), skills extracted from task demonstrations cover much of a new route, but the illustrated repertoire lacks the final diagonal motion needed to reach the goal. Continuous coordinates make intermediate requests expressible; meaningful interpolation must also make them executable while preserving the recorded motions. A continuous motion family organized by physical behavior can therefore serve two purposes: providing structured neighborhoods for predictive generalization and additional motion choices for planning.

Building such a space requires both motion experience and a way to organize it. Trajectory abstraction and reward-free exploration offer complementary sources of reusable behavior (Ajay et al., 2021; Eysenbach et al., 2019; Sharma et al., 2020). Geometric structure can organize the resulting coordinates by physical behavior (Park et al., 2022; 2024), complementing reconstruction of individual recorded segments. Execution introduces a further requirement: a moving robot and a stationary robot can need different controls to perform the same local motion. A behavior-grounded representation should therefore separate the motion request from its execution context. Learning this representation and its conditional executor from task-independent local experience establishes an interface that different tasks can reuse while learning their own predictive models and control strategies.

We introduce *PaLMR* (**Pa**rameterized **L**atent **M**otions for Model-Based **R**L), which extends PDP-style behavior parameterization (Zhang et al., 2026) into a generative motion manifold for learning and planning with world models. PaLMR learns geometry-informed behavior codes from task-independent local experience, then trains its *Parameterized Flow Matching* (PFM) module to generate action chunks conditioned on the frozen encoder's codes and the current execution context. The encoder and generator then remain frozen as each downstream task learns a world model of their

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.

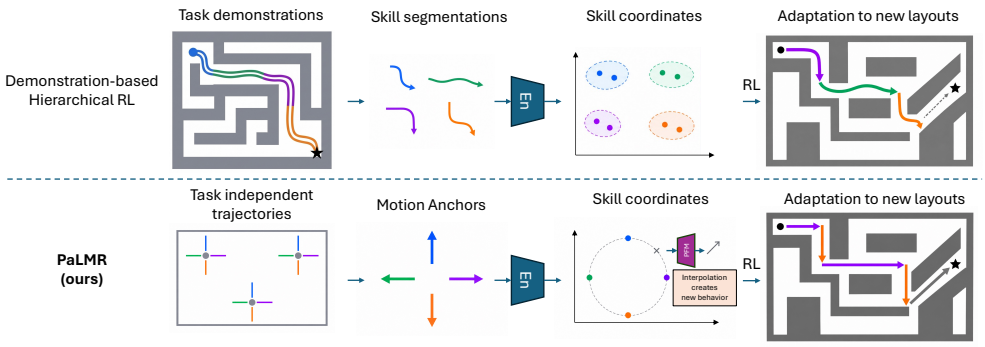

Figure 1: **An executable motion space for world-model learning and planning. Top:** Skills extracted from task demonstrations cover part of a new route but lack the final diagonal motion. **Bottom:** Task-independent motion experience grounds a generative manifold whose intermediate queries provide additional executable motions. These coordinates serve as action inputs to task-specific world models and as variables for selecting and composing motions. Colors associate motions and codes within each row; geometry and paths are schematic.

transition and reward consequences and a high-level controller composes their motions. This separates reusable motion execution from task-specific prediction and decision-making. Figure 1 (bottom) illustrates how the resulting manifold supplies intermediate motions for the controller to select and compose.

Our contributions are:

- We introduce PaLMR, which pretrains behavior-grounded motion coordinates and their conditional executor separately from task dynamics, then exposes the frozen interface to task-specific world-model learning and planning.
- We derive bounds on predictive generalization, motion-sequence approximation and policy error, connecting execution regularity and motion coverage to prediction and planning under stated assumptions.
- Controlled studies show that geometry-aligned coordinates improve motion prediction, interpolated requests benefit planning with fixed models, and PFM supports motion interpolation and adaptation to changed initial velocities.
- Across all fifty Meta-World tasks, one shared frozen motion interface achieves the highest task-balanced mean success in the common-budget comparison: 79.67%, compared with 67.87% for flat TD-MPC2.

2 Related Work

Skill learning and model-based control. Movement primitives and action chunking provide temporally extended units of control (Ijspeert et al., 2002; Li et al., 2025). SeCTAR, SPiRL and OPAL learn trajectory abstractions or skill priors for hierarchical control (Co-Reyes et al., 2018; Pertsch et al., 2021; Ajay et al., 2021). SkiMo jointly pretrains skills and predictive skill dynamics for downstream planning (Shi et al., 2023).

Behavior discovery and geometric representations. DIAYN and DADS acquire reusable behaviors without downstream task rewards (Eysenbach et al., 2019; Sharma et al., 2020). LSD and METRA study the geometric organization of behaviors in latent skill spaces (Park et al., 2022; 2024). These approaches connect reward-free behavior acquisition with behavior-grounded coordinates for downstream control.

Parameterized generative policies. Diffusion Policy and flow matching provide expressive conditional generators (Chi et al., 2023; Lipman et al., 2023). PDP grounds behavior coordinates in trajectory similarity for context-conditioned interpolation and demonstration-guided latent fitting (Zhang

et al., 2026). DSRL and LPS steer pretrained policies through latent variables (Wagenmaker et al., 2025; Im et al., 2026), while DPPO, ReinFlow and FPO++ update generative policies using task reward (Ren et al., 2025; Zhang et al., 2025; Yi et al., 2026). PaLMR pretrains a task-independent motion interface and freezes it before learning task-specific world models and controllers, separating motion execution from SkiMo's joint skill–dynamics pretraining and PDP's adaptation of a pretrained task policy.

3 BACKGROUND

Motion requests and execution. Consider a controlled Markov process with complete state s, task observation $o = O(s)$ and primitive motion action $a^{\mathrm{mot}} \in \mathbb{R}^{d_a}$. A recorded segment contains representative-point positions $p_{0:H}$ and motion actions $a^{\mathrm{mot}}_{0:H-1}$; its pre-action relative trace is $b = (p_k - p_0)_{k=0}^{H-1}$. A behavior request z lies in a compact query region $\mathcal{Z} \subseteq \mathcal{S}^{d_z-1}$ with chord distance $\rho(z, z') = \|z - z'\|$. Given execution context $c = c(s)$ and Gaussian noise $\xi \sim \mathcal{N}(0, I_{Hd_a})$, a decoder $G_\theta(c, z, \xi)$ generates an H-step physical motion chunk. The controller executes $K \le H$ actions, optionally assembled with an auxiliary command η, before requesting another chunk. We use T for the number of high-level decisions in a plan and $\beta \in [0, 1)$ for the per-decision discount. Under a fixed execution and reward-aggregation convention, a frozen decoder induces the transition kernel $P_G(\cdot \mid s, z)$ and expected decision-level reward $R_G(s, z)$, integrating generation and environment randomness.

Conditional flow matching. For a normalized action-chunk target y, flow matching (Lipman et al., 2023) learns a vector field along $x_t = (1 - t)\xi + ty$, with independent $t \sim \mathcal{U}[0, 1]$ and Gaussian noise independent of the data:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\Big[\|v_\theta(x_t, t, c, z) - (y - \xi)\|^2\Big]. \tag{1}$$

The ideal flow integrates $\dot{x}_t = v_\theta(x_t, t, c, z)$ from $x_0 = \xi$ to $t = 1$; deployment uses a discrete clipped sampler followed by action denormalization and the physical cap. Here t is generative flow time (Appendix B.1).

Model-based planning. Let $h_0 = e_{\mathrm{state}}(o)$ be a task-state representation, $\widehat{f}$ a transition model, $\widehat{r}$ a reward model and $\widehat{v}$ a terminal value readout on representations. A model-based controller such as TD-MPC2 (Hansen et al., 2024) scores candidate requests by

$$\widehat{J}_T(h_0, z_{0:T-1}) = \sum_{j=0}^{T-1} \beta^j \widehat{r}(h_j, z_j) + \beta^T \widehat{v}(h_T), \qquad h_{j+1} = \widehat{f}(h_j, z_j). \tag{2}$$

It executes the first request and replans. We write $\widehat{V}(s) = \widehat{v}(e_{\mathrm{state}}(O(s)))$ for the corresponding value on complete states; e_{state} is the identity when $h = o$. The score is written in canonical requests; Section 4.3 specifies their deployed parameterization. Task-state representation h, motion request z and generation noise ξ have distinct roles.

Regularity assumptions for analysis. On a common complete-state/request domain with finite-first-moment kernels, the metric-based analysis assumes bounded rewards and constants satisfying

$$\begin{aligned} W_1(P_G(\cdot \mid s, z), P_G(\cdot \mid s', z')) &\le L_s d(s, s') + L_z \rho(z, z'), \\ |R_G(s, z) - R_G(s', z')| &\le C_s d(s, s') + C_z \rho(z, z'). \end{aligned} \tag{3}$$

Here W_1 uses the task-state metric d. Bounds are uniform over the compared states and requests under the same execution, reward and discount conventions. Auxiliary commands are fixed or the bounds hold uniformly over them; infinite-horizon arguments require a forward-invariant domain. The analysis is conditional on these regularity hypotheses. Appendix B.2 specifies integrability, observation and boundary conditions; Appendix B.4 distinguishes task termination from external truncation.

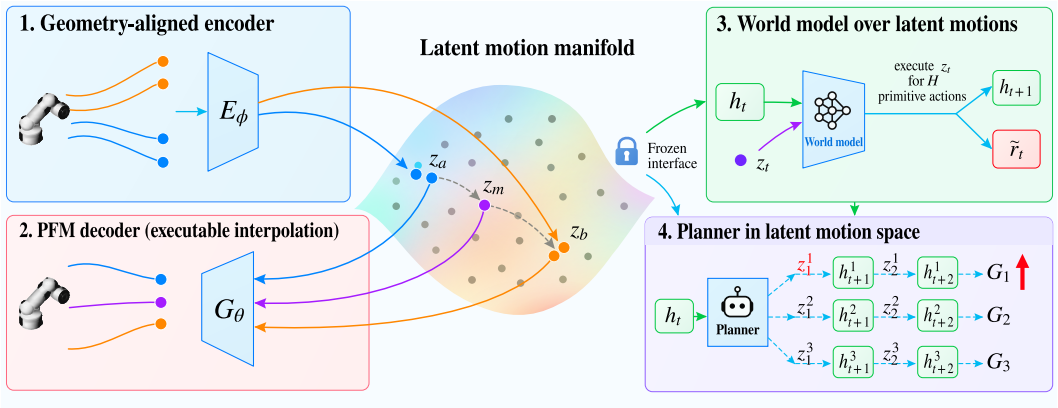

Figure 2: **A shared motion manifold for execution, prediction and planning.** (1) Geometry-aligned encoding organizes training motions into behavior-grounded requests. (2) PFM decodes anchors and an interpolated request into context-conditioned motions. (3) The world model predicts motion-level state transitions and rewards. (4) The planner scores candidate sequences and executes the first request of the chosen sequence. The surface and trajectories are schematic; panel (3) illustrates $K = H$, while execution generally uses a $K \leq H$ prefix before replanning.

4 PaLMR: Learning and Planning with Latent Motions

PaLMR learns a reusable motion interface from diverse local motions collected in open environments without downstream task rewards or task-specific demonstrations; collection details are given in Appendix F.2. Figure 2 links motion encoding and execution to prediction and planning through this shared space; Algorithm A.1 gives the training and control procedure.

4.1 Learning Geometry-Aligned Motion Representations

A motion encoder maps a recorded relative trace b and its motion actions to a normalized request,

$$z = E_\phi(b, a^{\text{mot}}) = \text{normalize}(\mu_\phi(b, a^{\text{mot}})). \tag{4}$$

Relative traces remove the coordinate origin from the motion description. A training-only reconstruction head recovers both the trace and its controls, with $\mathcal{L}_{\text{rec}} = \text{MSE}(\hat{b}, b) + \text{MSE}(\hat{a}, a^{\text{mot}})$. To organize requests by motion similarity, let d_τ be a detached, clamped soft-DTW dissimilarity (Cuturi & Blondel, 2017; Blondel et al., 2021) between traces. After reconstruction warm-up, TRAIN quantiles calibrate fixed thresholds m, M, and the encoder minimizes

$$\mathcal{L}_E = \mathcal{L}_{\text{rec}} + \tfrac{1}{2}\mathbb{E}[(\rho(z, z') - Md_\tau)_+] + \tfrac{1}{2}\mathbb{E}[(md_\tau - \rho(z, z'))_+]. \tag{5}$$

Reconstruction preserves each recorded motion; the two hinges encourage similar traces to have nearby codes and distinct traces to remain separated, as illustrated in Figure 2 (1). We then discard the reconstruction head and freeze the encoder. Section 5.2 evaluates this objective through motion prediction. Transformer architecture, training perturbations and calibration are specified in Appendix F.3.

4.2 Learning Context-Conditioned Motion Execution

PFM, our Parameterized Flow Matching module, trains a conditional generator on the frozen encoder's codes using Equation 1. The code specifies the requested motion, while context $c(s)$ supplies the current kinematic conditions needed to realize it. Figure 2 (2) illustrates the same decoder acting on two encoded requests, z_a, z_b, and their interpolant z_m. Auxiliary actuation, such as a gripper profile, is selected separately by the high level and assembled with the generated motion actions.

The decoder objective supplements flow matching with action-proxy distillation, request-cycle consistency and local action-sensitivity regularization. An inverse head learns to recover dataset codes from actions; generated interpolated requests are recovered under shuffled contexts, and the

sensitivity term penalizes excessive local action changes. The exact objective, weights, update schedule and surrogate sampler are given in Appendix F.3. Deployment uses ten clipped Euler steps followed by physical action denormalization and capping. The encoder and generator then remain frozen throughout downstream task learning.

A useful motion space should allow small code adjustments without abrupt physical changes, while keeping distinct requests distinguishable. The following result makes these two requirements explicit for the complete execution chain.

Lemma 1 (Conditional sensitivity and separation)**.** *Fix a complete initial state, auxiliary command and common randomness. Let a_z be the generated physical chunk, $b_z = B(a_z)$ its complete trace including the endpoint, and $e(z)$ its request re-encoded from pre-action positions and actions. Assume uniformly on the query region that a_z is L_u-Lipschitz in z, B is L_B-Lipschitz into the trace metric δ, $\|e(z) - e(z')\| \le L_E \delta(b_z, b_{z'})$ for $L_E > 0$, and $\|e(z) - z\| \le \varepsilon_I$. Then*

$$\frac{\rho(z, z') - 2\varepsilon_I}{L_E} \le \delta(b_z, b_{z'}) \le L_B L_u \rho(z, z'). \tag{6}$$

The upper bound limits changes between neighboring requests; the lower bound preserves separation beyond the inversion-error scale. Appendix C.3 proves the lemma. Proposition C.1 further relates target accuracy to anchor fidelity and context/noise variation. Section 5.4 evaluates the corresponding execution behavior through response prediction, interpolated requests and changed initial velocities.

4.3 Learning world models over motion requests

Each downstream task learns its own model and controller over the frozen interface. The controller parameterizes requests by raw actions u, whose motion component is mapped to $z = N(u^{\mathrm{mot}})$ for physical execution; replay and task-model inputs retain u. Executing a request for $K \le H$ primitive steps produces a transition $(o, u, \bar{r}, o')$, where $\bar{r}$ is the accumulated reward. With $h = e_{\mathrm{state}}(o)$, or $h = o$ when no state encoder is used, $\widehat{f}(h, u)$ predicts the next task-state representation and $\widehat{r}(h, u)$ the macro reward, denoted $\tilde{r}_t$ in Figure 2 (3). Transition, reward and value components use the high-level learner's native objectives. The state representation h describes the environment; the motion code z is an action coordinate. Model learning and planning alternate within the online interaction loop. The figure and analysis use canonical requests; Appendix D.3 relates the two parameterizations, including radial prediction error and the implemented normalization branches.

To connect these coordinates to prediction, fix a bounded $L_{\widehat{V}}$-Lipschitz value $\widehat{V}$ and define the true one-step backup $Q_{\widehat{V}}(s, z) = R_G(s, z) + \beta \mathbb{E}_{P_G}[\widehat{V}(S')]$. Let $\widehat{Q}_m$ be the fixed model's predicted backup. The next result separates calibration on observed pairs from generalization to nearby states and requests.

Proposition 1 (Predictive generalization over motion requests)**.** *On a common domain containing queried and calibration pairs, suppose calibration expected-backup error is at most ε_A, and every query has one calibration pair within state distance h_s and request distance h_z. If true and predicted backups have Lipschitz constants $L_{Q,s}, L_{Q,z}$ and $\widehat{L}_{Q,s}, \widehat{L}_{Q,z}$ under the same state/request metrics, then*

$$\begin{aligned} \varepsilon_M := \|Q_{\widehat{V}} - \widehat{Q}_m\|_\infty \le \overline{\varepsilon}_M, \\ \overline{\varepsilon}_M := \varepsilon_A + (L_{Q,s} + \widehat{L}_{Q,s}) h_s + (L_{Q,z} + \widehat{L}_{Q,z}) h_z. \end{aligned} \tag{7}$$

Prediction error depends on calibration accuracy, joint state–request coverage and true and learned regularity. Equation 3 permits $L_{Q,z} = C_z + \beta L_{\widehat{V}} L_z$, linking execution regularity to this bound. Section 5.2 examines this motivation for organized coordinates through held-out motion prediction on fixed transitions. Appendix D.1 proves the result and states its calibration scope.

4.4 Planning and composing latent motions

The controller proposes request sequences and scores them with its task model, reward and terminal value using Equation 2. In Figure 2 (4), z_j^i denotes request j of candidate i, and G_i its predicted return $\widehat{J}_T$. For the first selected request, it maps the raw motion action to $z = N(u^{\mathrm{mot}})$ and executes

$$a^{\mathrm{mot}}_{0:H-1} = G_\theta(c, z, \xi), \qquad a_{0:K-1} = \mathrm{Assemble}(a^{\mathrm{mot}}_{0:K-1}, \eta). \tag{8}$$

The controller replans from the new observation. Candidate proposals explore the continuous request space; an optional TRAIN-fitted prior changes proposals without updating the frozen interface (Appendix G.11).

A nonempty finite menu $B \subset \mathcal{Z}$ describes candidate resolution within this continuous family; the main algorithm searches continuous requests. Let $h_B = \sup_{z \in \mathcal{Z}} \min_{b \in B} \rho(z, b)$ and let $J_T^v(s, \mathbf{z})$ be the true expected T-decision discounted return with bounded L_v-Lipschitz terminal value v. Requests are open loop within a scored sequence, while each chunk is generated in its current execution context.

Theorem 1 (Motion coverage and model-based planning)**.** *Under Equation 3 and the conventions of Appendix B.2, suppose $|\widehat{J}_T - J_T^v| \le \varepsilon_J$ uniformly on $\mathcal{Z}^T$ and the chosen $\widehat{\mathbf{b}} \in B^T$ is within $\varepsilon_{\mathrm{opt}}$ of the best predicted score over B^T. Then*

$$\sup_{\mathbf{z} \in \mathcal{Z}^T} J_T^v(s, \mathbf{z}) - J_T^v(s, \widehat{\mathbf{b}}) \le \mathcal{C}_T h_B + 2\varepsilon_J + \varepsilon_{\mathrm{opt}}, \tag{9}$$

where $\mathcal{C}_T$ depends on the task regularity constants, horizon, discount and terminal-value regularity, as specified in Equation E.3.

The bound separates motion coverage, score prediction and candidate search. Adding requests preserves anchors and can reduce h_B; prediction and search errors determine how well the planner uses this coverage. Section 5.3 examines this mechanism by enriching the request menu with fixed models and equal scoring budgets. Appendices E.3 and E.4 give the proof and replanning analysis. Theorem D.1 further links uniform model, value-consistency and selection errors to the quality of a fixed fitted policy.

5 Experiments

We collect task-independent trajectories using scripted local motions with varied directions, distances and initial velocities in open space, and free end-effector motions in `reach-v3`, without downstream rewards or task-specific demonstrations. One frozen motion interface per domain supports downstream RL in three PointMaze layouts (U, Medium and Hard) and goal-observable Meta-World v3 manipulation tasks (Yu et al., 2020); open-space 2D executions support the controlled studies. Collection and evaluation protocols are in Appendix F.1. Our experiments answer whether PaLMR improves downstream learning efficiency and task performance, whether geometry-aligned coordinates improve motion prediction, whether fixed models can exploit interpolated motion options, and whether PFM realizes predictable, executable and state-adaptive motions.

5.1 Main results: downstream model-based learning

We evaluate all fifty Meta-World tasks and show three illustrative tasks in Figure 3(d–f). The comparison includes PaLMR, primitive-action baselines flat TD-MPC2 (Hansen et al., 2024) and SAC (Haarnoja et al., 2018), and four generative-policy RL baselines: DSRL-SAC (Wagenmaker et al., 2025), DPPO (Ren et al., 2025), ReinFlow (Zhang et al., 2025) and FPO++ (Yi et al., 2026). Every method is trained with **three independent online RL seeds per task**. Curves report periodic success against primitive environment interactions; offline pretraining is accounted separately. PaLMR uses prior mixing in navigation. Evaluation budgets and aggregation are specified in Appendix F.1, and baseline descriptions in Appendix F.4.

Navigation. Figure 3(a–c) shows rapid learning in U-Maze and strong performance as the layouts become more demanding. PaLMR attains higher final success than all four generative-policy baselines in every layout. The same frozen interface, pretrained on open-space motions rather than task-specific routes, supports goal-directed control across these different obstacle layouts without low-level retraining. Full learning curves and independent final evaluations are reported in Appendices G.2 and G.4.

Manipulation. Across the complete fifty-task evaluation, PaLMR achieves the highest task-balanced mean success in the common-budget comparison: 79.67%, compared with 67.87% for flat

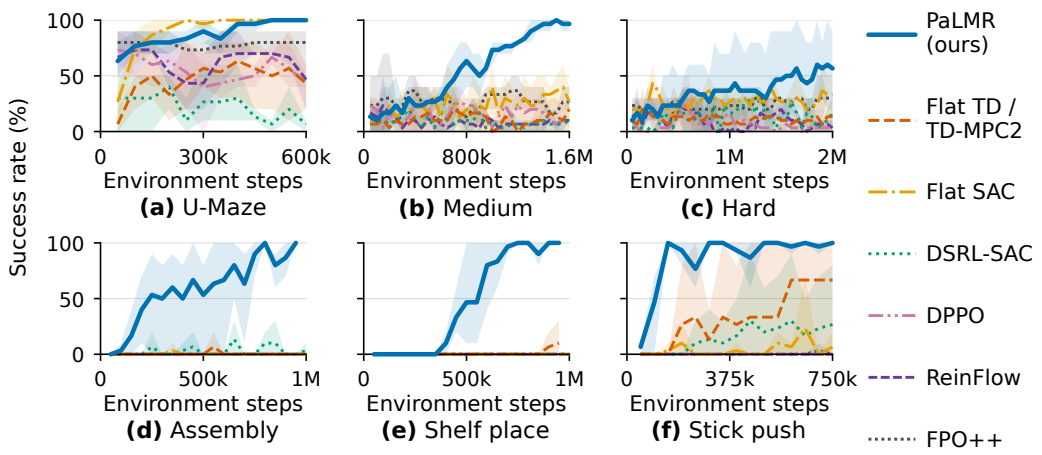

Figure 3: **Learning curves for navigation and manipulation.** (a–c) PointMaze layouts; (d–f) three outcome-selected examples from the fifty-task Meta-World benchmark, summarized in Appendix Table G.1. Lines and bands show the mean and min–max range over three online RL seeds, with ten episodes per checkpoint. Full curves, task selection and display-window rules are in Appendix G.2.

TD-MPC2. It also achieves the highest normalized learning-curve AUC among all seven systems. Appendix Table G.1 and Appendix G.6 report the full-cohort aggregates and benchmark-level differences; exploratory task-specific endpoints and their budget-selection rule are detailed in Appendix G.1.

The assembly, shelf place and stick push examples in Figure 3(d–f) illustrate how this advantage appears in multi-stage task behavior. PaLMR learns successful behavior in all three examples and maintains its advantage on stick push as the strongest baseline improves. Complete curves and the example-selection protocol are given in Appendix G.2.

Together, these results support the utility of the reusable motion interface for online learning efficiency and downstream task performance. The following studies examine the predictive coordinates, planning options and physical execution underlying this complete-system result.

5.2 Geometry-aligned coordinates for motion prediction

We test whether the geometric objective in Section 4.1 improves the predictive usefulness of learned motion coordinates. Geometry-informed and reconstruction-only encoders share pretraining data, architecture, code dimension and training updates, differing only in the geometric hinges. Each frozen encoder supplies coordinates for the same recorded PFM executions. Matched predictors take the initial velocity and these coordinates to predict endpoint displacement and final velocity, using the same training budget and three paired seeds. Only the predictors are trained in this comparison; physical records and the motion executor remain fixed. This is a fixed-record representation diagnostic; Section 5.3 evaluates model-based selection using requests available before execution.

Geometry-informed coordinates reduce withheld-direction endpoint error from 56.85 to 23.30 mm (59.0%), and withheld-distance error from 37.49 to 17.37 mm (53.7%), as shown in Figure 4(a). These gains support the role of request geometry in predictive generalization (Proposition 1), here evaluated through held-out motion prediction. Appendix G.7.1 provides the full protocol and seed-level results. A complementary coordinate-remapping study finds similar accuracy under an orthogonal transformation but higher error under a nonlinear twist, further illustrating the importance of coordinate organization (Appendix G.7.2).

5.3 Planning with interpolated motion options

Section 5.2 isolated predictive coordinates on fixed physical transitions; we now ask whether fixed learned models can use additional motion requests. Candidate requests are available before execution: 47 TRAIN request entries (43 distinct codes), extended by 24 predefined direction and distance

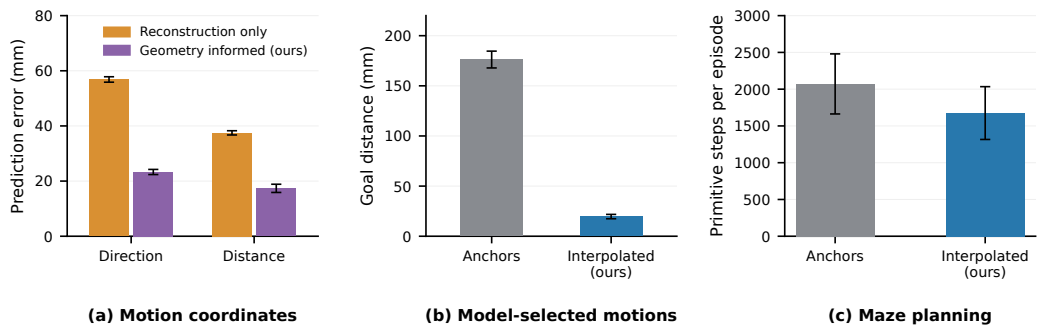

Figure 4: (a) Endpoint prediction error for geometry-informed and reconstruction-only coordinates. (b) Goal distance of model-selected motions and (c) primitive steps per Medium Maze episode, using anchors alone or with interpolated requests. Bars show means; error bars indicate one standard deviation across three paired predictor seeds in (a), and one standard error across 24 targets in (b) and 30 paired episodes in (c).

interpolations to 71 entries (67 distinct codes). The same frozen encoder and PFM execute every request. We evaluate choice by a frozen response predictor at planning length $T = 1$ and navigation with a fixed $T = 3$ planner in Medium.

Frozen-model choice. A fixed request-conditioned response predictor, trained on the 47 anchor response groups with three seeds, chooses among 71 scored slots per menu by predicted final position before any outcome is read; the anchor arm repeats 24 requests to match the scoring budget. Across 24 local targets from a common resting context, the mean actual goal distance of its choices falls from 176.22 to 19.61 mm when interpolated requests are added (Figure 4(b)). The local endpoint hit rate, the fraction of complete chunk endpoints within 50 mm of the target, rises from 0% to 100%. Under all three predictor seeds, the enriched menu selects an added request for every target. Thus, the frozen predictor can select useful interpolated motions without further training. Outcomes are evaluated using the five recorded executions of each selected request.

Fixed multi-step planner. We next give the two menus to the native $T = 3$ planner of three frozen prior-off Medium checkpoints. World model, reward, value, actor and PFM remain unchanged and receive no optimizer update. Both arms project each candidate code to its nearest permitted menu code before scoring; each decision generates $H = 100$ actions and executes up to $K = 100$ from identical paired resets. Both menus succeed in the same $25/30$ episodes. Across all thirty episodes, the enriched menu lowers mean primitive steps from 2071.3 to 1675.1 (19.1%) (Figure 4(c)), and the planner executes an added request in 200 of 505 decisions (39.6%). The shorter episodes with fixed models and equal scoring budgets support the coverage and selection mechanism of Theorem 1. Appendix G.10 gives the training budget, paired protocol and Table G.12.

Online proposals. Candidate generation during learning is a separate question: mixing TRAIN-fitted proposals into the navigation planner raises Medium/Hard final success from $94.67/34.67\%$ to $100/46\%$. These task models are trained separately under each proposal with the same frozen motion interface, so this comparison measures the complete online system (Appendices G.5 and G.11).

5.4 PFM REALIZES PREDICTABLE, EXECUTABLE AND STATE-ADAPTIVE MOTIONS

Section 5.2 changed predictive coordinates while keeping physical records fixed; here the shared geometry-informed encoder and requested codes stay fixed, and the low-level decoder that realizes them changes. PFM is compared with a deterministic regression decoder trained on the same data, reading the same codes and current velocity, with approximately matched capacity (3.476M versus 3.481M parameters) and the same 10^6-update budget. The regression decoder predicts action sequences directly using an action squared-error objective. Decoder architecture and training objective change together (Appendix F.3). Each method is a single frozen low-level checkpoint. Figure 5 tests three properties of the executed motion relevant to planning.

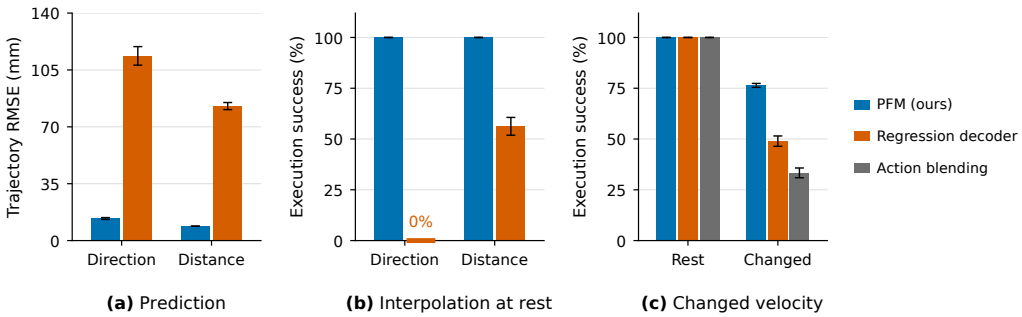

Figure 5: 2D decoder comparison: (a) trajectory-coordinate RMSE for held-out direction and distance requests; (b) execution success for interpolated requests from rest; (c) success for known motions from rest and under changed initial velocity. Success requires completing $H = 100$ actions with endpoint error ≤ 50 mm. Error bars indicate one standard error across three predictor seeds in (a) and one binomial standard error across physical executions in (b–c).

Predictable responses. Each predictor learns its own decoder's actual outcomes, so this controlled diagnostic changes the physical responses while keeping the requested coordinates fixed. Both predictors read the pre-execution velocity and code and output 100 relative XY positions, using identical architecture, 5000 updates and three seeds. On held-out visited-context requests, trajectory-coordinate RMSE is 13.58 versus 113.67 mm for direction and 8.87 versus 82.78 mm for distance (PFM versus regression decoder), with every PFM seed below every regression-decoder seed (Figure 5a). A separate 96-group test keeps codes known and holds out initial velocities; trajectory RMSE is 5.36 versus 83.48 mm. PFM's recorded responses are therefore easier to predict in these matched-budget comparisons. Appendix G.8 reports their protocol, resting-request comparison and predictor seeds.

Accurate execution of intermediate requests. Direction and distance interpolation requests from rest test the executable options that motivate Section 5.3. PFM succeeds in all 320 direction and 640 distance executions (100/100%), while the regression decoder succeeds on 0/64 and 72/128 queries (0/56.25%). Both solve every known/start request from rest (PFM 1920/1920, regression decoder 384/384). PFM thus executes every tested intermediate request in this panel. Appendix G.12 gives the corresponding success counts.

State-adaptive execution of known requests. We keep the codes, goals and positions of the 384 known-motion/start queries fixed and replace resting initial velocity with velocities recorded during downstream training, without retraining. All three methods succeed on every resting query. Under changed velocity, PFM succeeds in 76.41% of executions (1467/1920), versus 48.96% for the regression decoder (188/384) and 33.33% for action blending (128/384). Both learned decoders receive current velocity; action blending uses fixed weighted mixtures of stored action sequences without velocity conditioning. Together, these interpolation and velocity tests support the role of accurate execution across requests and contexts in Section 4.2 and Proposition C.1. Appendices G.13 and G.12 specify the protocol and counts.

6 Conclusion and Limitations

We presented PaLMR, a reusable generative motion manifold that connects motion execution, world-model learning and planning. Its geometry-informed codes and context-conditioned decoder turn task-independent experience into executable motions that downstream controllers can interpolate and compose. Our analysis relates execution regularity and motion coverage to prediction and planning error, while controlled studies demonstrate more accurate prediction and effective use of intermediate motions. Across fifty manipulation tasks, a shared frozen interface achieves the highest aggregate early success and learning-curve AUC among the evaluated systems. A limitation is that evaluation is confined to simulation; transfer to physical robots remains future work.

Reproducibility statement

Implementation and evaluation protocols are detailed in Appendix F. Code is included in the supplementary material.

AI use statement

Generative AI assisted with code implementation, mathematical verification, and manuscript polishing.

References

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 264–273, 2018. URL https://proceedings.mlr.press/v80/asadi18a.html.

Mathieu Blondel, Arthur Mensch, and Jean-Philippe Vert. Differentiable divergences between time series. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3853–3861, 2021. URL https://proceedings.mlr.press/v130/blondel21a.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2303.04137v4.

John Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1009–1018, 2018. URL https://proceedings.mlr.press/v80/co-reyes18a.html.

Marco Cuturi and Mathieu Blondel. Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 894–903, 2017. URL https://proceedings.mlr.press/v70/cuturi17a.html.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pp. 1398–1403, 2002. URL https://doi.org/10.1109/ROBOT.2002.1014739.

Hokyun Im, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. Latent policy steering through one-step flow policies. In *Proceedings of Robotics: Science and Systems*, 2026. URL https://arxiv.org/abs/2603.05296.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.07969.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00914.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.08887.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204, 2021. URL https://proceedings.mlr.press/v155/pertsch21a.html.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.00588.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1907.01657.

Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2262–2272, 2023. URL https://proceedings.mlr.press/v205/shi23a.html.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282, 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control, 2026. URL https://arxiv.org/abs/2602.02481.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Renhao Zhang, Haotian Fu, Mingxi Jia, George Konidaris, Yilun Du, and Bruno Castro da Silva. From noise to control: Parameterized diffusion policies. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.00336.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.22094.

A PaLMR Training and Control Algorithm

Algorithm A.1 summarizes Section 4. One encoder–generator pair is pretrained per physical domain and reused across its downstream tasks. Training objectives and domain-specific execution/update schedules are specified in Appendices F.3 and F.5.

Algorithm A.1 PaLMR: reusable motion pretraining and task-specific model-based control

Require: Task-independent motion data $\mathcal{D}$; generation/execution horizons H, K; planning horizon T; pretraining and task-interaction budgets.
Ensure: Frozen motion encoder E_ϕ, executor G_θ, and a learned model/controller for each task.
Phase I: learn motion coordinates
Initialize E_ϕ and the training-only reconstruction head.
Warm up with $\mathcal{L}_{\text{rec}}$; calibrate m, M on TRAIN pairs and keep them fixed.
Train with Equation 5, using the prescribed stochastic code perturbation.
Freeze E_ϕ, use its normalized mean codes, and discard the reconstruction head.
Phase II: learn conditional execution
Fit action-normalization statistics on TRAIN and form targets y and contexts c.
Encode each recorded motion: $z_i \leftarrow E_\phi(b_i, a_i^{\text{mot}})$.
Initialize the conditional flow field v_θ and action-inversion proxy q_a.
for each decoder update ℓ **do**
    Sample TRAIN tuples (y, c, z), flow times t, and Gaussian noises ξ.
    Form $x_t = (1-t)\xi + ty$ with target velocity $y - \xi$.
    Update the flow and proxy with Equation F.1 and its auxiliary schedule.
end for
Freeze the low level; retain the ten-step clipped sampler for execution.
Fit a TRAIN-only proposal prior when enabled; keep it frozen downstream.
Phase III: learn and plan in each downstream task
for each downstream task **do**
    Initialize its task model, value/proposal components, replay $\mathcal{R}$, and step counter $n \leftarrow 0$.
    Reset the task and obtain observation o and execution context c.
    while n is below the task's primitive-interaction budget **do**
        Choose u by warm-up exploration, or refine raw length-T action sequences with the learned score (Equation 2) and take the selected sequence's first action.
        Form $z \leftarrow N(u^{\text{mot}})$ and the auxiliary command η from u.
        Draw fresh ξ and generate $a^{\text{mot}}_{0:H-1} \leftarrow G_\theta(c, z, \xi)$.
        Assemble auxiliary controls and execute at most K actions, stopping at an episode end or the remaining interaction budget.
        Observe o', next context c', accumulated reward $\bar{r}$, actual step count L, and termination/truncation flags.
        Store $(o, u, \bar{r}, o')$ with the learner's bootstrap metadata in $\mathcal{R}$; set $n \leftarrow n + L$.
        Perform the scheduled native replay updates of the task model, reward, value and proposal components; keep E_ϕ, G_θ frozen.
        Continue from (o', c'), or reset the task after termination/truncation.
    end while
end for

The model predicts task consequences in its state representation h, with $h = o$ when no state encoder is used. It scores raw controller actions u; only physical execution applies N to obtain z. Imagined transitions therefore use the learned task model rather than simulator rollouts of the generator. Planning and task-model updates alternate online, while the motion encoder, decoder and optional pretrained prior remain fixed. Finite menus are experimental interventions and analytical approximations, not a restriction in Algorithm A.1.

B Extended Background and Analysis Conventions

This appendix expands the notation and assumptions of Section 3. It separates generative time, physical execution and high-level decisions, and gives the common setting used by the proofs that follow. The analysis conditions on a fixed motion interface, fitted predictive models and a fixed execution convention. Under the stated regularity, calibration and domain assumptions, the results characterize how prediction error and motion coverage affect planning. The experiments examine the

corresponding mechanisms through finite-sample diagnostics; the conditional bounds concern fixed fitted systems rather than convergence of online training.

B.1 Conditional generation and physical execution

A normalized recorded motion chunk $y \in \mathbb{R}^{Hd_a}$ is paired with execution context c and request z. Assume $\mathbb{E}\|y\|^2 < \infty$ and draw $\xi \sim \mathcal{N}(0, I)$ independently of (y, c, z), with independent $t \sim \mathcal{U}[0, 1]$. The path $x_t = (1-t)\xi + ty$ in Equation 1 has derivative $y - \xi$. Squared-error regression on this target has the conditional minimizer $v^*(x, t, c, z) = \mathbb{E}[y - \xi \mid x_t = x, t, c, z]$, almost everywhere under the training joint law. Conditioning on (x_t, t, c, z) decomposes the regression error into squared distance from this conditional mean and a variance independent of the predictor. This identity characterizes the flow-matching target on the training law; t is generative time rather than a physical control step (Lipman et al., 2023).

Let $\tilde{G}_\theta$ be the deployed sampler in normalized action coordinates, and let Φ apply TRAIN denormalization and the physical action cap. The physical generator is $G_\theta = \Phi \circ \tilde{G}_\theta$. A request z identifies motion behavior, $c(s)$ supplies execution-state information, and ξ initializes conditional generation. The online controller changes the request and context, not the pretrained generator weights. Appendix C.2 analyzes the actual discrete clipped sampler separately from the population regression identity.

B.2 Task state, induced dynamics and regularity

Let $(\mathcal{S}, d)$ be a Polish space of complete task states and $\mathcal{Z}$ a nonempty compact metric query region, common to all compared states. The observation $o = O(s)$ and execution context $c = c(s)$ are measurable maps; observation noise or controller memory, when relevant, is included in the complete state. The learned observation representation need not itself be Markov. Policies, kernels and selectors are measurable; the Bellman and sequence suprema used below are measurable and admit measurable approximate maximizers. Compact action sets with backups measurable in the state and continuous in the action are a sufficient alternative for attained maxima and measurable selectors. All regularity/error constants are finite and nonnegative, $0 \le \beta < 1$, and all rewards and value functions used below, including reference-class rewards, are bounded.

Kernels after averaging generator/environment randomness and all compared rollout laws have finite first moments under d. Bounds used in an infinite-horizon argument hold uniformly on a common forward-invariant state domain and every compared action. Finite-horizon arguments require the bounds on every pair of states in the supports of the compared rollouts, including cross-pairs used by a coupling. If a learned kernel is rolled out, its states must also lie in a domain on which the true kernel and the stated errors are defined. These domain requirements include both calibration pairs and queried rollout states.

Suppress a fixed auxiliary command. For a fixed nominal prefix K, let $a(s, z, \xi) \in \mathbb{R}^{Kd_a}$ be the generated physical action prefix before any environment stopping event. With $P^{(K)}$ the physical transition kernel under that prefix and $r^{(K)}$ its expected accumulated reward,

$$P_G(\cdot \mid s, z) = \mathbb{E}_\xi P^{(K)}(\cdot \mid s, a(s, z, \xi)), \qquad R_G(s, z) = \mathbb{E}_\xi r^{(K)}(s, a(s, z, \xi)).$$

Generator noises are fresh and independent across decisions and of future environment innovations, conditional on the current state. The physical kernel integrates environment randomness. Early task termination is represented by absorption with zero subsequent reward, so these fixed-length prefixes remain well defined; the simulator need not execute the padded tail. Other stopping conventions are distinguished in Appendix B.4.

Sufficient physical conditions for Equation 3. Suppose, under a common-base-noise coupling,

$$\mathbb{E}\|a(s, z, \xi) - a(s', z', \xi)\| \le U_s d(s, s') + U_z \rho(z, z').$$

Assume, for every compared state and fixed-length physical prefix,

$$W_1(P^{(K)}(\cdot \mid s, a), P^{(K)}(\cdot \mid s', a')) \le D_s d(s, s') + D_a \|a - a'\|,$$
$$|r^{(K)}(s, a) - r^{(K)}(s', a')| \le D_{r,s} d(s, s') + D_{r,a} \|a - a'\|.$$

The mixture inequality for W_1 gives

$$W_1(P_G(\cdot \mid s,z), P_G(\cdot \mid s',z')) \le D_s d(s,s') + D_a \mathbb{E}\|a(s,z,\xi) - a(s',z',\xi)\|.$$

Consequently one may choose the valid upper-bound constants

$$L_s = D_s + D_a U_s, \quad L_z = D_a U_z, \qquad C_s = D_{r,s} + D_{r,a} U_s, \quad C_z = D_{r,a} U_z.$$

Common noise supplies a mathematical coupling; actual decisions use fresh draws. Auxiliary commands are fixed throughout a comparison; joint optimization uses a state-independent product action class with regularity uniform over the auxiliary coordinate. When only motion coordinates are rounded and all auxiliary commands remain available, the sequence domains are $(\mathcal{Z} \times \mathcal{E})^T$ and $(B \times \mathcal{E})^T$, with suprema if necessary; the rounding keeps η unchanged. Alternatively cover the full product request. All policy suprema must use that same action class, and actual model inputs include η.

Scope of the regularity hypotheses. Equation 3 imposes regularity on the complete task transition and reward, including contact and termination. For example, $r(x) = \mathbf{1}\{x \ge 0\}$ has $|r(\epsilon) - r(-\epsilon)|/(2\epsilon) = 1/(2\epsilon)$ and no finite global Euclidean Lipschitz constant. A stochastic expected macro reward can be smoother, with its regularity established separately. The coverage and metric-generalization results apply on the declared metric/domain where these hypotheses hold. The Bellman-residual policy argument in Appendix D.2 also applies without metric regularity when a uniform backup-error bound is supplied directly.

B.3 Model predictions and canonical action coordinates

Write $e_O(s) = e_{\text{state}}(O(s))$ for the task-state representation pulled back to complete states, or $e_O(s) = O(s)$ when no state encoder is used. A task model can form its scalar backup through

$$\widehat{Q}_m(s,z) = \widehat{r}(e_O(s),z) + \beta \widehat{v}(\widehat{f}(e_O(s),z)), \qquad \widehat{V}(s) = \widehat{v}(e_O(s)).$$

Any terminal mask is included consistently in these scalar predictions. Model regularity in the proofs concerns these functions of (s,z) under d, not merely the network as a function of h. The theoretical model error concerns the scalar backup, including information retained by e_O; physical next-position and trajectory errors are different diagnostic quantities. A fixed observation-based controller is a policy on complete states, while V_G^* denotes the full-state Markov optimum over the specified interface.

Canonical requests express the physical action family, while the deployed model reads raw u. On its exact-normalization branch, $N(u) = u/\|u\|$. Navigation instead uses $N_\epsilon(u) = u/\max(\|u\|, \epsilon)$ with $\epsilon = 10^{-6}$, whose image for $\|u\| < \epsilon$ is inside the unit ball. Appendix D.3 separates the sphere-valued branch from the full implemented action image, including manipulation's fixed-direction fallback. Sphere-valued encoder means and anchor mixtures also require nonzero normalization inputs. A positive lower bound on those input norms supplies inverse-norm control for the usual normalization Lipschitz bound.

B.4 Execution horizons, discount and reference classes

The generator produces H primitive actions, a decision executes a nominal prefix $K \le H$, and a plan contains T high-level decisions. Every comparison fixes the execution interval, auxiliary-actuator class, reward aggregation and discount. With fixed K and raw-step discount γ, within-chunk rewards $\sum_{j=0}^{K-1} \gamma^j r_j$ and $\beta = \gamma^K$ reproduce the raw-time objective. Summed undiscounted chunk rewards with a fixed β instead define a per-decision objective, as used in PointMaze. These conventions are not interchangeable across interfaces with different decision intervals.

True terminal states absorb with zero subsequent rewards and zero terminal continuation values. A task-defined time limit is made Markov by including remaining time in s and terminating at its boundary. An external collection timeout with continued bootstrapping is instead a censored trajectory of the continuing objective, not an absorbing task termination. PointMaze's reported time-limit bootstrapping uses the latter interpretation; capped evaluation success and continuing discounted value are distinct quantities. A final prefix shortened only by the training budget has a duration-specific transition law and cannot silently be identified with the nominal-K kernel. It must

be accounted for by an augmented execution state/duration or kept outside a fixed-K calibration claim.

For genuinely variable decision duration ℓ, a raw-time objective uses the joint next-state/duration law and the discounted operator $\mathbb{E}[\gamma^{\ell}V(S')]$, rather than replacing it by $\beta\mathbb{E}[V(S')]$ with an unrelated fixed β. The current results use the fixed-per-decision convention stated above. The reference class contains the generator's stochastic control laws under the same state information, stopping and discount conventions, so its representation gap is nonnegative.

Request fidelity specifies a descriptor and horizon. The recorded encoder trace b in Section 3 contains pre-action positions through $H-1$ and omits p_H. A complete execution trace $(p_k - p_0)_{k=0}^{H}$ includes the terminal position. Restricting this complete trace to a prefix is non-expansive in the unnormalized Euclidean norm. Thus a bound on the complete trace controls its prefix and endpoint, whereas a bound only on the encoder's pre-action trace need not control p_H. Full-H endpoint accuracy does not by itself imply an accurate K-step task transition, including object and contact state.

C Low-Level Regularity and Request-Fidelity Proofs

This appendix proves Lemma 1 from Section 4.2 and the supplementary request-fidelity result. Deployment conventions and the task-level setting are given in Appendix B.

C.1 Physical maps, metrics and coupling

Fix a complete admissible physical state s_0, an auxiliary command η, a common sampler draw ξ and simulator randomness. Write $a_z = \Phi(\tilde{G}_\theta(c(s_0), z, \xi))$ for the generated physical chunk and $b_z = B_{s_0}(a_z)$ for its complete relative execution trace, including the terminal position. Let $e(z)$ be the frozen encoder applied to the pre-action restriction of this trace and the associated actions. The trace metric δ is unnormalized time-aligned Euclidean distance. Encoder inputs and complete physical descriptors thus retain their different horizons (Appendix B.4). TRAIN denormalization is counted once in the physical decoder constant L_u.

All conditions are uniform on the declared query/physical-execution region, including contact and stopping conventions. The detached, clamped soft-DTW quantity used in training is a dissimilarity, not the trace metric used in these proofs. Because the encoder also reads actions, the trace-only condition on $e(z)$ in Lemma 1 is imposed on realized traces together with their associated action realizations. For fixed initial state and randomness, nearby codes with small physical separation must have correspondingly close re-encodings. This trace-only condition is assumed uniformly on the execution region and is separate from neural-encoder regularity in its full input space.

C.2 Regularity of the deployed sampler

Fix context and common noise. The normalized sampler uses $N = 10$ clipped Euler steps,

$$x_{j+1} = \Pi(x_j + N^{-1} v_\theta(x_j, j/N, c, z)), \qquad x_0 = \xi.$$

Assume Π is non-expansive and v_θ is uniformly L_x-Lipschitz in x and L_z^v-Lipschitz in z on a product domain containing all compared iterates and crossed inputs $(x_j(z'), z)$. Splitting the field difference at that crossed input gives

$$\Delta_{j+1} \le (1 + L_x/N)\Delta_j + (L_z^v/N)\rho(z, z'), \qquad \Delta_j = \|x_j(z) - x_j(z')\|, \quad \Delta_0 = 0.$$

Induction yields

$$\|\tilde{G}_\theta(c, z, \xi) - \tilde{G}_\theta(c, z', \xi)\| \le L^{\mathrm{norm}} \rho(z, z'), \qquad L^{\mathrm{norm}} = \frac{L_z^v}{N} \sum_{j=0}^{N-1} (1 + L_x/N)^j. \tag{C.1}$$

For $L_x > 0$, the constant is $\frac{L_z^v}{L_x}[(1 + L_x/N)^N - 1]$; for $L_x = 0$, it is L_z^v. Uniformity over Gaussian sampler draws requires constants covering their initial-input domain. The superscript distinguishes the vector-field constant from the task-transition constant L_z.

Coordinatewise denormalization has operator norm $\kappa_\Phi = \frac{1}{2}\max_i r_i$, where r_i are the effective TRAIN ranges used by the deployed affine map, including any range floor. The physical action cap is non-expansive, so

$$L_u^{\mathrm{phys}} \le \kappa_\Phi L^{\mathrm{norm}}. \tag{C.2}$$

The physical constant composes with the rollout map in Lemma 1.

C.3 Proof of Lemma 1

The physical maps above instantiate the lemma with $B = B_{s_0}$ and chord distance $\rho(z, z') = \|z - z'\|$.

Proof. Composing decoder and physical-map regularity gives

$$\delta(b_z, b_{z'}) \le L_B \|a_z - a_{z'}\| \le L_B L_u \rho(z, z').$$

For separation,

$$\rho(z, z') \le \|z - e(z)\| + \|e(z) - e(z')\| + \|e(z') - z'\| \le 2\varepsilon_I + L_E \delta(b_z, b_{z'}).$$

Divide by $L_E > 0$ and rearrange. The displayed lower bound is informative when $\rho(z, z') > 2\varepsilon_I$; otherwise nonnegativity of the metric is stronger. □

Sensitivity and separation are distinct: clipping preserves the upper-bound argument, while the lower bound requires realized inversion. Task-state and reward regularity are imposed separately in Equation 3.

C.4 Linking the action proxy to realized inversion

The proxy q_a reads context and normalized actions, with the objective in Appendix F.3. Write $\tilde{a}_z = \tilde{G}_\theta(c, z, \xi)$. If the deployed map satisfies both uniform conditions

$$\|q_a(c, \tilde{a}_z) - z\| \le \varepsilon_{\mathrm{cyc}}, \qquad \|q_a(c, \tilde{a}_z) - e(z)\| \le \varepsilon_{\mathrm{proxy}},$$

then the triangle inequality gives $\varepsilon_I \le \varepsilon_{\mathrm{cyc}} + \varepsilon_{\mathrm{proxy}}$. The proxy-matching condition connects action-space inversion to the realized re-encoding. Training uses a four-step unclipped surrogate, whereas deployment uses ten clipped steps; the two uniform conditions above concern the deployed map. The proxy shares the flow optimizer through a separate parameter group, with gradient clipping applied to the flow parameters.

C.5 Physical request fidelity

Let q index a physical request, $b^\star(q)$ its independently specified endpoint or trace, and $z(q)$ its code constructed from a nonempty finite set of TRAIN anchors. Use a common physical-descriptor metric δ throughout. Let $F_s(z, \xi)$ be the descriptor realized from complete state s with fixed auxiliary command. Here environment randomness is conditioned on; equivalently it may be included in a common base-noise variable when using a stochastic execution map. For reference state s_r and noise ξ_r, define

$$\begin{aligned} \Delta_c(q, s) &= \delta(F_s(z(q), \xi_r), F_{s_r}(z(q), \xi_r)), \\ \Delta_\xi(q, s, \xi) &= \delta(F_s(z(q), \xi), F_s(z(q), \xi_r)). \end{aligned} \tag{C.3}$$

These terms measure actual context and noise variation at a fixed request; context adaptation and sampler stability correspond to small values of these deviations.

Proposition C.1 (Conditional request-fidelity bound)**.** *Suppose $F_{s_r}(\cdot, \xi_r)$ is L_F-Lipschitz in chord distance on a region containing every query and anchor. For anchor q_i, define its reference execution error by*

$$\varepsilon_i = \delta(F_{s_r}(z(q_i), \xi_r), b^\star(q_i)).$$

Then

$$\begin{aligned} &\delta(F_s(z(q), \xi), b^\star(q)) \\ &\quad \le \min_i \left[\varepsilon_i + L_F \|z(q) - z(q_i)\| + \delta(b^\star(q_i), b^\star(q))\right] + \Delta_c(q, s) + \Delta_\xi(q, s, \xi). \end{aligned} \tag{C.4}$$

Proof. For any anchor q_i, insert the reference-noise, reference-state and anchor executions:

$$\begin{aligned}\delta(F_s(z(q),\xi),b^\star(q)) \le\ & \delta(F_s(z(q),\xi),F_s(z(q),\xi_r))\\ &+\delta(F_s(z(q),\xi_r),F_{s_r}(z(q),\xi_r))\\ &+\delta(F_{s_r}(z(q),\xi_r),F_{s_r}(z(q_i),\xi_r))\\ &+\delta(F_{s_r}(z(q_i),\xi_r),b^\star(q_i))+\delta(b^\star(q_i),b^\star(q)).\end{aligned}$$

The first two terms are Δ_ξ and Δ_c, the third is bounded by $L_F\|z(q)-z(q_i)\|$, and the fourth is ε_i. Minimize over the finite anchor set. □

If every request has an anchor simultaneously satisfying code distance h_z, target distance h_b and anchor error ε_A, its error is at most $\varepsilon_A + L_F h_z + h_b + \Delta_c + \Delta_\xi$. These three requirements must hold for the same anchor. Fixed-mode start panels use translated realized TRAIN-anchor displacements as targets, not generated test outcomes. Appendix E.1 connects expected executed-prefix fidelity to physical coverage; its expectation requires integrable descriptors.

C.6 Execution context and physical diagnostics

For the one-step scalar system $d_{\text{motion}} = v + a$, displacement $d^\star$ is realized by $a = d^\star - v$. If velocities v_1, v_2 have the same uninformative context and receive the same action, the triangle inequality gives

$$\max\{|v_1 + a - d^\star|, |v_2 + a - d^\star|\} \ge \tfrac{1}{2}|v_1 - v_2|.$$

The same conclusion holds pointwise under a common draw for an identical randomized action law. Execution-state information therefore serves a role separate from code geometry. Section 5.4 measures physical interpolation and context execution, while Section 5.3 studies the use of additional requests.

D World-Model Generalization and Policy-Error Proofs

This appendix proves Proposition 1, then converts model, value and selection residuals into policy error. All suprema below use the stated complete-state/action domain and fixed fitted networks. The common setting is in Appendix B.2; planning arguments follow in Appendix E.

D.1 Proof of Proposition 1

Proof. Fix the bounded $L_{\widehat{V}}$-Lipschitz value $\widehat{V}$ and its true backup from Section 4.3. Kantorovich–Rubinstein duality and Equation 3 give valid choices

$$L_{Q,s} = C_s + \beta L_{\widehat{V}} L_s, \qquad L_{Q,z} = C_z + \beta L_{\widehat{V}} L_z.$$

Smaller valid constants may also be used. Let $D_M = \{(s_i, z_i)\}$ have expected-backup residual at most ε_A. For every evaluated (s, z), select one pair satisfying both $d(s, s_i) \le h_s$ and $\rho(z, z_i) \le h_z$. Insert the true and predicted backups at that pair:

$$\begin{aligned}|Q_{\widehat{V}}(s,z) - \widehat{Q}_m(s,z)| &\le |Q_{\widehat{V}}(s,z) - Q_{\widehat{V}}(s_i,z_i)| + \varepsilon_A + |\widehat{Q}_m(s_i,z_i) - \widehat{Q}_m(s,z)|\\ &\le \varepsilon_A + (L_{Q,s} + \widehat{L}_{Q,s})h_s + (L_{Q,z} + \widehat{L}_{Q,z})h_z.\end{aligned}$$

Taking the supremum on that same domain proves Equation 7 and Proposition 1. □

State and request proximity are evaluated at the same calibration pair. The quantity ε_A bounds expected scalar-backup error; statistical estimation from noisy rewards and next states requires a sampling-error argument, with suitable independence or uniform control when the value/model is data-dependent. The proposition is a deterministic conditional bound. Regularity of the learned backup is measured after pulling its observation representation back to complete states, as in Appendix B.3.

D.2 Representation, model and decision error

The Bellman argument uses a uniform backup-error bound and a fixed discount. Proposition 1 supplies such a bound when its conditions hold throughout a common forward-invariant domain, including every action optimized over below.

Theorem D.1 (Representation, model, and decision error)**.** *Fix a bounded value $\widehat{V}$ and a model backup with $\|Q_{\widehat{V}} - \widehat{Q}_m\|_\infty \le \bar{\varepsilon}_M$. Let V_G^* be the optimal value over $\mathcal{Z}$ and V_{ref}^* the optimum of a reference class containing those policies under the same state information, stopping, reward and discount conventions. Set $\Delta_{\mathrm{rep}}(s) = V_{\mathrm{ref}}^*(s) - V_G^*(s)$. Suppose, uniformly on the common invariant domain,*

$$\left\| \widehat{V} - \sup_z \widehat{Q}_m(\cdot, z) \right\|_\infty \le \varepsilon_B, \qquad \sup_z \widehat{Q}_m(s,z) - \widehat{Q}_m(s, \widehat{\pi}(s)) \le \varepsilon_{\mathrm{sel}}.$$

For this fixed measurable policy and $0 \le \beta < 1$,

$$0 \le V_{\mathrm{ref}}^*(s) - V_G^{\widehat{\pi}}(s) \le \Delta_{\mathrm{rep}}(s) + \frac{2(\bar{\varepsilon}_M + \varepsilon_B) + \varepsilon_{\mathrm{sel}}}{1-\beta}. \tag{D.1}$$

Proof. Let $\mathcal{T}_G, \mathcal{T}_G^{\widehat{\pi}}$ be the true optimal and policy Bellman operators, and set $\widehat{\mathcal{T}}\widehat{V} = \sup_z \widehat{Q}_m(\cdot, z)$. With $\varepsilon_M = \|Q_{\widehat{V}} - \widehat{Q}_m\|_\infty$ and $e = \varepsilon_M + \varepsilon_B$,

$$\|\mathcal{T}_G \widehat{V} - \widehat{V}\|_\infty \le \|\mathcal{T}_G \widehat{V} - \widehat{\mathcal{T}}\widehat{V}\|_\infty + \|\widehat{\mathcal{T}}\widehat{V} - \widehat{V}\|_\infty \le e.$$

The first inequality uses $|\sup f - \sup g| \le \sup |f - g|$. Contraction of the true Bellman operator gives $\|V_G^* - \widehat{V}\|_\infty \le e/(1-\beta)$. Likewise,

$$\begin{aligned} |\mathcal{T}_G^{\widehat{\pi}} \widehat{V}(s) - \widehat{V}(s)| &\le |Q_{\widehat{V}}(s, \widehat{\pi}(s)) - \widehat{Q}_m(s, \widehat{\pi}(s))| \\ &\quad + |\widehat{Q}_m(s, \widehat{\pi}(s)) - \widehat{\mathcal{T}}\widehat{V}(s)| + |\widehat{\mathcal{T}}\widehat{V}(s) - \widehat{V}(s)| \\ &\le \varepsilon_M + \varepsilon_{\mathrm{sel}} + \varepsilon_B. \end{aligned}$$

Policy-operator contraction gives $\|V_G^{\widehat{\pi}} - \widehat{V}\|_\infty \le (e + \varepsilon_{\mathrm{sel}})/(1-\beta)$. Add the two distances and the representation gap, then use $\varepsilon_M \le \bar{\varepsilon}_M$. Nonnegativity follows from class inclusion. No contraction of the learned model is used. □

For a fixed stochastic selector, replace its selected backup by its expectation and use the corresponding policy Bellman operator. The same proof applies. Persistent planner memory must be included in the state with the relevant domain assumptions. Appendix E.4 supplies a selection-residual bound for the sequence planner under its additional hypotheses.

D.3 Raw controller actions and radial prediction error

Suppress auxiliary coordinates. First consider an exact-normalization domain $\mathcal{U}_{\mathrm{reg}}$ with $N(u) = u/\|u\|$, $N(\mathcal{U}_{\mathrm{reg}}) = \mathcal{Z}$, and a section $\sigma(z) = z \in \mathcal{U}_{\mathrm{reg}}$. In navigation this restricts norms to at least $\epsilon = 10^{-6}$, not merely nonzero norms. On this domain the raw and canonical actions induce the same physical laws, so $Q_{\widehat{V}}^{\mathcal{U}}(s,u) = Q_{\widehat{V}}(s, N(u))$. For the learned raw backup, define

$$\varepsilon_M^{\mathcal{U}} = \sup_{s, u \in \mathcal{U}_{\mathrm{reg}}} |Q_{\widehat{V}}(s, N(u)) - \widehat{Q}_m^{\mathcal{U}}(s,u)|, \qquad \widehat{Q}_{\mathrm{can}}(s,z) = \widehat{Q}_m^{\mathcal{U}}(s, \sigma(z)).$$

Let

$$\varepsilon_{\mathrm{rad}} = \sup_{s, u \in \mathcal{U}_{\mathrm{reg}}} |\widehat{Q}_m^{\mathcal{U}}(s,u) - \widehat{Q}_{\mathrm{can}}(s, N(u))|.$$

If $\|Q_{\widehat{V}} - \widehat{Q}_{\mathrm{can}}\|_\infty \le \bar{\varepsilon}_M^{\mathrm{can}}$, the triangle inequality gives $\varepsilon_M^{\mathcal{U}} \le \bar{\varepsilon}_M^{\mathrm{can}} + \varepsilon_{\mathrm{rad}}$. Applying Theorem D.1 on this raw domain yields

$$V_{\mathrm{ref}}^*(s) - V_{\mathcal{U}}^{\widehat{\pi}_{\mathcal{U}}}(s) \le \Delta_{\mathrm{rep}}(s) + \frac{2(\bar{\varepsilon}_M^{\mathrm{can}} + \varepsilon_{\mathrm{rad}} + \varepsilon_B^{\mathcal{U}}) + \varepsilon_{\mathrm{sel}}^{\mathcal{U}}}{1-\beta}.$$

Canonical calibration must use the fixed representatives $\sigma(z)$; arbitrary raw-radius training samples cannot be substituted without accounting for radial prediction dependence.

Full implemented action domain. Navigation uses $N_\epsilon(u) = u/\max(\|u\|, \epsilon)$. For $0 < \|u\| < \epsilon$, $\|N_\epsilon(u)\| = \|u\|/\epsilon < 1$, and $N_\epsilon(0) = 0$. These are not unit-sphere requests, so the preceding sphere-domain identity cannot simply use $u/\|u\|$ there. On the full raw domain, define $Q^{\mathcal{U}}_{\text{exec}}$ from the actual executor and use $\varepsilon^{\mathcal{U}}_{M,\text{exec}} = \|Q^{\mathcal{U}}_{\text{exec}} - \widehat{Q}^{\mathcal{U}}_m\|_\infty$ directly. The same Bellman proof holds with that error and the representation gap relative to the full implemented action class. Manipulation's fallback maps degenerate requests to a fixed unit direction; its actual map can likewise be used. Extending a sphere coverage or calibration bound to additional executed requests requires covering those requests explicitly, rather than assuming the extension from nonzero input alone.

E Motion Coverage and Model-Based Planning Proofs

This appendix proves Theorem 1 and its coverage and replanning connections. All results use the common metric, integrability and execution conventions of Appendix B.2. Wasserstein propagation follows the same principle as Asadi et al. (2018); policy-error conversion is in Appendix D.2.

E.1 From request fidelity to physical coverage

For a nonempty finite menu $B \subset \mathcal{Z}$, let $h_B = \sup_{z\in\mathcal{Z}} \min_{b\in B} \rho(z, b)$. Fix model-independent physical targets $b^\star_s(q)$ and metric δ. Let $F_s(z, \omega)$ describe the executed prefix, where ω contains the sampler noise and any environment randomness. Assume integrability and, uniformly over the relevant states and requests, a code $z_s(q) \in \mathcal{Z}$ with

$$\mathbb{E}_\omega \delta(F_s(z_s(q), \omega), b^\star_s(q)) \le \varepsilon_{\text{exec}}, \qquad \mathbb{E}_\omega \delta(F_s(z, \omega), F_s(z', \omega)) \le L_F \rho(z, z').$$

The second inequality uses common randomness. Define

$$\kappa(B) = \sup_{s,q} \min_{b\in B} \mathbb{E}_\omega \delta(F_s(b, \omega), b^\star_s(q)).$$

For each fixed (s, q), choose a nearest code $b \in B$ to $z_s(q)$ before drawing ω. The triangle inequality gives

$$\kappa(B) \le \varepsilon_{\text{exec}} + L_F h_B. \tag{E.1}$$

Thus the expectation is taken before minimization; the menu choice cannot adapt to a realized noise draw. Nested menus satisfy $\kappa(B) \le \kappa(A)$ for $A \subseteq B$. A strictly better added request improves that target's error but need not strictly improve the worst-target supremum. Appendix G.9 specifies the finite empirical estimands; distance to a mean endpoint and mean distance to a target are not generally equal.

E.2 Motion-sequence coverage: statement and proof

Fix $T \ge 1$ and a bounded L_v-Lipschitz terminal value v. Define

$$J^v_T(s, \mathbf{z}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \beta^t R_G(S_t, z_t) + \beta^T v(S_T)\right], \qquad \mathbf{z} = (z_0, \dots, z_{T-1}).$$

Requests are open loop within a sequence, while each chunk is generated in its current execution context. Put $\Lambda_t(L) = \sum_{j=0}^{t-1} L^j$, with $\Lambda_0(L) = 0$.

Theorem E.1 (Motion-sequence coverage)**.** *Under Equation 3 and Appendix B.2, for a nonempty finite menu $B \subset \mathcal{Z}$,*

$$0 \le \sup_{\mathbf{z}\in\mathcal{Z}^T} J^v_T(s, \mathbf{z}) - \max_{\mathbf{b}\in B^T} J^v_T(s, \mathbf{b}) \le \mathcal{C}_T h_B, \tag{E.2}$$

where

$$\mathcal{C}_T = \sum_{t=0}^{T-1} \beta^t \big[C_z + C_s L_z \Lambda_t(L_s)\big] + \beta^T L_v L_z \Lambda_T(L_s). \tag{E.3}$$

Proof. For any $\mathbf{z}$, choose a coordinatewise nearest $\mathbf{b} \in B^T$, so $\rho(z_t, b_t) \le h_B$. Let μ_t, ν_t be their state laws from the same initial state, and write $P_z = P_G(\cdot \mid \cdot, z)$. For any coupling λ of those laws, the mixture inequality gives

$$W_1(\mu_t P_{z_t}, \nu_t P_{b_t}) \le \int W_1(P_G(\cdot \mid x, z_t), P_G(\cdot \mid y, b_t))\, d\lambda(x, y).$$

This inequality also follows directly from Kantorovich–Rubinstein duality by bounding each 1-Lipschitz test-function expectation; no pointwise choice of an optimal transition coupling is required. Apply task regularity and take the infimum over λ to obtain

$$e_{t+1} \le L_s e_t + L_z h_B, \qquad e_t = W_1(\mu_t, \nu_t), \quad e_0 = 0.$$

Induction gives $e_t \le L_z h_B \Lambda_t(L_s)$. The expected reward difference is at most $C_s e_t + C_z h_B$, and terminal-value difference at most $L_v e_T$. Discounted summation gives $|J_T^v(s,\mathbf{z}) - J_T^v(s,\mathbf{b})| \le \mathcal{C}_T h_B$. Apply this to arbitrarily near-maximizing sequences and use $B^T \subseteq \mathcal{Z}^T$. □

For $t \ge 1$, $\Lambda_t(1) = t$ and $\Lambda_t(0) = 1$. At $T = 1$ the coefficient reduces to $C_z + \beta L_v L_z$. No condition $\beta L_s < 1$ is needed for finite T.

E.3 Proof of Theorem 1

Proof. Let $\mathbf{b}^*$ maximize the true score over the finite set B^T. Uniform score accuracy and predicted-score suboptimality imply

$$\begin{aligned} J_T^v(s,\mathbf{b}^*) - J_T^v(s,\widehat{\mathbf{b}}) &\le [J_T^v(s,\mathbf{b}^*) - \widehat{J}_T(s,\mathbf{b}^*)] + \varepsilon_{\mathrm{opt}} \\ &\quad + [\widehat{J}_T(s,\widehat{\mathbf{b}}) - J_T^v(s,\widehat{\mathbf{b}})] \le 2\varepsilon_J + \varepsilon_{\mathrm{opt}}. \end{aligned}$$

Add Theorem E.1 to obtain Equation 9. □

For randomized search this holds for each realization with its actual search residual; averaging replaces that residual by its expectation if a uniform bound is not assumed. Search suboptimality is represented explicitly by $\varepsilon_{\mathrm{opt}}$. For a general searched subset $\mathfrak{C}_T \subseteq \mathcal{Z}^T$, use the true coverage gap $\sup_{\mathcal{Z}^T} J_T^v - \sup_{\mathfrak{C}_T} J_T^v$ and approximate maximizers. A larger physical reference class adds its nonnegative sequence-optimum gap under the same convention.

A sufficient sequence-score bound. Suppose a learned kernel $\widehat{P}$ and the true kernel are both defined on a common complete-state metric domain containing all states of both rollouts, with finite first moments. Assume uniform errors $W_1(P_G, \widehat{P}) \le e_P$, $|R_G - \widehat{R}| \le e_R$, and $\|v - \widehat{v}_{\mathcal{S}}\|_\infty \le e_v$, where $\widehat{v}_{\mathcal{S}}$ is a terminal approximation on complete states, with true-kernel/state-reward regularity as above. For identical requests, insert the true kernel applied to the learned state law:

$$W_1(\mu_t P_{z_t}, \widehat{\mu}_t \widehat{P}_{z_t}) \le W_1(\mu_t P_{z_t}, \widehat{\mu}_t P_{z_t}) + W_1(\widehat{\mu}_t P_{z_t}, \widehat{\mu}_t \widehat{P}_{z_t}) \le L_s e_t + e_P.$$

Thus $e_t \le e_P \Lambda_t(L_s)$, reward error is at most $e_R + C_s e_t$, and terminal error at most $e_v + L_v e_T$. Their sum gives a valid uniform score-error bound

$$\varepsilon_J \le \sum_{t=0}^{T-1} \beta^t [e_R + C_s e_P \Lambda_t(L_s)] + \beta^T [e_v + L_v e_P \Lambda_T(L_s)], \tag{E.4}$$

where here ε_J denotes the supremum of the actual sequence-score error. For an implicit latent-state model not identified with a kernel on $\mathcal{S}$, Theorem 1 instead assumes scalar score accuracy directly.

E.4 Finite-horizon planning and replanning

Fix $v = \widehat{V}$, the same frozen model, and the same canonical action domain for the one-step and sequence scores. Define

$$\overline{Q}_T(s,z) = \sup_{z_1,\dots,z_{T-1}} J_T^{\widehat{V}}(s,(z,z_1,\dots,z_{T-1})), \qquad a_T = \|\overline{Q}_T - Q_{\widehat{V}}\|_\infty.$$

For $T = 1$, the tail is empty and $a_1 = 0$. In general a_T includes horizon/value mismatch and the difference between open-loop and feedback continuation; one-step prediction accuracy alone does not make it small. For the first executed request $\widehat{b}_0$, two applications of the uniform comparison inequality give

$$\begin{aligned} \sup_z \widehat{Q}_m(s,z) - \widehat{Q}_m(s,\widehat{b}_0) &\le 2\varepsilon_M + \sup_z Q_{\widehat{V}}(s,z) - Q_{\widehat{V}}(s,\widehat{b}_0) \\ &\le 2\varepsilon_M + 2a_T + \sup_z \overline{Q}_T(s,z) - \overline{Q}_T(s,\widehat{b}_0) \\ &\le 2\varepsilon_M + 2a_T + \mathcal{C}_T h_B + 2\varepsilon_J + \varepsilon_{\mathrm{opt}}. \end{aligned}$$

The last step uses $\sup_z \overline{Q}_T = \sup_{\mathcal{Z}^T} J_T^{\widehat{V}}$, $\overline{Q}_T(s, \widehat{b}_0) \geq J_T^{\widehat{V}}(s, \widehat{\mathbf{b}})$, and Theorem 1. To substitute into the infinite-horizon policy bound, all these residuals must hold uniformly on its invariant domain. Raw-action planning requires its own score/selection residuals and the executed action image in Appendix D.3; the canonical estimate is not transferred by ignoring radial dependence.

E.5 Query refinement and supplementary coverage consequences

Nested menus obey

$$\max_{A^T} J_T^v \leq \max_{B^T} J_T^v \leq \sup_{\mathcal{Z}^T} J_T^v.$$

If a compact query region has covering number $\mathcal{N}(\mathcal{Z}, \rho, h) \leq (C/h)^{d_z}$ on the stated range of $h > 0$, a Cartesian-product cover under the maximum-coordinate metric has at most $(C/h)^{d_z T}$ elements. A bounded full-dimensional family of unrestricted K-step chunks has ambient sequence dimension $K d_a T$. This is a geometric cover-size comparison within the specified families. A subset of the deployed sphere $\mathcal{S}^3$ admits the sphere's at-most-three-dimensional covering bound; separately covered auxiliary coordinates contribute their own factors.

A strict sequence example. Take $q(\theta) = (\cos\theta, \sin\theta)$ for $\theta \in [0, \pi/2]$, $p_{t+1} = p_t + q(\theta_t)$, reward $-\|q(\theta_t) - q(\pi/4)\|$ and zero terminal value. Both anchors $0, \pi/2$ have distance $d_0 = \sqrt{2 - \sqrt{2}}$ from the desired midpoint. The anchor-only optimum is $-d_0 \sum_{t=0}^{T-1} \beta^t$; adding $\pi/4$ achieves zero, giving a strict improvement under the same decision interval and reward.

Infinite-horizon coverage. Assume the common domain is forward invariant, action sets are state independent, rewards are bounded, Equation 3 holds uniformly, and $\beta L_s < 1$. Starting value iteration at zero, a state-Lipschitz value with constant L has optimal Bellman backup constant at most $C_s + \beta L_s L$. The geometric series and uniform value convergence yield

$$\mathrm{Lip}(V_G^*) \leq \frac{C_s}{1 - \beta L_s}, \qquad L_{Q^*,z} \leq C_z + \frac{\beta L_z C_s}{1 - \beta L_s},$$

where $L_{Q^*,z}$ may be chosen as the displayed upper bound. Let $\mathcal{T}_B$ optimize only over B. Rounding a maximizing or approximately maximizing request gives

$$0 \leq V_G^* - \mathcal{T}_B V_G^* \leq L_{Q^*,z} h_B.$$

Using $V_B^* = \mathcal{T}_B V_B^*$ and contraction,

$$0 \leq V_G^*(s) - V_B^*(s), \qquad \|V_G^* - V_B^*\|_\infty \leq \frac{L_{Q^*,z} h_B}{1 - \beta}. \tag{E.5}$$

This additional stability condition is specific to the infinite-horizon Lipschitz bound, not the finite-horizon coverage theorem or the Bellman-residual policy theorem.

Correspondence with experiments. The finite-menu intervention examines added executable requests and fixed-model selection; the fixed-transition study examines predictive coordinates; downstream benchmarks evaluate task utility. Together, these diagnostics examine complementary parts of the execution, prediction and planning mechanism.

F Implementation and Evaluation Protocols

This appendix describes PaLMR data collection, implementation, hyperparameters and execution, together with the evaluation protocols used in Section 5. Algorithm A.1 summarizes the training and deployment workflow. Supporting measurements are grouped separately in Appendix G.

F.1 Domains, interfaces and evaluation contracts

Domains and motion pretraining. We evaluate three PointMaze layouts (U, Medium and Hard) and all fifty installed Meta-World v3 tasks (Yu et al., 2020). Each domain uses one pretrained encoder and motion decoder, shared with frozen weights across its downstream tasks. PointMaze pretraining

Table F.1: **Deployed motion interfaces.** H is the generated horizon and K the executed prefix. Each domain uses its own frozen low level. Task observations are supplied to the high-level controller; the table lists the decoder context.

	PointMaze	Meta-World
Trace / motion dimension	2 / 2	3 / 3
H / chunk dimension	100 / 200	25 / 75
Executed prefix K	up to 100 raw steps	5 raw steps
Code	$z \in \mathcal{S}^3 \subset \mathbb{R}^4$	$z \in \mathcal{S}^3 \subset \mathbb{R}^4$
Context	velocity, $\mathbb{R}^2$	hand position, aperture, previous ΔEE, $\mathbb{R}^7$
Auxiliary actuation	none	constant or linear gripper profile
High-level action box	$[-2, 2]^4$	$[-1, 1]^5$ constant; $[-1, 1]^6$ linear
Physical action cap	norm ≤ 0.25, unit gain	coordinate clip to $[-1, 1]$
High-level learner	project TD-MPC2-style	official TD-MPC2

collects open-field burst–coast motions in eight directions and three distances, including resting and moving starts; the data contain 5399 TRAIN and 540 validation chunks. Robot pretraining collects free end-effector motion in `reach-v3`, using 26 directions and three distances with 9984 TRAIN and 1248 validation chunks. Collection descriptors organize support and held-out queries, while codes are learned from realized traces and actions. Appendix F.2 specifies collection, preprocessing and query construction.

Task learning and comparisons. Each task learns its own high-level controller, with three independent online-policy seeds for every compared method. Navigation generates and executes $H = K = 100$ actions per request, while manipulation generates $H = 25$ and executes a $K = 5$ prefix; native termination can end either execution early. PaLMR uses the interfaces in Table F.1; robot results use the linear gripper profile. Matched navigation macro controls share the high-level learning core and execution schedule. Native algorithm comparisons retain their objectives, model architectures and update schedules. All generative baselines use the same recorded local-motion collection. Comparison methods are described in Appendix F.4, and PaLMR hyperparameters are given in Appendix F.5.

Budgets and evaluation. Online budgets count actual simulator calls: executing K primitive actions consumes K interactions. Offline collection and pretraining are accounted separately. PointMaze uses 2M training calls, forty ten-episode periodic evaluations and a separate fifty-episode final evaluation per seed. Actual counters and curve aggregation are specified in Appendix F.6. The primary robot comparison uses the common observed 50k–200k prefix, selected retrospectively from the available curves, with all fifty tasks, seven methods and three seeds represented at four checkpoints. Each periodic evaluation uses ten episodes per seed. A separate exploratory endpoint analysis applies the same PFM-informed budget to every method within each task: 250k/500k/750k/1M for $27/8/2/13$ tasks, with fifty final episodes per seed. Appendix G.1 specifies the outcome-informed stopping rule and task registry. Physical PFM tests use five fixed sampling noises per query, conditional on one low-level training seed per domain. Predictor seeds vary supervised model fitting, physical flow draws vary execution from a fixed decoder, and online-policy seeds vary downstream RL; these are distinct evaluation units. Periodic learning curves and independent final-fifty evaluations are not interchangeable. The main figure's task selection and common display windows are specified in Appendix G.2.

Experimental questions and controlled comparisons. The experiments progress from whole-system performance to three mechanisms. **Q1** asks whether the complete interface improves downstream task learning (Section 5.1). **Q2** tests whether coordinate organization changes predictive accuracy when physical records, supervision and predictor budget stay fixed (Section 5.2); its learned-encoder contrast is conditional on recorded geometry-PFM transitions. **Q3** tests whether additional executable requests improve coverage and can be selected by fixed models, changing the allowed request menu while preserving the task and motion modules (Section 5.3). **Q4** tests predictability, interpolation and state adaptation by changing the 2D decoder while fixing the geometry-informed encoder and requested codes (Section 5.4). These controlled interventions examine individual mecha-

nisms alongside the complete-system comparison. Benchmark summaries include all fifty tasks, and frozen-planner summaries include all thirty paired episodes.

F.2 Reward-free collection and held-out construction

The canonical PointMaze library D1 contains 5399 TRAIN and 540 validation executed chunks. Collection uses eight directions at 45° intervals and distances $\{0.3, 0.6, 0.9\}$ m over 100 actions, with an L_2 cap of 0.25, under burst–coast feedback control. Half the starts are at rest and half draw each velocity component uniformly from $[-0.05, 0.05]$. Starting XY is uniform on $[-3, 3]^2 \setminus [-1.25, 1.25]^2$; the held-out start panel lies inside $[-0.6, 0.6]^2$. All 5939 collection attempts passed the fixed acceptance criteria. Pretraining uses these task-reward-free physical motions.

The encoder uses pre-action positions $p_0, \ldots, p_{99}$; endpoint evaluation uses p_{100}. The soft-DTW target uses stride-four indices $0, 4, \ldots, 96$. Direction and distance descriptors specify collection support and evaluation targets, while the learned code is inferred from recorded relative motion and actions. Training-cycle queries use spherical interpolation. Runtime mixtures use normalized weighted sums of TRAIN anchor codes, with a mixture failure recorded when the unnormalized norm is below 10^{-3}. Held-out targets are specified independently of the encoder.

F.3 Implementation and execution protocol

Encoder and decoder. The encoder is a bidirectional transformer with width 128, three layers, four heads, feedforward width 512, dropout 0.1 and GELU. It reads interleaved projected position/action tokens with sinusoidal positions (200 tokens for $H = 100$, 50 for $H = 25$), using its final token for four-dimensional mean and log-variance heads. The deployed code is the normalized mean. Training samples $\mathrm{normalize}(\mu + \sigma \odot \epsilon)$ with each standard deviation capped at 0.5. A two-hidden-layer width-256 SiLU reconstruction head takes $[z, k/(H-1)]$. The flow is an eight-layer width-512 feedforward network with 32-dimensional time and 128-dimensional code embeddings; q_a has two width-256 hidden layers. The 2D encoder trains for 2000 epochs (44,000 updates), batch 256, Adam 10^{-4} and gradient clip 1. The flow trains for 10^6 updates with the same optimizer settings. Actions are flattened and normalized using TRAIN min/max, with range floor 10^{-6}. The 2D context is the two-dimensional velocity slice [2:4].

Active objectives and calibration. Reconstruction MSE averages over batch, time and channel. Hinge pairs are formed within each batch by pairing row i with $i-1$ modulo batch size. The implementation uses $\rho = \sqrt{\|z_i - z_j\|^2 + 10^{-8}}$ and detached $d_\tau = \sqrt{\max(\mathrm{softDTW}_\gamma(b_i, b_j), 0) + 10^{-8}}$, with $\gamma = 10^{-5}$. The $H = 100$ traces are downsampled by four; $H = 25$ traces use every position. Epochs 1–50 optimize reconstruction. At epoch 51, 4096 nonidentical TRAIN pairs calibrate $m = 0.8 q_{0.05}(\rho/d_\tau)$ and $M = 1.25 q_{0.95}(\rho/d_\tau)$, fixed thereafter. Values are $m = 0.3198, M = 1.1544$ in 2D and $m = 1.943, M = 6.827$ in 3D. The active encoder objective is reconstruction plus the two unsquared hinges. The cross-context term is inactive in both domains.

Flow auxiliaries and sampling. Let G_4 be the differentiable four-step unclipped surrogate. The auxiliary branch uses the first 64 entries every fourth update:

$$\mathcal{L}_{\mathrm{distill}} = \mathbb{E}\|q_a(c, y) - z\|^2, \qquad \mathcal{L}_{\mathrm{cycle}} = \mathbb{E}\|q_a(c', G_4(c', z_\alpha, \xi)) - z_\alpha\|^2.$$

Here z_α spherically interpolates two codes at $\alpha \sim \mathcal{U}(0, 1)$ and c' is shuffled context. With $z_b = \mathrm{normalize}(z_\alpha + \delta)$, $\|\delta\| = 0.1$, the sensitivity term is

$$\mathcal{L}_{\mathrm{sens}} = \mathbb{E}\left(\frac{\|G_4(c, z_b, \xi) - G_4(c, z_\alpha, \xi)\|}{\max(\|z_b - z_\alpha\|, 10^{-6})} - \sqrt{H d_a}\right)_+ .$$

Writing ℓ for the decoder-update index, the complete objective is

$$\mathcal{L}_{\mathrm{flow}} = \mathcal{L}_{\mathrm{FM}} + \mathbb{1}[\ell \bmod 4 = 0]\big(\mathcal{L}_{\mathrm{distill}} + \tfrac{1}{2}\mathcal{L}_{\mathrm{cycle}} + \tfrac{1}{10}\mathcal{L}_{\mathrm{sens}}\big). \tag{F.1}$$

Deployment uses ten Euler steps, per-step clipping to $[-3, 3]$, TRAIN denormalization and the physical action cap, with fresh noise per request. Low-level training uses seed zero. Five physical-evaluation noises are repeated sampler draws from the corresponding fixed checkpoint.

Table F.2: Locked request construction. Held-out physical targets are specified independently of encoded test trajectories.

Axis	Anchors / weights	Physical target	Queries
Direction	adjacent θ_j, θ_{j+1}, $r = 0.6$, equal weights	$r = 0.6$, shortest-arc angle $\theta_j + 22.5^\circ$	$8 \times 8 = 64$
Distance	same θ_j, adjacent radii $(0.3, 0.6)$ or $(0.6, 0.9)$, equal weights	$r = 0.45$ or 0.75	$16 \times 8 = 128$
Start	one anchor for each of 24 modes	anchor's realized full-H relative displacement at translated start	$24 \times 8 \times 2 = 384$

PointMaze task and accounting. The environments are `PointMaze_{UMaze,Medium,Large}-v3`, with sparse reward, `continuing_task=False`, `reset_target=False` and a 6000-raw-step limit. Observations are $(x, y, v_x, v_y, g_x, g_y)$ and actions have norm cap 0.25. Reward is one within goal distance 0.45, followed by termination, and zero otherwise. A macro transition stores its summed raw rewards and true final observation. Discount 0.99 is applied once per decision, with bootstrapping masked by true termination. Time-limit truncation retains bootstrapping; remaining training budget shortens the last chunk. A collision alone leaves the chunk running. Primitive and macro configurations retain their respective feedback intervals and raw-time discount interpretations.

The first training reset uses the run seed; subsequent resets use native randomness, including resampled start/goal positions. Evaluation stops at termination or truncation, records any native success and preserves training RNG state. Final episode j uses reset seed $60000 + s + j$ for online seed s. The macro learner uses 256 candidates, horizon three, three refinement iterations and temperature 0.5, executing the best sampled candidate's first request. Batch size is 256. Updates begin after $10{,}000$ raw warm-up calls and a full replay batch, at one update per two macro decisions. Navigation uses epsilon-clamped code normalization at 10^{-6}; manipulation maps a degenerate direction to a fixed unit code.

Locked 2D queries. For each required mode, one valid resting TRAIN row is selected from a sorted index pool with a fixed seed and encoded with the frozen mean. Runtime mixtures normalize their weighted code sum. Seen XY and held-out XY use a separate fixed seed on the regions stated above. The 576 resting queries are paired with the downstream-visited-velocity intervention specified in Appendix G.13.

Deterministic controls. The regression decoder predicts action sequences from the same frozen request codes and velocity context using action MSE. In the decoder comparison, it shares the flow's data, normalization and training budget, with approximately matched capacity. Action blending applies the physical action cap to stored actions and combines them using fixed request weights, independently of the current velocity.

Robot collection and gripper profiles. The single `reach-v3` collection contains 78 modes: normalized $v \in \{-1, 0, 1\}^3 \setminus \{0\}$ crossed with radii $\{0.03, 0.06, 0.09\}$ m. Each contributes 128 TRAIN and 16 validation accepted chunks, giving $9984/1248$. The encoder trains for 2000 epochs ($78{,}000$ updates). Starts lie in $[-0.20, 0.20] \times [0.55, 0.75] \times [0.15, 0.30]$ m, excluding a cube of half-width 0.045 centered at $(0.10, 0.65, 0.225)$; held-out queries use the inner cube of half-width 0.02. Half the accepted rows per mode are settled and half moving. Moving starts use a twelve-call motion-preparation prefix, accounted for in preparation cost. Collection gripper commands $\{-1, 0, 1\}$ are constant within chunks, with 3328 TRAIN chunks per value. The motion encoder and flow use the three motion channels.

Robot context consists of hand position, gripper aperture and the previous raw-step end-effector displacement. The constant-gripper interface has five high-level coordinates; the linear interface has six. Over the nominal five-step schedule, the gripper is either constant or $g[t] = (1 - t/(K-1))\,\mathrm{clip}(g_{\mathrm{start}}) + (t/(K-1))\,\mathrm{clip}(g_{\mathrm{end}})$. A shortened decision executes the corresponding prefix. The decoder generates twenty-five motion actions and the executor applies the first five.

F.4 COMPARISON METHODS

The main comparison contains PaLMR and six baselines, spanning primitive-action control and reinforcement learning with pretrained generative policies. Every method uses three independent online RL seeds per task. All high-level controllers receive task observations; shared data and evaluation protocols are specified in Appendix F.1.

PaLMR. PaLMR supplies a frozen context-conditioned PFM motion interface to model-based RL. PointMaze uses the project's TD-MPC2-style learning core with TRAIN-fitted proposal mixing; Meta-World uses official TD-MPC2 (Hansen et al., 2024) and the linear gripper profile. Low-level weights are reused across tasks within each domain, while downstream prediction and control are learned from task rewards.

Primitive-action baselines. Flat TD-MPC2 (Hansen et al., 2024) learns a task model and plans directly over primitive controls without the pretrained motion interface. The PointMaze implementation is labeled *Flat TD* in the figures; Meta-World uses official TD-MPC2. Flat SAC (Haarnoja et al., 2018) is the project's off-policy maximum-entropy actor–critic operating on primitive actions, also without motion pretraining. These controls compare the complete motion-interface construction with direct motor-action learning.

Latent steering. DSRL-SAC (Wagenmaker et al., 2025) learns a SAC controller over the noise input of a frozen pretrained diffusion policy. The steered generator produces an action chunk, in contrast to PaLMR's behavior-grounded request coordinates. DSRL-NA is an additional navigation comparison reported in the appendix, not one of the seven systems in the main figure.

Generative-policy optimization. DPPO (Ren et al., 2025) fine-tunes a diffusion policy using policy gradients through its denoising process. ReinFlow (Zhang et al., 2025) introduces learnable noise into the flow sampling chain to enable likelihood-based online policy optimization. FPO++ (Yi et al., 2026) uses a flow-matching-based policy-gradient objective without requiring tractable policy likelihoods. These methods update generative-policy parameters under downstream reward, whereas PaLMR freezes its motion generator. All use the same domain-specific collection of local motions for pretraining and retain their native learning algorithms.

Online interaction counts use primitive environment calls. The navigation macro-interface controls share the high-level learning core and execution schedule. Appendix F.5 gives PaLMR's hyperparameters and offline costs.

F.5 PALMR HYPERPARAMETERS AND OFFLINE COST

PointMaze high-level learner. PaLMR uses the project's TD-MPC2-style learner with width-256 networks, Adam 3×10^{-4}, batch 256, replay capacity 200,000, 10,000 raw-call warm-up and one update per two decisions after a full batch is available. Planning uses horizon three, 256 candidates, three refinement iterations, temperature 0.5 and the best sampled candidate. Discount is 0.99 per decision.

Robot high-level learner. PaLMR uses the size-five TD-MPC2 configuration with latent dimension 512 and feedforward hidden width 512, a two-layer width-256 encoder, five Q functions, SimNorm groups of eight, 101 value bins on $[-10, 10]$, batch 256, learning rate 3×10^{-4}, encoder rate multiplier 0.3, gradient clip twenty and target rate 0.01. Planning uses horizon three, 512 samples, 64 elites, six iterations, twenty-four policy trajectories and temperature 0.5. Discount is 0.95 per $K = 5$ decision. Training uses 5000 raw warm-up calls, the native startup update burst and one subsequent update per decision. Replay capacity follows $\min(\texttt{buffer_size}, \texttt{cfg.steps})$: the original four-task runs planned 2M primitive calls and use 400k decision rows; the remaining runs planned 1M and use 200k rows. Each task trains an independent high-level policy with the same frozen motion interface.

PFM samples with ten Euler steps. In Meta-World, the high-level controller additionally selects two gripper endpoints for the linear profile in Appendix F.3.

Table F.3: PFM conditioning and decoder capacity. H/K gives generated/executed primitive actions per decision.

Domain	Condition	Output	H/K	Parameters
PointMaze	z_4, velocity$_2$	motion$_2$	$100/100$	3,476,008
Meta-World	z_4, context$_7$	motion$_3$	$25/5$	3,350,443

Table F.4: PaLMR training-loop elapsed hours on an RTX 2080 Ti. Each flow receives one million updates at batch 256; encoder update counts are listed separately. The trained interface is reused across downstream tasks and seeds.

Stage	PointMaze hours	Meta-World hours
Encoder (44k / 78k updates)	1.207	1.305
PFM flow	10.173	9.694

Offline collection cost. Robot collection attempted $14{,}503$ chunks, accepted $11{,}232$ and rejected 3271 (22.55%). It used $1{,}531{,}831$ raw environment calls: $1{,}169{,}256$ preparation calls and $362{,}575$ motion calls. This count includes rejected attempts and moving-start preparation, but excludes internal reset simulation, preliminary calibration and pilot searches. Offline collection and pretraining are separate from downstream online interaction budgets.

Encoder and flow training use Adam 10^{-4} with gradient norm clip one. Their architectures, normalization, objectives and sampling procedure are specified in Appendix F.3.

Motion collection controllers. For PointMaze, clamp the controller request $u^{\mathrm{dir}} \in [-2, 2]^2$, set $\theta = \pi u_0^{\mathrm{dir}}/2$ and $r = 0.3 + 0.6(u_1^{\mathrm{dir}} + 2)/4$, with direction $(\cos\theta, \sin\theta)$. Sixteen-iteration free-flight bisection selects initial target speed. Burst–coast feedback gain is twenty-five and the norm cap is 0.25. The hundred-action sequence uses predicted velocity from measured mass, damping, gear and timestep.

For the robot, normalize the first three controller coordinates, defaulting zero to $+x$, and map the fourth to $r \in [0.03, 0.09]\,\mathrm{m}$. Select $j = \arg\min_j \|rd - r_j d_j\|$ over seventy-eight calibrated modes. With compensation c_j, initial mocap position m_0 and offset $b_0 = p_{\mathrm{EE}} - m_0$, set $m_1 = p_{\mathrm{EE}} + rd + c_j - b_0$ and

$$w_k = m_0 + (m_1 - m_0)J(\min((k + 1 + 8)/20, 1)), \qquad J(x) = 10x^3 - 15x^4 + 6x^5.$$

Clip w_k to the box spanning m_0, m_1 enlarged by $0.05\,\mathrm{m}$, then apply $\mathrm{clip}((w_k - m_{\mathrm{current}})/\texttt{action_scale}, -1, 1)$. Collection measures the offset after forty settling calls and holds the gripper command constant within each chunk. Compensation uses three additive rounds of six TRAIN-support rollouts per mode at seed zero, independent of reward and held-out targets. Table F.5 lists the compensation coefficients.

F.6 Learning-curve counters, evaluation panels and displayed tasks

Recorded evaluation schedules. The original logs contain forty periodic evaluations per PointMaze policy, triggered at nominal 50k-step intervals through 2M primitive training calls. Flat TD and flat SAC hit these thresholds exactly. Macro-action policies finish their current chunk before evaluating, giving recorded counters between the threshold and 99 calls above it; every run ends at exactly 2M. All seven main-figure methods, three layouts and three seeds have complete histories. These counters exclude evaluation rollouts.

For the displayed manipulation tasks, all seven methods and all three seeds record periodic evaluations at exactly every 50k primitive training calls. Within their task budgets, stick-push has fifteen observations through 750k, peg-insert-side ten through 500k, and lever-pull five through 250k. Assembly and shelf-place have twenty through 1M, except PaLMR and flat TD-MPC2, whose last periodic evaluation is at 950k. Every periodic value averages ten episodes, and each separate final panel averages fifty episodes.

Curve and uncertainty conventions. Figures 3, G.1, G.2 and G.3 use every recorded periodic observation in the stated display range, without smoothing, interpolation or extrapolation. The mean line pairs the three observations at the same nominal evaluation index and uses the mean of their actual counters as its horizontal coordinate; shading spans their minimum and maximum success rates. For manipulation the three counters coincide. Straight segments connect these observations. The fifty-task panel averages tasks with equal weight within each seed index before taking the mean and range of the three seed-index aggregates. All curves start at their first recorded evaluation; no zero-step value is plotted.

Periodic and final evaluation. The periodic endpoint and independent final-50 panel are stored and analyzed separately. For native navigation algorithms, final optimization can occur after the periodic evaluation at 2M and before the independent final evaluation. Figure 3 uses periodic observations throughout; task-result tables identify the independent final panel.

Task selection and full budgets. Appendix G.2 describes the selection and display windows of the Meta-World tasks in Figure 3. The three manipulation examples in Figure G.3 were chosen after inspecting all fifty task curves to illustrate learning behavior across tool-mediated pushing, insertion and articulated-object interaction. Every method and seed is observed through the same within-task budget. Table G.1 and the complete task registry retain all fifty tasks. The final-budget column uses the same $27/8/2/13$ tasks at 250k/500k/750k/1M, with the PFM-informed budget rule in Appendix G.1. Offline collection and pretraining costs are separate from the online axes and appear in Appendix F.5.

F.7 Robot collection-controller calibration

The following fixed compensation values specify the nearest-mode controller in Appendix F.5. The integer direction tuple is normalized to a unit vector before multiplying by the radius. Radii and compensation components are in millimeters; calibration values are rounded only at the displayed 10^{-6} mm precision. The table is shared across downstream tasks and was fitted without their rewards.

Table F.5: Compensation coefficients for the 78 robot motion-collection modes.

Direction tuple	Radius	c_x	c_y	c_z
(-1,-1,-1)	30	0.197687	-0.458430	-0.059923
(-1,-1,-1)	60	0.525744	-1.237728	0.025172
(-1,-1,-1)	90	0.483121	-1.500697	-0.386457
(-1,-1,0)	30	0.288160	-0.618473	-0.078096
(-1,-1,0)	60	0.236244	-1.124941	-0.008390
(-1,-1,0)	90	0.578911	-1.116619	0.083233
(-1,-1,1)	30	-0.002545	-0.549331	0.028948
(-1,-1,1)	60	0.487524	-0.884359	0.233118
(-1,-1,1)	90	0.324333	-1.167709	0.752865
(-1,0,-1)	30	-0.134820	0.253650	0.032670
(-1,0,-1)	60	-0.315916	0.629061	0.111800
(-1,0,-1)	90	-0.494093	1.180637	-0.018492
(-1,0,0)	30	-0.098387	0.516402	-0.059754
(-1,0,0)	60	-0.436520	1.149543	0.040302
(-1,0,0)	90	-0.717889	2.336292	-0.212560
(-1,0,1)	30	-0.052739	0.255911	-0.032966
(-1,0,1)	60	0.099760	0.320394	0.218965
(-1,0,1)	90	-0.234131	1.159319	0.367259
(-1,1,-1)	30	-0.614019	1.279266	0.045593
(-1,1,-1)	60	-1.089805	2.864575	-0.052970
(-1,1,-1)	90	-1.537508	10.555731	-1.561802
(-1,1,0)	30	-0.808035	2.396146	-0.149423
(-1,1,0)	60	-0.433997	4.133491	-0.117674
(-1,1,0)	90	-3.753197	12.385482	-1.362535

Direction tuple	Radius	c_x	c_y	c_z
(-1,1,1)	30	-0.390319	1.585347	-0.006345
(-1,1,1)	60	-1.269086	3.977101	0.064693
(-1,1,1)	90	-1.081308	5.287665	0.132590
(0,-1,-1)	30	0.298705	-1.551685	0.013696
(0,-1,-1)	60	0.723629	-2.205165	-0.138499
(0,-1,-1)	90	1.106709	-2.802058	-0.355747
(0,-1,0)	30	0.277436	-0.979242	-0.142430
(0,-1,0)	60	0.723169	-2.497889	0.018820
(0,-1,0)	90	0.882999	-2.746818	-0.534209
(0,-1,1)	30	0.316879	-1.218042	0.198780
(0,-1,1)	60	0.488955	-2.001205	0.571182
(0,-1,1)	90	0.499150	-1.273441	0.649293
(0,0,-1)	30	-0.034138	-0.056924	0.053420
(0,0,-1)	60	-0.450442	0.046218	0.052164
(0,0,-1)	90	-0.610735	0.174246	-0.114813
(0,0,1)	30	0.060416	0.062864	-0.008866
(0,0,1)	60	0.272566	-0.449339	0.667811
(0,0,1)	90	0.458505	-0.131083	1.141709
(0,1,-1)	30	-0.316036	1.517597	0.004549
(0,1,-1)	60	-0.920279	5.257652	-0.389419
(0,1,-1)	90	-1.646021	8.346269	-0.654514
(0,1,0)	30	-0.485248	1.395375	0.094753
(0,1,0)	60	-1.390920	4.707016	-0.215792
(0,1,0)	90	-2.297009	12.320577	-0.489868
(0,1,1)	30	-0.381496	1.627371	-0.012887
(0,1,1)	60	-0.586260	2.813856	0.013656
(0,1,1)	90	-1.067704	7.638925	-0.036321
(1,-1,-1)	30	0.166011	-1.124770	-0.000999
(1,-1,-1)	60	0.443561	-2.360724	-0.107629
(1,-1,-1)	90	0.062104	-3.929435	-0.161345
(1,-1,0)	30	0.252840	-1.535544	0.033204
(1,-1,0)	60	0.227226	-2.818734	0.252745
(1,-1,0)	90	0.644873	-2.771691	-0.075137
(1,-1,1)	30	0.247338	-1.258676	0.103392
(1,-1,1)	60	0.403244	-1.796971	0.399477
(1,-1,1)	90	0.388855	-1.925051	0.400023
(1,0,-1)	30	0.063792	-0.485116	0.058658
(1,0,-1)	60	0.067758	-0.689549	0.083836
(1,0,-1)	90	-0.324198	-0.803448	0.383562
(1,0,0)	30	0.001991	-0.419393	0.044903
(1,0,0)	60	-0.046893	-0.606593	0.078558
(1,0,0)	90	0.284646	-0.653707	0.077787
(1,0,1)	30	0.147531	-0.113399	-0.031069
(1,0,1)	60	0.141922	-0.135320	0.087428
(1,0,1)	90	0.004721	-1.544062	0.838921
(1,1,-1)	30	-0.095328	1.142208	0.049061
(1,1,-1)	60	-0.438156	2.926323	-0.204298
(1,1,-1)	90	-0.243967	4.187366	-0.018370
(1,1,0)	30	-0.115773	1.240495	-0.063688
(1,1,0)	60	-0.203058	2.652401	0.076067
(1,1,0)	90	-0.412154	4.383275	0.034866
(1,1,1)	30	0.028959	1.578725	-0.133705
(1,1,1)	60	-0.435969	1.379863	0.048893
(1,1,1)	90	-0.577181	4.900426	0.145618

The production collection uses at most 80 P-controller positioning calls with a 2 mm stopping threshold, followed by 40 settling calls and an optional 12-call moving prefix. Its fixed moving-prefix calibration lookup records dimensionless action magnitude, final end-effector displacement per raw step in millimeters, and total displacement over the twelve-step prefix in millimeters. Reported collection-preparation totals include the moving-prefix calls.

Action magnitude	Final ΔEE/step (mm)	Prefix displacement (mm)
0.050	0.470	4.000
0.100	0.890	7.900
0.200	1.880	16.200
0.300	2.920	25.500
0.500	4.920	42.400
0.700	6.730	59.200
0.900	8.690	76.800
1.000	9.460	80.900

G Supplementary Experimental Results

Results follow the main experimental questions: downstream task learning, fixed-transition predictive coordinates, planning with added motion requests, and physical execution. Each diagnostic retains its own sample unit and evaluation population.

G.1 Fifty-task manipulation results and budget registry

All fifty goal-observable tasks, seven methods and three independent online RL seeds per method and task contribute to Table G.1. It reports task-balanced success at the common 200k checkpoint, normalized AUC over 50k–200k and final success at matched per-task budgets. Figure G.1(a) gives the all-task learning curve, and Appendix G.6 reports benchmark differences using unrounded means.

Table G.1: **All fifty Meta-World tasks, three online seeds per method.** Common-budget columns use periodic ten-episode evaluations: success at 200k steps and normalized trapezoidal AUC over 50k–200k. The final column uses independent fifty-episode evaluations at PFM-informed task budgets, identical across methods within each task (Table G.2). Entries are equal-weight means over tasks and seeds, in percent and rounded to one decimal; bold marks the highest mean.

	Common online budget		Task-specific budget
Method	Success at 200k ↑	AUC, 50k–200k ↑	Final success ↑
PaLMR	**79.7**	**71.2**	**93.7**
Flat TD-MPC2	67.9	57.8	82.4
Flat SAC	40.3	32.0	44.5
DSRL-SAC	24.7	17.2	33.1
DPPO	6.7	4.4	6.9
ReinFlow	7.1	5.2	7.3
FPO++	2.7	2.5	2.9

The exploratory task budgets were set after inspecting PFM learning. Candidate endpoints 250k, 500k and 750k require three consecutive periodic evaluations ending there, each with three-seed mean at least 90% and no seed below 80%, plus current PFM mean at least 90% and no seed below 80% in the available final-fifty panel or each of the latest three common periodic evaluations of an unfinished run. The earliest qualifying endpoint is selected; otherwise the task uses 1M. This PFM-outcome-informed rule gives 27/8/2/13 tasks at the four budgets, applied identically across methods within each task. Every final cell uses fifty episodes per seed. Low-level weights, gain and horizon are shared across tasks. The seven-method robot cohort includes DSRL-SAC; DSRL-NA belongs to the maze comparison.

Table G.2: All fifty manipulation tasks and their matched final-evaluation budgets in thousands of primitive interactions. Each budget applies to all seven systems and three online seeds for that task. Aggregate numerical results are reported in Table G.1.

Task	Budget (k)
assembly	1000
basketball	1000
bin-picking	500
box-close	250
button-press-topdown	250
button-press-topdown-wall	500
button-press	250
button-press-wall	250
coffee-button	250
coffee-pull	500
coffee-push	1000
dial-turn	1000
disassemble	1000
door-close	250
door-lock	250
door-open	250
door-unlock	250
drawer-close	250
drawer-open	1000
faucet-close	250
faucet-open	250
hammer	250
hand-insert	500
handle-press-side	250
handle-press	250
handle-pull-side	250
handle-pull	250
lever-pull	250
peg-insert-side	500
peg-unplug-side	250
pick-out-of-hole	1000
pick-place	750
pick-place-wall	1000
plate-slide-back-side	250
plate-slide-back	250
plate-slide-side	250
plate-slide	250
push-back	1000
push	500
push-wall	1000
reach	250
reach-wall	250
shelf-place	1000
soccer	1000
stick-pull	1000
stick-push	750
sweep-into	250
sweep	500
window-close	250
window-open	500

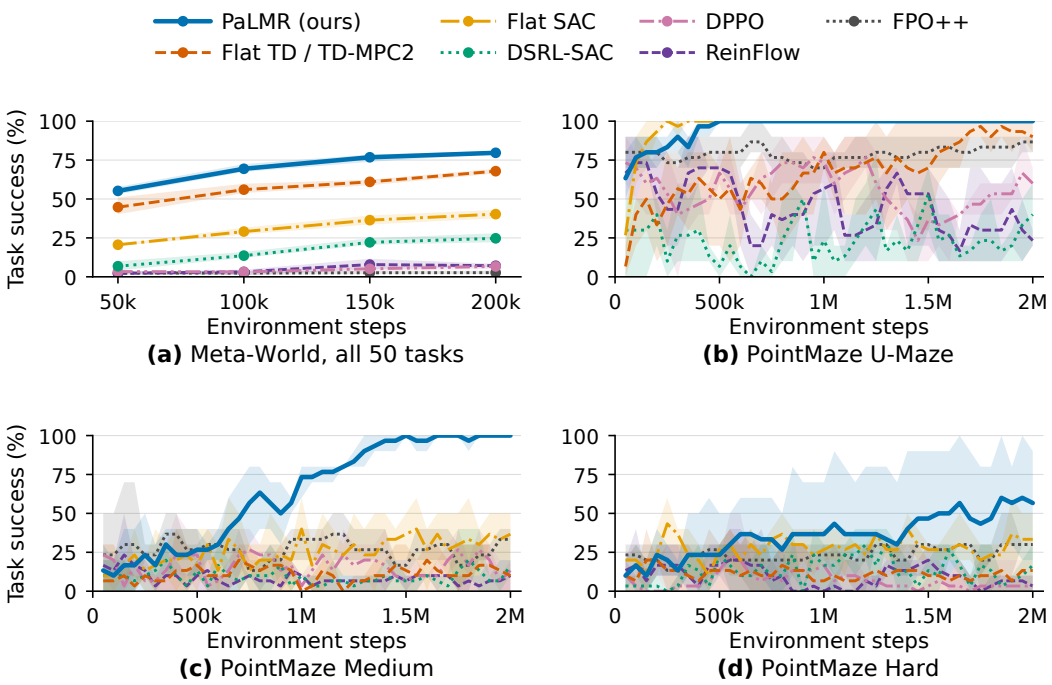

Figure G.1: **All fifty Meta-World tasks and complete PointMaze training.** (a) Task success over all fifty Meta-World tasks at the four recorded common-budget evaluations; tasks are averaged with equal weight within each RL seed index before aggregation. (b–d) PointMaze periodic success over 2M environment steps; each ordinal evaluation is placed at the mean of the three seeds' actual step counters. Lines show the mean and bands the minimum–maximum range over three RL seeds. Periodic evaluations use ten episodes.

G.2 Learning-curve overview and main-figure display windows

Figure G.1 shows the task-balanced curves over all fifty Meta-World tasks at the four common-budget evaluations and all three PointMaze layouts over the full 2M steps. The Meta-World tasks in Figure 3 were chosen after screening all fifty periodic curves, for a visible PaLMR learning advantage and distinct multi-stage operations. These examples complement the fifty-task summary in Table G.1.

Display windows apply to every method within a panel. The maze windows were fixed from the recorded PaLMR curves before plotting. U-Maze and Medium end at the nominal 600k and 1.6M evaluations; PaLMR periodic success is 100% from 500k and at least 93.3% from 1.35M, respectively. These windows include macro-action counters up to 99 steps past the threshold. Hard keeps the full 2M because its mean is still changing. The Meta-World windows span the original task budgets, 1M for assembly and shelf place and 750k for stick push, so the records after PaLMR's plateau are shown. Each line ends at its method's last recorded periodic evaluation in the window. PaLMR and flat TD-MPC2 record no periodic evaluation at 1M for assembly and shelf place, so their lines end at 950k; no value is carried forward. At its last recorded evaluation, PaLMR's periodic mean is 100% on all three tasks. The highest baseline mean at its last recorded evaluation is 3.3% (DSRL-SAC, 1M) on assembly, 10.0% (flat TD-MPC2, 950k) on shelf place and 66.7% (flat TD-MPC2, 750k) on stick push. These are ten-episode periodic values, not final-fifty results, and training budgets are unchanged.

Figure G.2 marks every recorded evaluation of the three tasks through their task budgets. On stick push, flat TD-MPC2 rises from 33.3% at 500k to 66.7% from 600k to 750k, while PaLMR stays between 96.7% and 100% over the same evaluations.

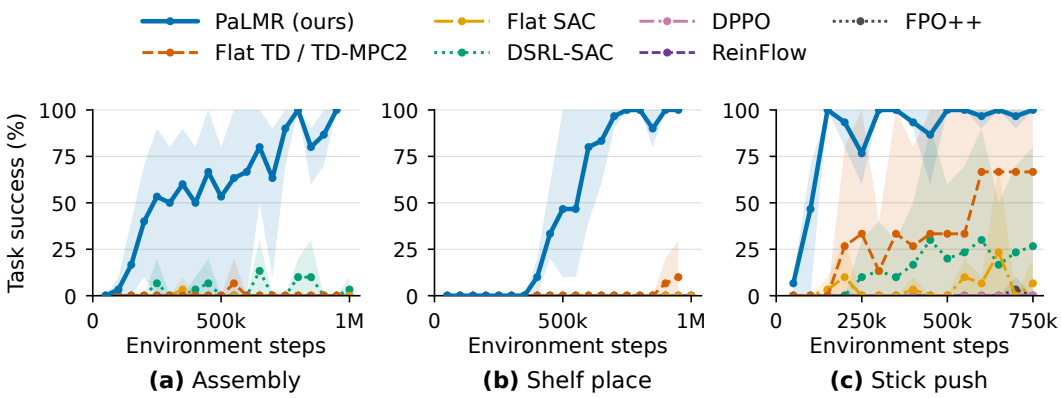

Figure G.2: **Main-figure Meta-World tasks through their task budgets.** Periodic ten-episode success of all seven systems on assembly (1M steps), shelf place (1M) and stick push (750k), with markers at every recorded 50k evaluation. These are the records shown in Figure 3(d–f); lines end at each method's last recorded evaluation. Lines show the mean and bands the minimum–maximum range over three RL seeds.

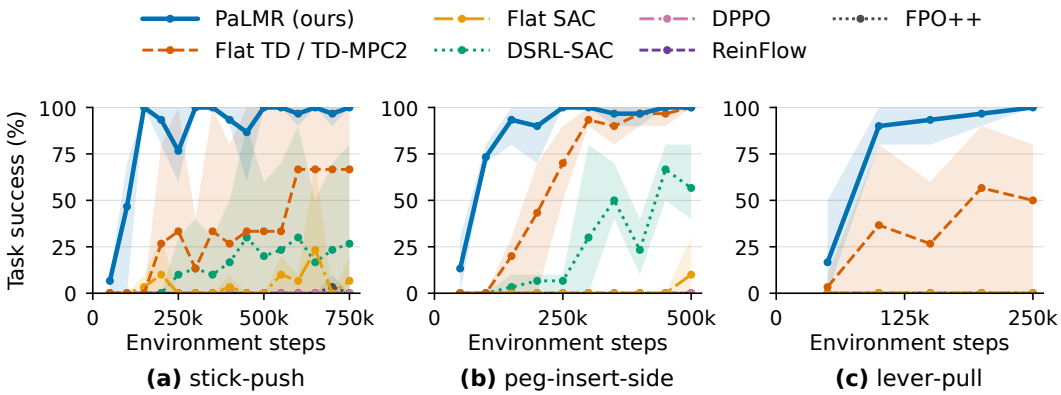

Figure G.3: **Additional Meta-World learning curves.** Periodic ten-episode success of all seven systems on stick-push (750k steps), peg-insert-side (500k) and lever-pull (250k). Lines show the mean and bands the minimum–maximum range over three RL seeds, with markers at every recorded 50k evaluation. These budgets are the task-specific budgets of Table G.1.

G.3 Additional manipulation examples

Figure G.3 presents task-level learning curves for stick push, peg insertion and lever pulling at their complete matched budgets. Appendix F.6 specifies their evaluation schedules and selection protocol.

G.4 PointMaze task results

G.5 Common-budget PointMaze proposal comparison

The following normalized success areas compare the shared frozen interface with and without TRAIN-fitted proposal mixing. Trapezoidal integration uses recorded periodic evaluations on the shared 50,092–2M primitive-call interval. Final-fifty evaluations remain separate from these learning-curve summaries.

Table G.3: PointMaze final success at 2M primitive training calls, with fifty episodes per seed and three online seeds. Macro interfaces share the TD-style high-level core and execution schedule; prior mix additionally changes proposals. Primitive and native-algorithm rows retain their respective decision and learning configurations.

Group	Method	U	Medium	Hard
Macro	PFM (prior mix)	1.000	1.000	.460
	PFM (prior off)	.993	.947	.347
	Regression decoder	1.000	.460	.233
	Small regression decoder	.993	.727	.267
Primitive	Flat SAC	1.000	.387	.133
	Flat TD ($h = 20$)	.820	.140	.060
Native RL	FPO++	.760	.333	.200
	DPPO	.433	.167	.120
	ReinFlow	.233	.060	.100
	DSRL-SAC	.220	.173	.040
	DSRL-NA	.153	.120	.087

Table G.4: Mean normalized PointMaze success AUC (%) for the proposal intervention, averaged over three online seeds.

Proposal	U	Medium	Hard
PFM prior mix	96.62	64.10	36.15
PFM prior off	91.11	43.29	24.57

G.6 Benchmark aggregates at the common prefix

All fifty tasks and three online seeds contribute their recorded ten-episode periodic evaluations at 200k primitive interactions. Each benchmark difference is PaLMR's task-balanced mean minus the corresponding baseline's mean. A seed-index aggregate averages one online seed over all fifty tasks, conditional on the pretrained interfaces.

PaLMR's mean advantage over flat TD-MPC2 is 11.80 percentage points, with positive differences for all three seed-index benchmark aggregates. Its mean AUC advantage is 13.44 points over the common 50k–200k interval, using all four recorded periodic checkpoints. These are benchmark-level comparisons.

Table G.5: Common-200k benchmark success differences in percentage points. All fifty tasks contribute to every entry. Seed-index differences describe the three complete benchmark aggregates.

Baseline	Mean difference	Seed-index differences
Flat TD-MPC2	+11.80	+11.20, +12.80, +11.40
Flat SAC	+39.40	+36.00, +39.00, +43.20
DPPO	+73.00	+71.00, +74.00, +74.00
DSRL-SAC	+54.93	+56.80, +51.20, +56.80
ReinFlow	+72.53	+70.60, +72.40, +74.60
FPO++	+77.00	+77.20, +75.20, +78.60

G.7 Predictive-coordinate experiments

We first give the learned-encoder protocol and seed-level results for Section 5.2, then report a complementary intervention that changes the coordinates of fixed motion requests through invertible maps. These studies use separate recorded populations and do not retrain the motion executor or a downstream controller.

G.7.1 Learned-encoder comparison

Figure 4 compares geometry-informed and reconstruction-only encoder coordinates on the same recorded geometry-PFM executions. The two encoders share the 5399 pretraining trajectories, architecture, four-dimensional code and 44,000 training updates; their objectives differ by the geometric hinges. The predictor comparison trains no encoder, decoder or planner. Matched predictors ($6 \to 128 \to 128 \to 4$, Adam at 10^{-3}, batch 128, 5000 updates) share initialization and minibatch order within each of three seeds and train on the same 864 executions from twenty-four modes. Evaluation uses 320 withheld-direction executions (64 request groups, five physical draws each) and 640 withheld-distance executions (128 groups, five draws each). These populations and fits are separate from the deduplicated invertible-map comparison in Table G.7. The metric is mean Euclidean XY endpoint prediction error of the four-output endpoint/final-velocity predictor. Predictor seeds quantify fitting variation conditional on the fixed encoders and recorded decoder, not repeated low-level pretraining.

G.7.2 Coordinate-remapping and training-support study

To isolate the effect of coordinate organization, we keep the physical records, supervision and predictor budget fixed within each training-support condition and compare identity, orthogonal and nonlinear maps of the same four-dimensional request code. The maps are invertible, so the available motion information is unchanged. Figure G.4 pools the held-out groups; Table G.7 separates direction and distance requests.

Physical transition data and splits. The predictor study uses 5760 existing D1 unit-gain full-$H = 100$ PFM executions: twenty-four known modes, eight direction-midpoint codes and sixteen distance-midpoint codes at resting and downstream-visited velocities. Grouping by code, measured initial velocity quantized at 10^{-7} and sampler-noise key gives 3000 input/noise groups, or 600 code/velocity classes with five draws each. The earliest row in each group supplies the measured transition; translated duplicates and their residuals are separately recorded.

Training uses known-mode codes and noises zero, one and two. The twelve-mode condition contains four alternating directions at all three radii; the twenty-four-mode condition adds the other four directions. They contain 444 and 888 training groups with the same thirty-five measured training velocities. Both conditions use the same 1040 held-out-code test groups: 345 direction and 695 distance, including 800 at training velocities and 240 at held-out velocities. Known-code tests for the twenty-four-mode condition contain 592 seen-velocity/new-noise and 192 new-velocity/new-noise groups using noises three and four; the twelve-mode counts are 296 and 96.

Predictor and coordinate interventions. Inputs are $[v_x/0.25, v_y/0.25, f(z)]$ and targets are $[\Delta x/0.9, \Delta y/0.9, v'_x, v'_y]$. A two-hidden-layer width-128 SiLU feedforward network produces four

Table G.6: **Learned-encoder endpoint prediction error (mm).** Each seed column is the mean over that family's physical executions. The summary reports the mean and sample standard deviation across the three paired predictor seeds.

Family	Encoder	Seed 0	Seed 1	Seed 2	Mean ± SD
Direction	Geometry-informed	22.490	23.090	24.318	23.299 ± 0.932
Direction	Reconstruction-only	55.718	57.348	57.488	56.851 ± 0.984
Distance	Geometry-informed	15.653	18.344	18.108	17.368 ± 1.490
Distance	Reconstruction-only	36.642	38.143	37.685	37.490 ± 0.770

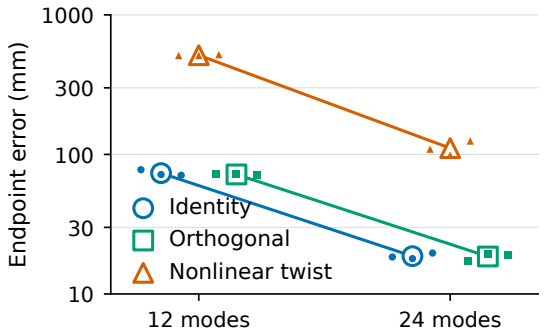

Figure G.4: **Coordinate organization on fixed physical transitions.** Mean Euclidean endpoint prediction error over all 1040 held-out input groups (345 direction and 695 distance request/velocity/noise groups), with twelve or twenty-four training modes. Small markers are three paired predictor seeds and large markers their means; the vertical axis is logarithmic. All maps use identical records within each support condition.

outputs. Adam uses 10^{-3}, batch 128 and exactly 5000 updates, evaluated at the final checkpoint. Three predictor seeds match initialization and minibatch indices across maps within each support condition.

The maps are identity, the normalized 4×4 Hadamard matrix, and a twist rotating (z_0, z_1) by $2\pi z_2$ while preserving z_2, z_3. Its inverse applies the opposite rotation. All maps preserve the physical data and available behavior; they alter only the coordinates supplied to the predictor. Endpoint prediction error is Euclidean displacement error in millimeters, velocity error is in meters per second and normalized four-state MSE is secondary. Each input/noise group receives equal weight, followed by equal weighting of predictor seeds.

Identity and orthogonal coordinates give similar accuracy, while the nonlinear map has higher error in every held-out family/seed aggregate. With twenty-four training modes, their pooled endpoint errors are 18.66, 18.55 and 110.93 mm, respectively. Expanding support reduces identity-coordinate pooled error from 73.43 to 18.66 mm, and withheld-direction and distance errors by 74.13% and 74.89%. This expansion doubles training groups from 444 to 888, so it measures expanded support together with more data rather than coverage at a fixed sample quantity. These eighteen fits use fixed physical transition data in open space and the shared pretrained decoder.

Table G.7: **Identical physical transitions, different predictive coordinates.** Mean Euclidean endpoint prediction error in mm over three paired predictor seeds. Every coordinate map uses the same physical records and predictor budget within a support condition; the two support conditions share the same held-out test groups. Identity and orthogonal coordinates preserve the original code geometry. Figure G.4 weights the two families by their 345 and 695 groups.

Training modes	Withheld codes	Identity	Orthogonal	Nonlinear twist
12	Direction	85.81	83.04	458.35
12	Distance	67.29	66.32	541.68
24	Direction	22.20	21.56	144.19
24	Distance	16.90	17.06	94.42

Table G.8: Coordinate-remapping results by predictor seed. Endpoint columns give each seed's mean error in millimeters; velocity averages over the same groups and seeds.

Modes / family	Map	Seed 0	Seed 1	Seed 2	Velocity
12 / Direction	Identity	91.308	84.442	81.680	0.0812
12 / Direction	Orthogonal	83.656	83.531	81.939	0.0776
12 / Direction	Twist	443.383	462.160	469.505	0.4354
12 / Distance	Identity	70.893	65.482	65.493	0.0639
12 / Distance	Orthogonal	66.049	66.165	66.741	0.0626
12 / Distance	Twist	545.283	536.427	543.326	0.5217
24 / Direction	Identity	22.180	21.009	23.398	0.0215
24 / Direction	Orthogonal	19.104	22.426	23.143	0.0199
24 / Direction	Twist	137.810	125.245	169.504	0.1429
24 / Distance	Identity	16.530	16.391	17.777	0.0179
24 / Distance	Orthogonal	16.188	17.899	17.106	0.0184
24 / Distance	Twist	94.824	85.890	102.537	0.0922

G.8 PFM AND REGRESSION DECODER RESPONSE-PREDICTION DIAGNOSTICS

This appendix supports the decoder comparison in Section 5.4. The two decoders receive the same geometry-informed request coordinates, and each predictor learns its own decoder's recorded physical responses. The fixed-transition coordinate interventions are separate experiments reported in Section 5.2 and Appendix G.7.

Predictor and evaluation units. A two-hidden-layer width-128 SiLU predictor maps initial velocity divided by 0.25 and the four-dimensional request code divided by a positive TRAIN scalar RMS to 100 relative XY positions, flattened to 200 outputs and divided by 0.9 m. Every fit has $43{,}208$ parameters and uses Adam at 10^{-3}, batch 128 and exactly 5000 updates. Three initialization/minibatch seeds are paired across the two decoder arms. Physical records come from one pretrained checkpoint per decoder. PFM has five recorded draws per request, while the deterministic regression decoder has one execution. Sampler draws are not independent low-level training seeds.

The two request populations are analyzed separately. The resting population contains 47 TRAIN anchor requests spanning 24 distinct action modes and carrying 43 distinct codes, plus 8 direction and 16 distance midpoint tests. Both endpoints of each midpoint recipe are TRAIN anchors. The regression-decoder records store relative motion only, so the resting records are matched by relative motion at zero initial velocity. Translated repetitions are not counted as independent support.

The visited-context population contains 288 TRAIN request groups, 64 held-out direction groups, 128 held-out distance groups and a separate 96 known-code/held-out-velocity groups. The measured reset states match across decoders. Three recorded PFM draws per TRAIN input supply supervision; held-out requests are evaluated using their five recorded draws. The resting and visited populations use distinct query constructions and are not pooled.

Prediction metrics. Evaluation averages each request's squared trajectory error over recorded draws and trajectory coordinates, then weights requests equally. Trajectory-coordinate RMSE is computed per predictor seed in physical millimeters and then averaged over seeds. Endpoint prediction error

Table G.9: **Response prediction under shared request coordinates.** Means over three terminal predictor fits. Trajectory RMSE and Euclidean endpoint prediction error are in mm. Within each population and query family, PFM and the regression decoder share the requested codes and predictor protocol but supply their own physical response targets.

		Trajectory RMSE		Endpoint error	
Population	Family	PFM	Regression decoder	PFM	Regression decoder
Resting	Direction	9.2282	38.2985	19.2695	90.5301
Resting	Distance	4.9632	19.5407	11.4558	41.1008
Visited	Direction	13.5798	113.6679	25.2020	166.3065
Visited	Distance	8.8745	82.7781	16.6747	137.9558

Table G.10: **Known requests at held-out initial velocities.** The 96 request groups share measured initial states across decoders. Prediction metrics average three predictor fits; execution success is the physical endpoint criterion, measured on five PFM draws per group and one deterministic regression decoder execution.

Decoder	Trajectory RMSE (mm)	Endpoint prediction (mm)	Execution success (%)
PFM	5.3623	10.2645	89.5833
Regression decoder	83.4750	120.4289	65.6250

is the mean Euclidean error of the final predicted XY point against the recorded executed endpoint. These metrics concern prediction of the motion that occurred, separately from whether it reached the requested target. Fitting squared error to a request's mean recorded trajectory has the same parameter gradient as fitting its individual draws, because their empirical within-request dispersion is independent of predictor parameters.

PFM response-prediction error is lower for both held-out request families in both populations, and every PFM predictor seed lies below every regression-decoder seed in trajectory RMSE. The decoder architectures have approximately matched capacity, while the predictors have identical capacity and update budgets. This comparison evaluates the predictability of the physical response maps fitted by these two decoder recipes.

Table G.11: Trajectory-coordinate RMSE in mm for each predictor seed of the PFM–regression-decoder comparison. All fits use the fixed terminal update 5000.

Population	Family	Decoder	Seed 0	Seed 1	Seed 2
Resting	Direction	PFM	9.7742	9.8863	8.0240
Resting	Direction	Regression decoder	39.6149	39.2213	36.0593
Resting	Distance	PFM	5.7368	4.4258	4.7271
Resting	Distance	Regression decoder	20.1293	19.5807	18.9121
Visited	Direction	PFM	14.4444	12.5196	13.7753
Visited	Direction	Regression decoder	117.7020	102.4153	120.8864
Visited	Distance	PFM	8.8305	8.8409	8.9521
Visited	Distance	Regression decoder	87.1390	80.7379	80.4575
Visited	Held-out velocity	PFM	5.3984	5.3264	5.3621
Visited	Held-out velocity	Regression decoder	87.4251	77.7152	85.2847

These supervised fits use recorded physical transitions and take no new environment steps.

G.9 Finite-menu physical motion coverage

The coverage estimators for Section 5.3 fix a common resting context and compare 47 TRAIN-anchor queries with their union with 24 predefined midpoint queries. The descriptive statistic in Table G.12

represents each query by its mean endpoint over five fixed sampler draws, then measures target-to-menu distance; every target's own added midpoint is the minimizing request in the enriched menu. Its overall distance changes from 176.001 to 19.584 mm, with direction 232.771 to 27.374 and distance 147.616 to 15.689 mm.

The best-available goal distance in Table G.12 first averages endpoint distances over sampler draws and then minimizes over queries, following the expectation-before-minimum form of Appendix E.1. Its overall error changes from 176.002 to 19.609 mm, with direction 232.771 to 27.388 and distance 147.618 to 15.720 mm. The maximum across the twenty-four targets falls from 234.230 to 44.976 mm. Winners and improvement directions agree between the two estimators; the model-selected distances in Figure 4(b) use the same expectation over recorded draws.

G.10 Planning with interpolated motion options: evaluation protocol

Table G.12 reports the aggregate model-choice and fixed-planner outcomes shown in Figure 4(b–c). All thirty paired fresh Medium episodes contribute, with ten episodes for each of three frozen checkpoints; means include successful and unsuccessful episodes. Mean macro decisions fall from 21.17 to 16.83. The intervention is specified below.

Menus. The anchor menu contains 47 TRAIN request entries in fixed order, 43 of them distinct numerical codes. The enriched menu appends 8 direction and 16 distance midpoint requests, giving 71 entries and 67 distinct codes. Codes are recorded unit-norm requests cast to float32, with common raw radius one and no goal- or outcome-dependent rescaling; the executor applies its unchanged normalization. Menu construction reads no outcome, goal or test trajectory. Ties are broken by the lowest menu or slot index.

Frozen-model choice. We evaluate three independently trained response predictors, each fitted for 5000 updates on the 47 anchor response groups. Its six inputs are the pre-execution velocity $v_0/0.25$ and the request code divided by the TRAIN scalar RMS; its 200 outputs are 100 relative XY positions divided by 0.9. It was trained on requested codes, rather than encodings of realized test futures. All tests share the resting start and 24 original targets. Both arms score 71 slots, with fixed repetitions of the first 24 anchor entries in the anchor arm, and select the smallest predicted final-position distance before reading any recorded endpoint.

Each choice is evaluated at the endpoints of the five recorded 100-step motions of the selected request. The $3 \times 24 \times 5$ scored combinations per arm reuse those recorded draws; they are not new independent rollouts. A local endpoint hit requires complete execution with endpoint distance at most 50 mm and is separate from native PointMaze task success. All these distances are Euclidean millimeters.

Best-available motion reference. For each target, an oracle evaluates the mean goal distance across the five recorded executions of every permitted request and selects the request with the smallest mean. Averaged over targets, this best-available distance is 176.00 mm for anchors and 19.61 mm for the enriched menu, compared with model-selected distances of 176.22 and 19.61 mm (Table G.12). The predictor therefore achieves nearly the best available performance in each menu, while interpolation supplies motions that reach closer to the targets. This oracle uses recorded outcomes as a reference; the predictor selects requests before reading those outcomes. Appendix G.9 distinguishes this expectation-before-minimum statistic from distance to the mean endpoint.

Fixed planner. We evaluate the three final prior-off Medium policies of the TD-MPC2-style learner, trained for 2M primitive steps (Appendix F.5). All task and motion modules remain frozen throughout evaluation. The native planner uses length $T = 3$, 256 candidate sequences, three refinements, 32 elites, temperature 0.5, best-sample selection and discount 0.99. Every candidate is projected to its nearest allowed code by squared Euclidean distance before scoring, in every iteration and both arms. Both menus use 768 sequence evaluations and 2304 model transitions per decision.

Episode e of policy seed s resets with seed $90000 + 1000s + e$; both menus receive identical initial observations and goals. Planner and decoder random streams are seeded separately for each policy seed, episode and decision, with matched seeds across the two menus. PFM retains stochastic decoding. Each decision generates $H = 100$ actions and executes up to $K = 100$; native termination or the 6000-step episode cap can stop a shorter prefix. Primitive steps count actual simulator calls.

Table G.12: **Fixed models use additional motion requests.** Open-space rows use the same 24 targets and five recorded draws per request. Best-available and model-selected goal distances report mean ± standard error across targets; model-selected values first average the three predictor seeds within each target. The first row instead measures distance to the mean endpoint (Appendix G.9). Figure 4(b) shows model-selected goal distances. Medium rows aggregate all 30 fresh paired episodes, ten for each of three frozen task checkpoints.

Measure	Anchor menu	Enriched menu
Oracle distance to mean endpoint (mm)	176.00	19.58
Best-available goal distance, $T = 1$ (mm)	176.00 ± 8.37	19.61 ± 2.17
Model-selected goal distance, $T = 1$ (mm)	176.22 ± 8.38	19.61 ± 2.17
Local endpoint hit rate, $T = 1$ (%)	0	100
Medium task success, $T = 3$	25/30	25/30
Mean primitive steps, all episodes	2071.3	1675.1
Mean macro decisions, all episodes	21.17	16.83

The evaluation records 1140 macro decisions without training. Added requests are executed in $0/635$ anchor-menu decisions and $200/505$ enriched-menu decisions.

G.11 Behavior-informed candidate proposals

Proposal intervention. The frozen TRAIN-fitted prior encodes all 5399 chunks, splitting $5129/270$ for fitting/validation. A two-hidden-layer width-256 ReLU network maps standardized initial XY to Gaussian mean/log-standard-deviation, clamping log-standard-deviation to $[-5, 2]$. It minimizes Gaussian negative log likelihood with Adam 10^{-3}, batch 256, for 200 epochs (4200 updates), selecting the validation-best checkpoint. Within the fitting XY box it uses the conditional Gaussian and outside it the frozen marginal. Samples are clipped to $[-2, 2]^4$ and normalized by the executor. Prior mix replaces exactly the first 64 candidates' root actions in the first planning refinement; other positions and iterations use the ordinary proposal. Prior off uses the ordinary sampler throughout.

Online task results. With the same frozen motion interface and separate online training under each proposal, prior mixing raises Medium/Hard final success from $94.67/34.67\%$ to $100/46\%$ and normalized AUC from $43.29/24.57\%$ to $64.10/36.15\%$. U-Maze final success changes from 99.33% to 100%. Appendix G.5 gives the three-layout AUC summaries. Section 5.3 reports the separate evaluation that freezes the task models and changes only the permitted motion menu.

G.12 Interpolation and velocity-response success

Table G.13 gives the resting and changed-velocity execution counts together. The known-mode/start partition fixes the requested motion while changing initial velocity; the full panel also contains the direction and distance intermediate requests. Query construction is specified in Table F.2.

Execution-success error bars. Figure 5(b–c) reports the observed success fraction $\widehat{p} = k/N$ with one binomial standard error, $\sqrt{\widehat{p}(1-\widehat{p})/N}$, converted to percentage points. Here N is the number of recorded physical executions in the corresponding table cell: five draws per PFM query and one execution per deterministic-method query. The error bars are zero when all recorded executions succeed or all fail. Figure 5(a) instead uses the sample standard deviation of the three predictor-seed RMSE values divided by $\sqrt{3}$.

G.13 Visited-velocity intervention

The visited-velocity panel assigns one velocity to each of the original 576 requests using forty-eight velocities from six D1 Hard-maze TRAIN sources: prior off and prior mix at seeds zero, one and two. Eligible macro records have decision index divisible by fifty. RNG seed 20260915 selects eight records uniformly without replacement from each source in fixed source order. Query i receives velocity $i \bmod 48$, so every velocity is used twelve times. Of the 576 rows, 552 have nonzero velocity

Table G.13: **2D interpolation and context-conditioned execution.** All entries are successes/physical executions. Resting direction, distance and known/start families contain 64, 128 and 384 queries; the full panel contains all 576. PFM uses five draws per query and deterministic controls one. Success requires complete 100-action execution and endpoint error at most 50 mm.

	Resting				Visited velocity	
Method	Direction	Distance	Known/start	All	Known/start	All
PFM	320/320	640/640	1920/1920	2880/2880	1467/1920	2216/2880
Regression decoder	0/64	72/128	384/384	456/576	188/384	225/576
Action blending	64/64	128/128	384/384	576/576	128/384	191/576

Table G.14: Locked 3D request identities. Positions are in meters. Start pairs use the fixed $+x$, 60 mm request and $\eta = 0$. Direction tuples are normalized before defining their physical spherical midpoint.

Start case	Seen position	Held-out position
0	$(-0.020, 0.650, 0.225)$	$(0.100, 0.650, 0.225)$
1	$(0.200, 0.650, 0.225)$	$(0.115, 0.650, 0.225)$
2	$(0.100, 0.730, 0.225)$	$(0.100, 0.665, 0.225)$
3	$(0.100, 0.570, 0.225)$	$(0.100, 0.650, 0.240)$
Direction case	**TRAIN direction 1**	**TRAIN direction 2**
0	$(-1, -1, 0)$	$(-1, 0, 0)$
1	$(-1, -1, 0)$	$(0, -1, 0)$
2	$(-1, 0, -1)$	$(-1, 0, 0)$
3	$(-1, 0, -1)$	$(0, 0, -1)$
4	$(-1, 0, 0)$	$(-1, 0, 1)$
5	$(-1, 0, 0)$	$(-1, 1, 0)$
6	$(-1, 0, 1)$	$(0, 0, 1)$
7	$(-1, 1, 0)$	$(0, 1, 0)$

and twenty-four originate after reset; speeds range from zero to 0.7704, with mean 0.2122. The 384 known-mode/start requests form the predefined partition used for the main context comparison. Positions, anchors, targets and the 50 mm success threshold remain fixed. Physical repetitions are conditional on the single frozen checkpoint of each learned decoder.

G.14 Production 3D geometry protocol

The panel uses eight contexts and five fixed sampler noises per query: four seen/held-out start pairs for one fixed $+x$, 60 mm TRAIN mode (320 executions), eight interpolated direction modes (320), and sixteen distance modes (640). Anchor codes are normalized means of accepted TRAIN encodings. Direction requests use spherical midpoints of neighboring anchor codes at 60 mm. Distance requests interpolate adjacent $30/60$ or $60/90$ mm radii to 45 or 75 mm in the same direction. The distance directions before normalization are $(-1, 0, 0)$, $(-1, -1, -1)$, $(-1, -1, 0)$, $(-1, -1, 1)$, $(-1, 0, -1)$, $(-1, 0, 1)$, $(-1, 1, -1)$ and $(-1, 1, 0)$.

Each physical trial executes all twenty-five generated motion actions. Preparation uses gain-ten proportional positioning until within 2 mm or eighty calls, then forty settling calls. Prepared trials require final position within 10 mm of the preparation anchor; fixed-start trials also require previous raw displacement norm at most 1 mm. These checks precede the tested chunk. All 1280 scheduled trials execute: forty fixed-code, fifteen direction and sixty-five distance trials violate a preparation tolerance, giving 1160 prepared trials. Task learning separately uses five-step prefixes.

G.15 Full-horizon production 3D fidelity

All 1280 scheduled PFM executions contribute, with preparation status recorded before the tested motion. The fixed-code start panel uses one $+x$, 60 mm mode. These measurements use full twenty-five-step chunks.

Table G.15: Production PFM request fidelity in 3D. Endpoint errors are in mm and angle errors in degrees. Prepared and all-executed populations are shown separately.

	Prepared			All executed	
Panel / measure	n	Mean	P90	n	Mean
Fixed code / endpoint	280	1.224	1.555	320	1.240
Held-out starts only	160	1.188	1.294	160	1.188
Direction / angle	305	6.198	16.270	320	6.384
Distance / endpoint	575	8.543	24.589	640	8.416