TD-MTP: COUPLING STRUCTURED PLANNING WITH MULTI-STEP VALUE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In model-based robot learning, a planner selects actions and generates data for its own value function. A planning change can therefore help under one value target and hurt under another. We study this dependence with TD-MTP, which combines a structured tensor-graph planner with replay-based multi-step value targets in TD-MPC2. We compare both planners under one-step and multi-step targets on ten continuous-control tasks. The structured planner improves mean return on one task under one-step targets and five under λ-returns; its effect is more favorable under λ-returns on eight. On `h1-stair`, the planner changes from reducing mean return by 232 to increasing it by 191. Each added alone, the multi-step target raises mean return above the baseline on eight tasks and the planner on one. Controls suggest that elite fitting contributes to this interaction, while the isolated contribution of tensor proposals remains unresolved. For fixed policies, our error bound separates bootstrap perturbation from behaviour–prior mismatch. It accounts for the terminal bootstrap retained by finite replay windows. Across 38 tasks in four benchmarks, TD-MTP records 10 wins, 22 ties and 6 losses against TD-MPC2 under stated tolerances. Mean learning curves show earlier gains on `humanoid-run`, but slower learning on `stick-pull`, relative to TD-MPC2. These results support evaluating planners and value targets together during robot learning. Learning curves, per-task scores and videos: `https://tdmtp.github.io/`.

1 INTRODUCTION

Model-based robot learning combines a learned dynamics model, a value function and online planning to select actions (Hansen et al., 2024). Planning can search beyond the learned policy, but the transitions collected by executing its actions also train the model and critic used in the next search. A planner therefore changes both the current action and the learner's future training data. We aim to improve learning within a limited budget of environment interactions. Our question is whether the same planning change helps under different value targets. This matters when a planner is judged on fixed checkpoints and then used throughout training: better search on a fixed model need not produce a better learned agent.

TD-MPC2 refines a Gaussian proposal over action sequences and learns values from a one-step target. Tensor-graph sampling offers a structured way to generate candidates (Le et al., 2025b); multi-step returns incorporate later rewards from collected trajectories (Sutton, 1988; Peng & Williams, 1996). TD-MTP combines these established ideas to study their interaction. The planner determines which rewards enter replay, the target determines how they update the critic, and the critic ranks the next planning candidates. Longer returns can reduce dependence on estimated future values, but they also follow more actions from a behavior policy that differs from the learned policy prior. This tradeoff motivates both our analysis and our experiments.

Empirical finding. We compare two planning configurations under two targets on seven Humanoid-Bench tasks at 2M environment steps and three DMControl tasks at 300k. All four methods use a common MPPI evaluation planner, so the comparison measures the effect of the training configuration. The structured planner's mean effect is positive on 1 of 10 tasks with a one-step target and 5 of 10 with a λ-return. The difference between these effects is positive on 8 of 10 (Figure 1). Added to the baseline separately, the λ-return raises mean return on 8 of these ten tasks and the structured planner

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.

on 1. On `h1-stair`, the planner alone scores 293.6 and the target alone 459.6. Both fall below the 525.4 baseline, but their combination reaches 651.0. The two planners are complete configurations that differ in fitting and execution rules as well as in proposals. Controls in §4.1 suggest a role for elite fitting; they do not establish that tensor proposals drive the interaction.

Analytical account. To interpret the target, we analyse its expected fixed-policy operator, building on Peng's Q(λ) (Kozuno et al., 2021). The error bound separates bootstrap perturbation from behavior–prior mismatch. We then account for the terminal value estimate in a finite replay window and give the exact return across episode boundaries. The analysis explains what increasing λ changes in policy evaluation; it does not predict return improvements during nonlinear control learning.

Contributions.

- A paired study that separates each component's gain from planner–target interaction, with controls for candidate generation and elite fitting within TD-MTP.
- A self-contained replay-target specification and fixed-policy error decomposition for infinite and finite windows, with explicit boundary handling and complete derivations.
- An evaluation on 38 tasks in four benchmarks, reporting task-dependent gains, learning curves, manipulation failures, and sensitivity to the training budget and target.

2 BACKGROUND

We consider a Markov decision process (MDP) with continuous actions $a \in [-1, 1]^A$, reward $r(s, a)$ and discount $0 < \gamma < 1$. TD-MPC2 (Hansen et al., 2024) maps an observation s to a learned representation $z = e(s)$ and predicts its evolution with a dynamics model $\hat{f}(z, a)$. It also learns a reward predictor, an optional termination predictor, an ensemble of action-value functions (the critic), and a stochastic policy $\pi(\cdot \mid z)$ (the policy prior). The critic estimates discounted future reward; the prior proposes actions for planning and value estimation. We write Q for the scalar critic value used in planning and Q^- for the value from the target ensemble used in learning. The target-ensemble reduction is stated in Appendix A. Stored transitions form the replay buffer, from which we sample training data. For a stored transition, let d_t mark true termination and z_t^+ encode the actual successor observation, before any reset. The reference one-step target combines the observed reward with a critic estimate of subsequent rewards, called a bootstrap:

$$y_t = r_t + \gamma(1 - d_t)Q^-(z_t^+, a_t^+), \qquad a_t^+ \sim \pi(\cdot \mid z_t^+). \tag{1}$$

Rewards from transitions after t do not enter y_t directly; repeated bootstrapping carries them back one step per update. Targets use encoded observed successors; the critic is regressed at model-unrolled latents, and model predictions also enter planning rollouts (Appendix A). We distinguish predicted latents $\hat{z}_h$ from encoded observations z_t. The index h denotes planning time, t a replay position, i a planner iteration, and k an environment step.

At acting time, the planner rolls out an H-step action sequence under the learned model, starting at $\hat{z}_0 = e(s)$ and applying $\hat{z}_{h+1} = \hat{f}(\hat{z}_h, a_h)$. Let $b_0 = 1$ and $b_{h+1} = b_h(1 - \hat{d}_h)$, where $\hat{d}_h \in \{0, 1\}$ is the predicted termination flag for transition h; take $\hat{d}_h = 0$ when no termination head is used. To compare candidate sequences, the planner adds predicted rewards to a terminal value:

$$J(a_{0:H-1}) = \sum_{h=0}^{H-1} \gamma^h b_h \hat{r}(\hat{z}_h, a_h) + \gamma^H b_H Q(\hat{z}_H, a_H^\pi), \qquad a_H^\pi \sim \pi(\cdot \mid \hat{z}_H). \tag{2}$$

The TD-MPC2-style reference configuration uses a model predictive path integral (MPPI) planner with I rounds of sampling and fitting. Each round combines policy-prior rollouts with draws from a diagonal Gaussian of mean $\mathbf{m} \in \mathbb{R}^{H \times A}$ and standard deviation $\varsigma \in \mathbb{R}_+^{H \times A}$ to form S candidate sequences, with actions clipped to $[-1, 1]$. The best K candidates, called elites, determine the next Gaussian fit. The planner samples a final elite using score weights and executes its first action, with exploration noise during data collection. The fitted mean initializes the next planning call. We use MPPI to refer to this elite-based implementation. The behavior policy μ is the planner's executed-action distribution, including any collection noise; it differs from the prior π. Our reference

method is a JAX port with the configuration differences listed in Appendix A; the DMControl suite also uses published-implementation PyTorch baselines, labeled separately. The reference learns from windows of H_{train} transitions. All primary comparisons use $H = H_{\text{train}} = 3$.

3 TD-MTP

TD-MTP couples a data-collection planner with a replay target: the planner supplies trajectories, and the resulting critic guides the next search. We specify both components, then analyze the target. Architecture, remaining losses, and optimization are shared across paired arms; Appendix A gives implementation details.

3.1 Structured planning with a tensor graph

A Gaussian refit can concentrate search around one solution or place its mean between separated elites. The executed action still comes from one elite, so the concern is the population sampled at the next iteration. MTP supplies additional candidates by recombining control points (Le et al., 2025b). Let $\Phi \in \mathbb{R}^{H\times M}$ linearly interpolate M control points to H actions; its entries are non-negative, and each row sums to one. At iteration i, the graph contains M layers of N nodes, $C^{(i)} \in \mathbb{R}^{M\times N\times A}$. Candidate p independently selects $j_m^p \sim \mathrm{Unif}\{1,\dots,N\}$ at each layer and follows the path

$$a^{(p)} = \Phi\,[C^{(i)}_{1,j_1^p};\dots;C^{(i)}_{M,j_M^p}] \in \mathbb{R}^{H\times A}. \tag{3}$$

We sample paths independently across candidates and layers. There are N^M paths, but different paths can yield the same sequence. For the reported interpolation at $H = M = 3$, the third layer carries no weight. The N^3 paths then give at most N^2 distinct sequences, with $a_1 = (a_0 + a_2)/2$. Appendix B derives their temporal covariance and a coverage bound for the finite graph. Whether this coverage improves exploration depends on node locations, scoring and action selection.

Trust region. We adapt the graph to the previous elite fit while retaining a minimum search width. Let $\Psi \in \mathbb{R}^{M\times H}$ map sequence statistics to control-point layers, with non-negative entries and row sums of one. From the previous iteration's fit,

$$C^{(i)}_{m,j} = \mathrm{clip}_{[-1,1]}\Big[(\Psi\mathbf{m}^{(i-1)})_m + \big(c_e(\Psi\varsigma^{(i-1)})_m + \rho_i\big)\odot\xi_{m,j}\Big]. \tag{4}$$

The symbol $\odot$ denotes coordinatewise multiplication; clipping is also coordinatewise. The noise vectors $\xi_{m,j}$ come from Gaussian-quantile grids, independently permuted for each layer and action coordinate. The floor ρ_i anneals monotonically from ρ_{start} to $\rho_{\text{end}} > 0$ across the I iterations, keeping the width before clipping positive. Clipping can still merge nodes at the action limits. The runs use a cosine schedule (Appendix A). We use $c_e = 0.5$, $M = 3$, $N = 16$ and linear interpolation. At $H = M = 3$, the rows of Φ are $(1,0,0)$, $(\frac{1}{2},\frac{1}{2},0)$ and $(0,1,0)$.

Candidate composition and fitting. Policy-prior rollouts and the best sequence found so far occupy reserved slots, leaving P free. At environment step k of a T-step run, set $\beta_k = \beta\max(0, 1-2k/T)$ and $n_{\text{tensor}} = \mathrm{clip}_{[0,P-1]}(\mathrm{round}(\beta_k P))$, with initial $\beta = 0.5$, so at least one free slot always takes a local draw. The remaining $P - n_{\text{tensor}}$ slots use local draws $\mathrm{clip}_{[-1,1]}(\mathbf{m}^{(i-1)} + \varsigma^{(i-1)}\odot\zeta^{(p)})$ with $\zeta^{(p)} \sim \mathcal{D}_{\text{loc}}$, where $\mathcal{D}_{\text{loc}} = \mathcal{N}(0,\mathrm{Id})$ in the reported runs. This noise is independent across sequences and distinct from the tensor quantiles ξ.

To refit the proposal, let $\mathcal{E}_i$ be the top-K index set ($1 \le K \le S$), and write $J_p = J(a^{(p)})$ and $J_{\max} = \max_{p\in\mathcal{E}_i} J_p$. With inverse temperature $\tau \ge 0$, the elite fit is

$$\begin{aligned} w_p &= \frac{\exp(\tau(J_p - J_{\max}))}{\sum_{v\in\mathcal{E}_i}\exp(\tau(J_v - J_{\max}))}, \qquad p\in\mathcal{E}_i, \\ \mathbf{m}^{(i)} &= \sum_{p\in\mathcal{E}_i} w_p a^{(p)}, \qquad v^{(i)} = \sum_{p\in\mathcal{E}_i} w_p(a^{(p)} - \mathbf{m}^{(i)})^2. \end{aligned} \tag{5}$$

Squares are coordinatewise. Both planners use $\tau = 2$; TD-MTP takes $\varsigma^{(i)} = \mathrm{clip}_{[0.05,2]}\sqrt{v^{(i)}/(1-\sum_p w_p^2)}$ over $K = 32$ elites, the reference $\mathrm{clip}_{[0.05,2]}\sqrt{v^{(i)} + 10^{-6}}$ over

$K = 64$. The TD-MTP terminal action in Equation 2 is the prior mean; the reference samples it. These choices and the execution rule below are part of the planning comparison. The variance denominator requires a numerical safeguard when weights concentrate (Appendix A).

Algorithm 1 TD-MTP planning during data collection at environment step k.

Require: Current latent z, initial $(\mathbf{m}^{(0)}, \varsigma^{(0)})$, initial best sequence $a^\star$; settings in §3.1
1: Set $P = S - 25$, β_k and n_{tensor}; obtain 24 policy-prior sequences from z
2: **for** $i = 1, \ldots, I$ **do**
3: Construct $C^{(i)}$ from $(\mathbf{m}^{(i-1)}, \varsigma^{(i-1)})$ using Eq. 4
4: Draw n_{tensor} tensor paths (Eq. 3) and $P - n_{\text{tensor}}$ local sequences
5: Add the 24 prior sequences and $a^\star$ to form the S-candidate population
6: Score each sequence from $\hat{z}_0 = z$ using Eq. 2; retain the top-K set $\mathcal{E}_i$
7: Set $a^\star$ to the highest-scoring candidate; refit and clamp using Eq. 5
8: **end for**
9: Execute $a_0^\star$ without added noise
10: **return** $\mathbf{m}^{(I)}$ and $a^\star$ for the next warm start

The planner scores S candidates per iteration; graph construction adds no model rollouts. Tensor candidates disappear after the midpoint of training, when β_k reaches zero, but the fitting and execution rules continue to affect data collection. This distinction motivates the component controls in Section 4.1.

3.2 Multi-step targets from collected transitions

Executing the planner supplies the trajectories used to train its critic. A multi-step target propagates the observed rewards within one update, so the next search can use information from several collected transitions. Unlike model-based value expansion (Feinberg et al., 2018; Buckman et al., 2018), this target needs no predicted rewards or transitions. It still depends on learned representations and bootstrap values, and its intermediate actions come from the behavior planner. Sample a window $(z_t, a_t, r_t, z_t^+, d_t, u_t)_{t=0}^{L-1}$ with $L = H_{\text{train}} + \ell \geq 1$. The binary flag d_t marks termination; u_t marks an external truncation or other collection boundary, so the next stored transition may belong to a different episode. The bootstrap uses the pre-reset successor z_t^+, which need not equal z_{t+1}. Set $q_{t+1} = Q^-(z_t^+, a_t^+)$ for $a_t^+ \sim \pi(\cdot \mid z_t^+)$. To use later rewards without crossing a reset, we distinguish between retaining the bootstrap and continuing the recursion:

$$\nu_t = 1 - d_t, \qquad \chi_t = (1 - d_t)(1 - u_t), \tag{6}$$

and compute backward through the window,

$$G_t = r_t + \gamma\nu_t\big[(1 - \lambda\chi_t)q_{t+1} + \lambda\chi_t G_{t+1}\big], \qquad G_L = q_L, \quad \lambda \in [0, 1]. \tag{7}$$

The critic is trained on $\text{stopgrad}(G_t)$ at supervised positions $\mathcal{T} \subseteq \{0, \ldots, L-1\}$, with no gradient through the target. Away from boundaries, λ replaces part of the next bootstrap with a return that follows the collected trajectory. This reduces the weight on nearby value estimates while increasing dependence on the behavior planner's subsequent actions.

Boundary cases. Termination gives $G_t = r_t$. External truncation gives $G_t = r_t + \gamma q_{t+1}$: it retains a valid bootstrap and stops propagation into the next episode (Pardo et al., 2018). Stopping the recursion and discarding the bootstrap are different operations. This convention treats time limits as collection cuts. If a time horizon is intrinsic to the task, its endpoint is a termination and the state must contain enough information to make the remaining horizon Markov. The last transition also reduces to $r_{L-1} + \gamma\nu_{L-1}q_L$. If sampled windows cross resets, the latent unrolls used in training must either restart from the new episode's observation or be masked after the boundary; stopping the return recursion does not by itself reset the predicted latents.

Window and one-step identity. At $\lambda = 0$, Equation 7 equals Equation 1 for every combination of termination and truncation flags. For $\lambda > 0$, extra context ℓ lets later stored rewards enter the target; Appendix C gives its exact expansion across episode boundaries. We use $\ell = 10$ for $\lambda > 0$ and $\ell = 0$ at $\lambda = 0$. Both modes supervise $\mathcal{T} = \{0, \ldots, H_{\text{train}} - 1\}$. The extra transitions supply rewards and bootstrap values but no additional loss terms. Isolating the return's effect requires matching context, supervised positions, loss normalization and update counts, even when the target formulas coincide.

3.3 BOOTSTRAP SENSITIVITY AND OFF-POLICY MISMATCH

Equation 7 uses stored rewards without importance weighting. Retrace and V-trace require likelihood ratios for the behavior that produced each action (Munos et al., 2016; Espeholt et al., 2018); the present replay specification does not provide those ratios for the planner's executed-action distribution. We therefore ask which policy the uncorrected target evaluates and how sensitive it is to bootstrap error. For fixed stationary policies in the underlying MDP, its expected boundary-free form is a Peng-style Q(λ) operator with bootstrap values averaged under the prior (Peng & Williams, 1996; Kozuno et al., 2021).

Let $Q^\pi(s,a)$ be the expected return after (s,a) when subsequent actions follow π, and define $(P^\kappa Q)(s,a) = \mathbb{E}_{s'|s,a}\mathbb{E}_{a'\sim\kappa(\cdot|s')}Q(s',a')$. The return continues under μ and bootstraps under π. Averaging over trajectories and summing over continuation lengths gives the infinite-window operator

$$T_\lambda Q = (\mathrm{Id} - \gamma\lambda P^\mu)^{-1}\big[r + \gamma(1-\lambda)P^\pi Q\big]. \tag{8}$$

Here Id is the identity operator. Its contraction factor is $\eta_\lambda = \gamma(1-\lambda)/(1-\gamma\lambda)$, and its fixed point evaluates the mixture policy $\lambda\mu + (1-\lambda)\pi$ (Kozuno et al., 2021). At $\lambda = 0$ it evaluates the prior; at $\lambda = 1$ it evaluates the behavior policy. The mixture chooses between the two policies at each successor state. Mismatch therefore measures departure from Q^π, not necessarily worse control. To separate this change in the policy being evaluated from sensitivity to value estimates, we next allow bounded perturbations of the bootstrap inputs. Algorithm 2 samples bootstrap actions; T_λ averages over both these samples and the subsequent behavior trajectory.

Proposition 1 (Bootstrap perturbation and policy mismatch)**.** *Assume bounded rewards, $0 < \gamma < 1$, $\lambda \in [0,1]$, and fixed stationary policies μ and π. Define $\varepsilon = \sup_s \mathrm{TV}(\pi(\cdot \mid s), \mu(\cdot \mid s))$ and $V_{\mathrm{sp}} = \mathrm{span}(Q^\pi)$, where $\mathrm{TV}(p,q) = \sup_B |p(B) - q(B)|$ and $\mathrm{span}(Q) = \sup Q - \inf Q$. Let $\widehat{T}_\lambda$ differ from the infinite-window operator T_λ only through perturbations of the bootstrap values bounded by $\delta \geq 0$ in sup norm, uniformly over Q. If $\widehat{T}_\lambda$ has a bounded fixed point $\widehat{Q}_\lambda$, then*

$$\big\|\widehat{Q}_\lambda - Q^\pi\big\|_\infty \;\leq\; \underbrace{\frac{\gamma(1-\lambda)}{1-\gamma}\,\delta}_{\textit{bootstrap perturbation}} \;+\; \underbrace{\frac{\gamma\lambda}{1-\gamma}\,\varepsilon V_{\mathrm{sp}}}_{\textit{behavior–prior mismatch}} \;=\; \frac{\gamma}{1-\gamma}\Big[(1-\lambda)\,\delta + \lambda\,\varepsilon V_{\mathrm{sp}}\Big]. \tag{9}$$

The bound is linear in λ with slope $\gamma(\varepsilon V_{\mathrm{sp}} - \delta)/(1-\gamma)$. For fixed δ, ε and V_{sp}, it decreases exactly when $\delta > \varepsilon V_{\mathrm{sp}}$.

The two coefficients express the tradeoff: increasing λ reduces sensitivity to bootstrap perturbations but increases the allowance for behavior–prior mismatch. Here δ bounds errors in the bootstrap inputs, not the final value error, critic disagreement, or an error added after the return update. A decreasing upper bound need not mean smaller actual error or higher control return.

Replay windows are finite, so even $\lambda = 1$ retains an endpoint value estimate. To apply the same argument to replay, we retain this endpoint when computing the total bootstrap weight. For $n \geq 1$ remaining transitions without boundaries ($n = L - t$ at supervised position t), the total bootstrap weight is

$$\eta_{\lambda,n} = (1-\lambda)\sum_{j=1}^{n-1}\gamma^j\lambda^{j-1} + \gamma^n\lambda^{n-1} \leq \gamma < 1. \tag{10}$$

The sum is empty at $n = 1$; zero powers equal one. Let $\widehat{Q}_{\lambda,n}$ be a bounded fixed point of the perturbed expected return operator truncated after n transitions, with its terminal bootstrap retained. Under the same assumptions, we obtain (Appendix C)

$$\|\widehat{Q}_{\lambda,n} - Q^\pi\|_\infty \leq \frac{\eta_{\lambda,n}}{1-\eta_{\lambda,n}}\delta + \frac{\gamma - \eta_{\lambda,n}}{(1-\gamma)(1-\eta_{\lambda,n})}\varepsilon V_{\mathrm{sp}}. \tag{11}$$

At $n = 1$ the mismatch term vanishes. At $\lambda = 1$ a finite window still bootstraps, with sensitivity $\gamma^n/(1-\gamma^n)$; as $n \to \infty$ the bound reduces to Equation 9. These are expected-operator statements for a stationary MDP, not guarantees for nonlinear critics, changing replay policies or learned latent dynamics.

Remark 1 (Choosing λ). The bound identifies a tradeoff, not an automatic choice of λ. Its uniform perturbation and policy-mismatch terms are not measured in our experiments. We choose λ empirically and report a negative sensitivity result on `reach` (Appendix E).

Table 1: **The main comparison.** Mean ± cross-seed SD. Bold marks the largest mean per task, without a significance claim. The column groups separate the two value targets; each group compares the two training planners.

	One-step target		λ-return target	
Task	TD-MPC2	+Planner	+Target	TD-MTP
DMControl: 300k steps, 5 seeds (P1)				
acrobot-swingup	489.0 ± 53	568.0 ± 47	507.4 ± 46	$\mathbf{612.5} \pm 15$
humanoid-run	72.9 ± 47	51.5 ± 52	169.6 ± 22	$\mathbf{197.6} \pm 46$
humanoid-walk	356.6 ± 72	193.5 ± 72	$\mathbf{612.6} \pm 96$	511.2 ± 92
HumanoidBench h1: 2M steps, 5 seeds (P2)				
stair	525.4 ± 188	293.6 ± 30	459.6 ± 33	$\mathbf{651.0} \pm 90$
pole	846.0 ± 102	562.8 ± 118	876.0 ± 45	$\mathbf{907.2} \pm 2$
stand	846.0 ± 2	840.3 ± 4	846.1 ± 9	$\mathbf{848.5} \pm 13$
hurdle	$\mathbf{597.7} \pm 24$	507.8 ± 24	558.3 ± 147	538.0 ± 95
run	792.6 ± 32	771.5 ± 39	$\mathbf{810.5} \pm 1$	778.1 ± 41
walk	821.2 ± 16	802.6 ± 36	$\mathbf{832.5} \pm 2$	760.2 ± 81
maze	362.7 ± 9	354.7 ± 12	$\mathbf{368.5} \pm 8$	367.4 ± 5

$\lambda = 0.6$ on acrobot-swingup and 0.8 elsewhere. Per-seed values are printed for stair and pole only (Table 12); every cell in this block reads the same five seeds. The 38-task suite uses separate stair runs at $\lambda = 0.5$.

4 Experiments

The analysis identifies a target tradeoff under fixed policies. We now test how the planner and target interact during learning, then evaluate the combined system across benchmarks.

Protocol. We evaluate HumanoidBench `h1` (Sferrazza et al., 2024) at 2M steps, DMControl (Tassa et al., 2018) at 1M, and Meta-World (Yu et al., 2020) and MyoSuite (Caggiano et al., 2022) at 2M. The paired 2×2 study uses separate runs on seven `h1` tasks at 2M and three DMControl tasks at 300k. The paired study reports the same five registered seeds per cell on both DMControl and HumanoidBench (Appendix F). Most suite cells contain three to five runs. Suite run counts can differ between methods; five DMControl TD-MTP entries pool 9–10 runs at a common λ (Appendix D).

All paired training arms evaluate checkpoints with the same reference MPPI planner. They measure changes induced during training, not a deployment-time planner swap. Unless noted otherwise, a run's score is its mean over the last three evaluations; we report the mean across runs. A difference below 3% of the baseline's return or within ± 0.05 success is a tie. This is a descriptive tolerance, not an equivalence test. We analyze the planner–target interaction post hoc. Few seeds limit statistical precision (Agarwal et al., 2021); we report uncertainty where seed-level data are available. Learning curves compare performance at equal numbers of training environment steps. They measure sample efficiency, rather than wall-clock cost.

4.1 The planner–target interaction

To quantify Figure 1, let $R_{p,v}$ be a task's mean return, with binary indicators $p = 1$ for the structured planner and $v = 1$ for the multi-step target. The interaction is the difference between planner effects:

$$\mathcal{I} = (R_{1,1} - R_{0,1}) - (R_{1,0} - R_{0,0}). \tag{12}$$

The method means give positive planner effects on 1 of 10 tasks under one-step targets and 5 of 10 under λ-returns, with $\mathcal{I} > 0$ on 8 of 10. These are descriptive summaries of a post-hoc analysis; tasks that share a benchmark are not independent replications. An additional DMControl seed set changes the last count to 7 of 10 (Appendix F); the task count is therefore sensitive to the run set.

The largest interactions occur on `stair` and `pole`. For `stair`, the four means are 525.4, 293.6, 459.6 and 651.0, giving $\mathcal{I} = 423.2$. Thus the planner reduces mean return with the one-step target and increases it with the multi-step target. Scoring each seed separately, on the same seeds and the

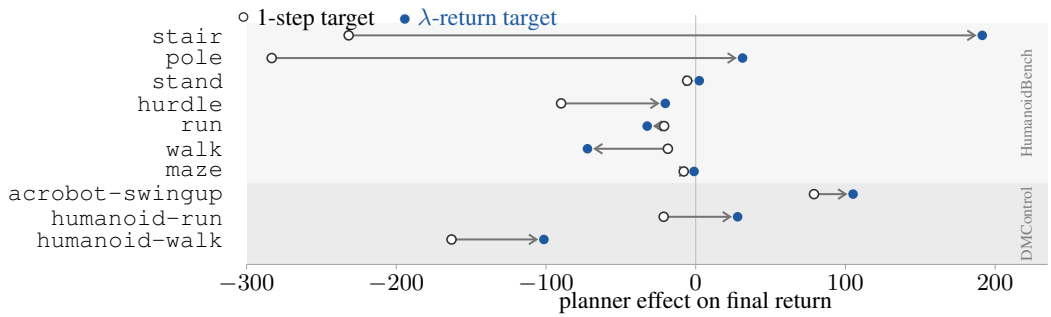

Figure 1: **The planning configuration's effect under each value target.** Each row shows a task in the paired study (protocols P1 and P2 of Table 6; means in Table 1). Open circles show the change in mean return from adding the planner under the one-step target; filled circles show the change under the λ-return target. Arrows show the interaction.

same last-three rule as Table 1, the interaction is positive on every seed of `stair` and on every seed of `pole`. Over the five registered seeds the mean is $+423.2$ on `stair` (95% interval $[+71, +776]$, positive on 5 of 5 seeds) and $+314.4$ on `pole` ($[+147, +482]$, positive on 5 of 5). Both intervals exclude zero. These are unadjusted Student-t intervals for a post-hoc contrast; the seeds are matched across targets by construction (Appendix F). On the remaining eight tasks the interval contains zero; Table 13 gives all ten. Reward scales differ, so we do not average the raw interaction magnitudes across tasks.

Component gains and interaction. A positive interaction need not mean that both components help on their own. Relative to the baseline, the λ-return alone raises mean return on 8 of 10 tasks, with losses on `stair` and `hurdle`; the planner alone raises it on 1. `stair` is also where the interaction is largest: both components alone fall below the baseline and their combination exceeds it. Under the 3% tie tolerance of Table 2, the target alone scores 4 wins, 4 ties, and 2 losses. Table 11 extends this target-only comparison to the manipulation tasks.

Learning curves at 300k steps. Figure 2 shows how the four methods reach their 300k scores. On `humanoid-run`, both multi-step methods gain return earlier than the one-step methods. Adding the planner increases the multi-step method's final mean by 28, but lowers the one-step mean. On `acrobot-swingup`, the leading method changes during training; TD-MTP leads at 300k. The planner improves the score on all five seeds under the multi-step target, compared with three under one-step targets. Sample efficiency depends on the task and training budget. On `humanoid-walk`, the planner lowers the 300k score under both targets (Table 1). Appendix D.1 adds longer-horizon `h1` comparisons and manipulation curves, including delays and failed runs.

What drives the interaction? In the main comparison, the λ-return target also uses a longer replay window. Controls with matched replay windows test which planner changes contribute (Appendix D.4). On `stair` at 2M, extending the one-step replay window changes return by $+66.4$ (95% interval $[-37, +169]$ over ten paired seeds); the available data do not rule out a material window effect. In five-seed controls, the isolated tensor-proposal interaction is -14.6 ($[-107, +78]$) on `stair` and $+7.9$ ($[-72, +88]$) on `humanoid-run`. These intervals do not establish either a proposal benefit or equivalence.

The clearest component evidence concerns elite fitting. With tensor candidates disabled on `stair`, replacing the top-K fit by an all-candidate softmax fit reduces the interaction by a reported mean within-seed share of 46.1% ($[39, 52]$%, five seeds). This concerns a separate experiment with matched replay windows, not the $+423.2$ interaction above. A stochastic-execution control gives an inconclusive change and does not reproduce the reference execution rule exactly. The controls suggest a role for fitting; measuring critic error on selected and unselected candidates would test the proposed explanation that selection amplifies errors which longer targets reduce.

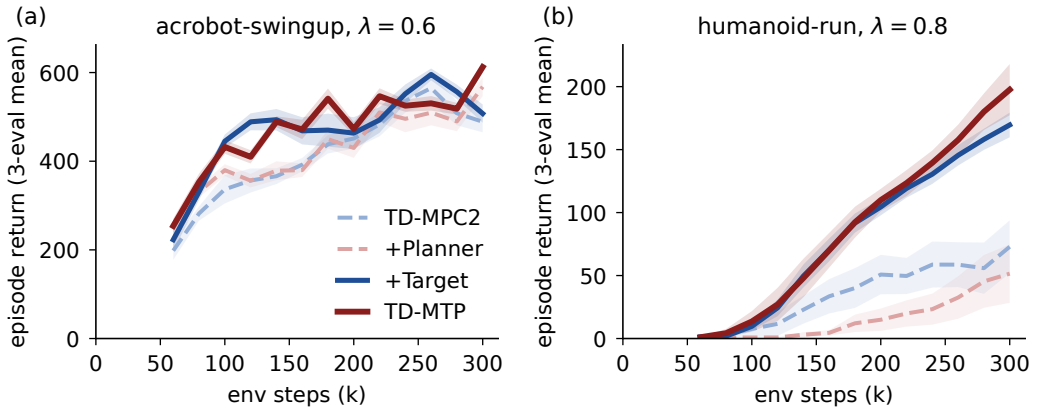

Figure 2: **Learning curves for the main comparison.** Five seeds per method, protocol P1; all checkpoints use the same MPPI evaluation planner. Lines show three-evaluation means averaged across seeds; bands show ± 1 standard error across seeds. Each evaluation uses five episodes. The endpoints match Table 1. $\lambda = 0.6$ on `acrobot-swingup` and 0.8 on `humanoid-run`.

Table 2: **Task outcomes across four benchmarks.** TD-MTP compared with TD-MPC2 on 38 tasks (S1–S5). Bold marks totals. Ties use the descriptive tolerances below, not a significance test.

Benchmark	Tasks	Wins	Ties	Losses	Steps
Episode return: difference below 3% is a tie					
HumanoidBench h1	14	5	8	1	2M
DMControl	8	2	5	1	1M
Success rate: difference within ± 0.05 *is a tie*					
Meta-World A	6	0	4	2	2M
Meta-World B	6	2	2	2	2M
MyoSuite (`-hard`)	4	1	3	0	2M
Total	**38**	**10**	**22**	**6**	—

Five ties are shared ceiling or floor outcomes. Meta-World A and B use different episode caps.

4.2 The combined system across four benchmarks

Task-dependent gains across benchmarks. To assess how often the combined system helps beyond the paired study, we evaluate 38 task–protocol pairs. Table 2 gives 10 wins, 22 ties and 6 losses under the stated tolerance. Examples include `balance_hard` (175 $\rightarrow$ 430), `stair` (490 $\rightarrow$ 664) and `KeyTurn` (0 $\rightarrow$ 0.84 success). Four losses occur on Meta-World, `reach` is the HumanoidBench loss and `finger-turn_hard` the DMControl loss. Counts summarize these evaluated task–protocol pairs; they are not a normalized performance aggregate or evidence of general superiority. Appendix D gives every task score and the available success-rate seeds. These show failed runs and unequal sample counts that benchmark totals alone can hide. Equal final success can also hide slower learning: TD-MTP and TD-MPC2 both solve `stick-pull`, but TD-MTP learns later (Figure 4).

Larger λ reduces return on `reach`. On `reach`, returns fall from 9053 to 7422 to 5240 as λ increases from 0 to 0.5 to 0.8 (n=3 per setting), with the same ordering on both planner arms (Appendix E). Reducing λ from 0.8 to 0.5 recovers 57% of the gap but does not remove it. We therefore treat λ as a task-specific hyperparameter. We select λ by return; this comparison does not measure either term in the theoretical bound.

5 LIMITATIONS

Task and budget dependence. On Meta-World protocol A, the target-only method records no wins, four ties and two losses against the baseline. On `stick-pull`, success falls from 1.000 to 0.622 under the primary scoring rule. The combined method also fails on one of three `pick-place-wall` runs. On MyoSuite `PenTwirl`, the mean across all three TD-MTP runs ties the baseline.

At 4M on `stair`, the interaction remains positive, but TD-MTP no longer leads the baseline. Adding the planner to the multi-step method reduces mean return under both tested λ values. The 4M comparison uses only three seeds and a different run set from the 2M study (Appendix E). It cannot separate a budget effect from seed variability.

Limits of the analysis and experiments. The value bounds concern stationary policies and expected operators. They do not guarantee improvement during nonlinear control learning. The experiments change replay context along with the target, and the planner comparison changes several rules. Matched-window and component controls narrow the attribution, but do not establish a tensor-proposal benefit. Return and success curves do not measure state coverage or isolate improved exploration.

6 RELATED WORK

Structured planning and learned search. TD-MTP inherits its tensor-graph proposals from MTP (Le et al., 2025b). Other structured search approaches include tensor motion planning (Le et al., 2025a), action smoothing (Nagabandi et al., 2020), particle MPC (Lambert et al., 2021) and generated trajectories (Janner et al., 2022; Ajay et al., 2023). Chang et al. (2026) study search-induced overestimation and mitigate it with pessimistic ensemble values. Our comparison tests how this search–learning relationship depends on the replay target.

Multi-step value estimation. TD(λ) and Peng's Q(λ) are established estimators (Sutton, 1988; Peng & Williams, 1996; Kozuno et al., 2021), including prior work on combining λ-returns with replay (Daley & Amato, 2019). Retrace, V-trace and Tree-Backup address off-policy evaluation (Munos et al., 2016; Espeholt et al., 2018; Precup et al., 2000); Q(λ) with off-policy corrections uses a different operator (Harutyunyan et al., 2016). The method of Daley (2026) adjusts trace coefficients to control the tradeoff between off-policy bias and credit assignment. Our termination and truncation indicators handle episode boundaries; they do not correct off-policy bias. MuZero uses stored multi-step targets (Schrittwieser et al., 2020), whereas Dreamer uses imagined λ-returns (Hafner et al., 2025). Model-based value expansion extends targets using predicted transitions (Feinberg et al., 2018; Buckman et al., 2018). We specify the boundary conventions (Pardo et al., 2018) for sequential replay and separate bootstrap sensitivity from behaviour–prior mismatch, including the endpoint retained by finite windows (§3.3, Appendix C).

Planner–prior mismatch. TD-M(PC)^2 constrains policy learning to reduce out-of-distribution value queries (Lin et al., 2026), and BMPC learns from planner outputs (Wang et al., 2025); BOOM aligns the policy with the planner's action distribution under a value-weighted loss (Zhan et al., 2025). Our ε is a worst-state total-variation distance; it is not the regularization objective or empirical discrepancy used by those methods. These methods change policy learning; TD-MTP changes the critic target. Whether their gains combine is a question for matched experiments. MTP (Le et al., 2025b) uses a supplied dynamics model without learning a critic; it does not study this training interaction.

7 CONCLUSION

Using TD-MTP, we find that a planning configuration's training effect can depend on its value target, with the strongest evidence on `stair`. Controls suggest a role for elite fitting; the tensor-proposal contribution remains unresolved. Our fixed-policy analysis separates bootstrap sensitivity from policy mismatch, including finite-window endpoint effects. Gains vary with task and budget; an early lead need not persist. These findings support evaluating planner and target choices jointly during training.

AI USE STATEMENT

We used AI tools for editing, literature cross-checks, mathematical review and verification code, and feedback on experimental methodology and interpretation. The authors are responsible for all methods, proofs, experiments, and reported results.

ETHICS STATEMENT

We study simulated continuous-control agents on public benchmarks. The work involves no human subjects, no personally identifying information, and no new data collection, and we release no datasets. These experiments do not establish safety in physical deployment.

REPRODUCIBILITY STATEMENT

Section 3 specifies the planner and replay target. Appendix A gives the implementation settings. Appendices B and C give the assumptions and complete derivations for the theoretical results. Protocols, per-task results and available seed-level scores appear in Appendix D. Appendix F states the software versions, run-selection choices, and limits to exact reproduction.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8234–8244, 2018.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSuite: A contact-rich simulation suite for musculoskeletal motor control. In *Learning for Dynamics and Control (L4DC)*, pp. 492–507, 2022.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2601.21306.

Brett Daley. Gated q-learning: Add off-policy bias to taste. *Reinforcement Learning Journal*, 2026. URL https://arxiv.org/abs/2607.28916.

Brett Daley and Christopher Amato. Reconciling λ-returns with experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1810.09967.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, pp. 1407–1416, 2018.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value expansion for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html.

Anna Harutyunyan, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. Q(λ) with off-policy corrections. In *Algorithmic Learning Theory (ALT)*, volume 9925 of *Lecture Notes in Computer Science*, pp. 305–320, 2016.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, pp. 9902–9915, 2022.

Tadashi Kozuno, Yunhao Tang, Mark Rowland, Rémi Munos, Steven Kapturowski, Will Dabney, Michal Valko, and David Abel. Revisiting Peng's Q(λ) for modern reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 5794–5804, 2021. URL https://proceedings.mlr.press/v139/kozuno21a.html.

Alexander Lambert, Adam Fishman, Dieter Fox, Byron Boots, and Fabio Ramos. Stein variational model predictive control. In *Conference on Robot Learning (CoRL)*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1278–1297, 2021.

An T. Le, Kay Hansel, João Carvalho, Joe Watson, Julen Urain, Armin Biess, Georgia Chalvatzaki, and Jan Peters. Global tensor motion planning. *IEEE Robotics and Automation Letters*, 2025a. doi: 10.1109/LRA.2025.3575307.

An T. Le, Khai Nguyen, Minh Nhat Vu, João Carvalho, and Jan Peters. Model tensor planning. *Transactions on Machine Learning Research (TMLR)*, 2025b. URL https://openreview.net/forum?id=fk1ZZdXCE3.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. TD-M(PC)^2: Improving temporal difference MPC through policy constraint. In *Learning for Dynamics and Control (L4DC)*, volume 331 of *Proceedings of Machine Learning Research*, pp. 705–736, 2026. URL https://proceedings.mlr.press/v331/lin26a.html.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1054–1062, 2016.

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1101–1112, 2020. URL https://proceedings.mlr.press/v100/nagabandi20a.html.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 4045–4054, 2018. URL https://proceedings.mlr.press/v80/pardo18a.html.

Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22(1–3):283–290, 1996. URL https://link.springer.com/article/10.1007/BF00114731.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766, 2000.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoid-Bench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. URL https://link.springer.com/article/10.1007/BF00115009.

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

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2503.18871.

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 *Conference on Robot Learning (CoRL)*, pp. 1094–1100, 2020.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=zNqDCSokDR.

A Implementation conventions

Algorithm 2 Multi-step targets for one sampled replay window.

Require: Window $(z_t, a_t, r_t, z_t^+, d_t, u_t)_{t=0}^{L-1}$, supervised set $\mathcal{T}$, λ, γ, Q^-, π
1: **for** $t = 0, \dots, L-1$ **do**
2: Set $\nu_t = 1 - d_t$ and $\chi_t = (1 - d_t)(1 - u_t)$
3: Draw $a_t^+ \sim \pi(\cdot \mid z_t^+)$ and set $q_{t+1} = Q^-(z_t^+, a_t^+)$
4: **end for**
5: $G_L \leftarrow q_L$
6: **for** $t = L-1$ **down to** 0 **do**
7: $G_t \leftarrow r_t + \gamma\nu_t[(1 - \lambda\chi_t)q_{t+1} + \lambda\chi_t G_{t+1}]$
8: **end for**
9: **return** $\{\mathrm{stopgrad}(G_t) : t \in \mathcal{T}\}$

We generate policy-prior candidates by rolling out π under $\hat{f}$ from the current latent. The projection Ψ maps standard deviations to layerwise search widths: it is a design rule, not covariance propagation through a stochastic transformation.

Configuration of every training run. Tables 3, 4 and 5 give the settings of the main training runs. Only λ was chosen per task. Besides the planner, two training conventions differ from reference TD-MPC2. The reference planner uses the TD-MPC2 elite fit, with $K = 64$ elites and inverse temperature $\tau = 2$; the structured planner uses $K = 32$ and $\tau = 2$. We use 1000 random seed steps instead of $\max(1000, 5L_{\mathrm{ep}})$, where L_{ep} is episode length. The reference rule gives 2500 seed steps on DMControl. We also fix $\gamma = 0.99$ instead of deriving it from episode length. The discount agrees on DMControl, where the reference rule also gives 0.99. HumanoidBench uses a different episode length; its suite comparison uses our port for both methods.

Trust-region schedule. The band floor follows a cosine schedule: $\rho_i = \rho_{\mathrm{end}} + \frac{1}{2}(\rho_{\mathrm{start}} - \rho_{\mathrm{end}})\big(1 + \cos(\pi\,\frac{i-1}{I-1})\big)$, giving $\rho_{1:6} = 0.500,\ 0.457,\ 0.345,\ 0.205,\ 0.093,\ 0.050$. With $c_e = 0.5$, $\varsigma \le 2$ and $\rho_i \in [0.05, 0.5]$ the band width stays in $[0.075, 1.5]$, so the clamp to $[0.05, 2]$ never binds.

Interpolation at $H = M = 3$. The linear matrix is built on $\lceil H/(M-1)\rceil(M-1) = 4$ parameter values over $[0, M-1)$ and truncated to H rows, so the three actions sit at parameters $0, \frac{1}{2}, 1$ and the third control-point layer receives zero weight. Each tensor candidate therefore depends on two layers, giving at most $16^2 = 256$ distinct sequences. The third layer is drawn but never used in the main study, component controls or benchmark experiments.

Table 3: **Implementation settings, part 1 of 3: planning.** Settings of the planning loop. "Both arms" entries are identical in the TD-MPC2 and TD-MTP arms of every paired block. $\Phi_{\mathcal{N}}^{-1}$ is the standard-normal quantile function. Tables 4 and 5 give the planner-specific and the learning settings.

setting	value
planning, both arms	
population S; iterations I	512; 6
elites K	64 reference planner; 32 structured planner
policy-prior rollouts	24 per decision, sampled from π, drawn once and reused in every iteration
horizon $H = H_{\text{train}}$; discount γ	3; 0.99, fixed for every task
standard-deviation clamp; initial value	$\varsigma \in [0.05, 2]$; $\varsigma^{(0)} = 2$ at every decision
terminal value in Eq. 2	mean of the five online critic heads; no termination head is trained (continue-loss weight 0), so $\hat{d}_h \equiv 0$ and $b_h \equiv 1$ in planning
clipping	every candidate sequence, every control point and the executed action are clipped to $[-1, 1]$

Table 4: **Implementation settings, part 2 of 3: the two planners.** Settings specific to each planner; the shared planning loop is in Table 3.

setting	value
structured planner (TD-MTP arms)	
M; N; interpolation	3; 16; linear
Φ at $H = M = 3$	rows $(1, 0, 0)$, $(\frac{1}{2}, \frac{1}{2}, 0)$, $(0, 1, 0)$: not Id_3 (see text)
Ψ	one-hot rows at steps $\lfloor m(H-1)/(M-1) \rceil$, $m = 0, \ldots, M-1$; equals Id_3 at $H = M = 3$
c_e; ρ_{start}; ρ_{end}; schedule	0.5; 0.5; 0.05; cosine (below), annealed across the I iterations of each decision
band clamp	$c_e(\Psi\varsigma^{(i-1)})_m + \rho_i$ is clipped to $[0.05, 2]$ before multiplying ξ
quantile grid	$\xi_j = \Phi_{\mathcal{N}}^{-1}\big((j + \frac{1}{2})/16\big)$, $j = 0, \ldots, 15$: ± 0.078, ± 0.237, ± 0.402, ± 0.579, ± 0.776, ± 1.010, ± 1.318, ± 1.863; permuted independently per layer and action coordinate at every iteration
β; minimum local draws f; P	0.5 with $\beta_k = \beta \max(0, 1 - 2k/T)$, held constant across the I iterations of one decision; 1; $P = S - 25 = 487$ ($n_{\text{tensor}} = 244$ at $\beta_k = 0.5$)
$\mathcal{D}_{\text{loc}}$	independent standard normal, redrawn at every iteration
elite fit	top-K of all S candidates; $\tau = 2$ (softmax of $-\text{cost}/0.5$); weighted variance divided by $1 - \sum_p w_p^2$; no momentum ($\alpha = 0$)
executed action	first action of $a^\star$, the best-scoring candidate of the final population; no collection noise
warm start	$\mathbf{m}$ and $a^\star$ shifted by one step with the last entry repeated; ς reset to 2; $\mathbf{m} = a^\star = 0$ at the first step of every episode
reference planner (our JAX arms; also their evaluation planner)	
elite fit	top-K; $\tau = 2$; variance $+10^{-6}$, no reliability factor; same clamp
executed action	collection: one elite sampled with probability w_p, plus $\mathcal{N}(0, \varsigma_0^2)$ noise, clipped; evaluation: the best elite, no noise
warm start	$\mathbf{m}$ shifted by one step with the last entry 0; ς reset to 2; $\mathbf{m} = 0$ at the first step of every episode
terminal action a_H^π	a sample from π (the structured planner uses the mean of π)

Table 5: **Implementation settings, part 3 of 3: training and evaluation.** Shared learning and evaluation settings, with target-dependent replay context stated explicitly. All retained comparisons use $H_{\text{train}} = 3$.

setting	value
learning, both arms	
window; supervised set	$L = H_{\text{train}} + \ell$ with $\ell = 10$ when $\lambda > 0$ and $\ell = 0$ for the one-step target; $\mathcal{T} = \{0, 1, 2\}$, the tail is never regressed on
temporal loss weights	$0.5^t / \sum_{t<3} 0.5^t = (4/7, 2/7, 1/7)$ on the consistency, reward and value losses
loss scales	consistency 20, reward 0.1, value 0.1; reward and value use two-hot cross-entropy over 101 symlog bins on $[-10, 10]$
target critic Q^-	minimum over two heads drawn without replacement from the five-head target ensemble, one draw per update; a_t^+ sampled from the current π
target update; optimizer	Polyak rate 0.01 per update; AdamW 3×10^{-4} (weight decay 10^{-4}) on the world-model heads, Adam 10^{-4} on the encoder, gradient norm clipped at 20
batch; update ratio; warm-up	256 windows; one update per environment step; 1000 uniformly random seed steps
model	latent 512, SimNorm groups of 8, encoder $\text{enc_dim} = 256$ with 2 layers (one 256-unit hidden layer, then a 512-d output), five critics of two 512-unit layers with dropout 0.01 and a 101-way head, entropy coefficient 10^{-4}
evaluation	reference MPPI, best elite, no noise; five episodes per checkpoint at the 20k or 40k cadence in Table 6; evaluation seeds $0, 1000, 2000, 3000, 4000$, the same five for every run (Appendix F)

Incumbent, prior and noise refresh. $a^\star$ occupies one slot of the pool at every iteration and is re-scored with the same model call as the new candidates; the rollout key is held fixed across iterations, so its score is reproduced exactly. Local Gaussian noise is independent across time. Tensor actions have the temporal correlations derived in Appendix B.

Fit and execution. Elites are the K highest-scoring of the S candidates, priors and $a^\star$ included, and $a^\star$ is the top-scoring one. The fit computes the weighted moments in Eq. 5, divides the variance by $1 - \sum_p w_p^2$, and clamps its square root to $[0.05, 2]$ per coordinate, carrying it forward as $\varsigma^{(i)}$. The TD-MTP arms collect with the best-candidate rule, and in every paired, seven-condition and component-control block the TD-MTP arm is evaluated by the reference MPPI planner, so the TD-MTP execution rule enters the results through the collected data only. The action is a deterministic function of a sampled population, but the resulting behavior policy remains stochastic because candidate generation is random. For finite scores and $K > 1$, every softmax weight is strictly between zero and one. Thus $1 - \sum_p w_p^2 > 0$ in exact arithmetic. Floating-point rounding still requires a safeguard before division. The reported runs floor $1 - \sum_p w_p^2$ at 10^{-6} before inverting it and reject $\tau > 100$ at construction. The temperature cap alone cannot prevent weight concentration: large score gaps can make one weight arbitrarily close to one. The denominator floor provides the numerical safeguard; the variance clamp follows the division.

Why the fitted proposal is not a behavior likelihood. The planner samples a population, scores it, selects elites, and executes one action. The executed-action distribution depends on all these random operations. The fitted Gaussian density alone is therefore not the behavior likelihood needed by Retrace or V-trace (Munos et al., 2016; Espeholt et al., 2018). An estimated behavior model would require a separate approximation and evaluation. The discrete-action Tree-Backup coefficient also uses probability mass, not a continuous density (Precup et al., 2000). A density can exceed one and depends on the action units. These distinctions explain our analysis of the uncorrected replay target; they do not show that off-policy corrections are ineffective.

Shared training settings. Our training comparisons use a shared JAX implementation of TD-MPC2. This includes the paired, component-control and suite runs, except the labeled reference PyTorch entries. Those entries follow native TD-MPC2 conventions: an episodic buffer, unnormalized temporal loss weights, and a discount derived from episode length. *Replay context.* At $H_{\text{train}} = 3$ a $\lambda > 0$ run samples windows of $L = 13$ transitions and the one-step method samples $L = 3$. The matched-window control (condition A0 of the seven-condition study) samples $L = 13$ and computes the one-step target from the first three positions, discarding the tail. *Supervised positions.* $\mathcal{T} = \{0, \ldots, H_{\text{train}} - 1\}$ in every mode. Main training comparisons use $H_{\text{train}} = 3$. The critic input at position t is the unrolled latent $\hat{z}_t$ with $\hat{z}_0 = z_0$, as in TD-MPC2, and the drawn target pair is used at every position of the length-L window. The two targets differ only in L and in the recursion. At $\lambda = 0$, the target reduces to the one-step target by Corollary 2.

Episode boundaries. Environments run under a next-step autoreset: the step that ends an episode returns the true final observation, which is stored as the successor z_t^+ together with d_t and u_t; the synthetic reset step is not stored. DMControl uses action repeat 2 and a 500-decision-step cap, matching the reference implementation. Thus, the 1M budgets need no frame conversion. Episodes end by truncation after 1000 simulator steps. HumanoidBench uses no action repeat and truncates at 1000 steps unless the task terminates earlier. The height threshold is 0.2 for `walk`, `run`, `hurdle`, `maze` and `stand`. It is 0.5 for `pole`; `stair` instead terminates when torso uprightness falls below 0.1. These terminations enter the replay target through d_t. Meta-World protocol A uses action repeat 2 and a cap of 100 decision steps. Protocol B uses no action repeat and a 500-step cap; success does not terminate the episode.

Replay windows are sampled sequentially and can cross boundaries. Latent unrolls do not restart inside a window. The ending transition remains supervised; subsequent transitions are masked out of the consistency, reward and value losses. The return recursion applies Equation 6, including across the extra replay context. All bootstrap latents in that context encode stored observations.

Beyond the termwise $\lambda = 0$ identity, reproducing the reference training trajectory requires matching replay sampling, random draws and update operations. At $\beta = 0$, tensor candidates disappear but the other configuration changes remain, so this is not the reference MPPI method.

B What the tensor graph changes

The path count in Equation 3 describes possible combinations of nodes. It does not describe their geometry. We derive the action covariance to show how interpolation shapes the candidates. We then bound coverage of an action region using the nodes of the actual finite graph. Both statements concern proposals before scoring.

Temporal action covariance. Fix one graph C at a planner iteration and suppress its iteration index. Let $U_m = C_{m,J_m}$, where the layer indices J_m are independent and uniform on $\{1, \ldots, N\}$. Treat each node as a column vector and define its layer mean and covariance:

$$\bar{C}_m = \frac{1}{N} \sum_{j=1}^{N} C_{m,j}, \qquad V_m = \frac{1}{N} \sum_{j=1}^{N} (C_{m,j} - \bar{C}_m)(C_{m,j} - \bar{C}_m)^\top. \tag{13}$$

Since $a_h = \sum_m \Phi_{hm} U_m$, its conditional moments are

$$\mathbb{E}[a_h \mid C] = \sum_m \Phi_{hm} \bar{C}_m, \qquad \mathrm{Cov}(a_h, a_{h'} \mid C) = \sum_m \Phi_{hm} \Phi_{h'm} V_m. \tag{14}$$

The mean follows by linearity. For the covariance, expand the two centered sums. Cross-layer terms vanish because the layer indices are independent given C. Terms within a layer give V_m. Coordinates within a selected node need not be independent.

Shared control points therefore induce temporal correlations. For the reported $H = M = 3$ interpolation,

$$a_0 = U_1, \qquad a_1 = \tfrac{1}{2}(U_1 + U_2), \qquad a_2 = U_2. \tag{15}$$

The middle action is fixed by the endpoints, and the third graph layer is inactive. Tensor proposals impose a temporal constraint that can help search or exclude useful sequences. The local Gaussian and policy-prior proposals broaden the candidate pool beyond this constraint.

Coverage of an action region. A nonzero trust-region width does not ensure that a finite graph contains nodes in every region of interest. The next bound states the node condition explicitly. Let $\mathcal{A} \subseteq [-1,1]^A$ be a box and let $\mathcal{B} = \{a : a_h \in \mathcal{A} \text{ for all } h\}$ be the corresponding sequence region. This region describes actions only; it need not be feasible or rewarding under the dynamics.

Proposition 2 (Coverage conditional on the graph)**.** *Fix C and let Φ have non-negative entries with each row summing to one. Define the active layers and their joint node fractions by*

$$\mathcal{M}_\Phi = \{m : \Phi_{hm} > 0 \text{ for some } h\}, \qquad q_m = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\{C_{m,j} \in \mathcal{A}\}, \qquad p_C = \prod_{m \in \mathcal{M}_\Phi} q_m. \tag{16}$$

Draw $n \in \{0, 1, \ldots\}$ paths independently as in Equation 3, and let Z count those in $\mathcal{B}$. With zero powers equal to one,

$$\mathbb{E}[Z \mid C] \geq np_C, \qquad \Pr(Z \geq 1 \mid C) \geq 1 - (1 - p_C)^n. \tag{17}$$

Proof. If each selected active node lies in $\mathcal{A}$, every interpolated action also lies there by convexity. Independence across layer indices gives this sufficient event probability p_C. Hence a single path has hit probability $q_C \geq p_C$. The n independent paths give $\mathbb{E}[Z \mid C] = nq_C$ and $\Pr(Z \geq 1 \mid C) = 1 - (1 - q_C)^n$. Both expressions are nondecreasing in q_C. □

For Equation 15, the sufficient event is also necessary: both endpoints must lie in $\mathcal{A}$. Thus $q_C = q_1 q_2$, and both bounds are equalities. For other interpolation maps, a zero q_m can make the bound vacuous even if some paths enter $\mathcal{B}$.

The fractions q_m count whole action vectors. They cannot generally be replaced by products of coordinate fractions. For example, take nodes $(-1, 1)$ and $(1, -1)$ in one layer, with $\mathcal{A} = [-1, 0]^2$. Each coordinate has marginal coverage $1/2$, but no node is in the box. Independent permutations used to construct the graph do not imply independent coordinates when selecting a node from that fixed graph.

In Algorithm 1, $n = n_{\text{tensor}}$ for that iteration. The bound is zero when tensor sampling ends halfway through training. It does not describe elite selection, the executed-action policy μ, or learning performance. Proposal coverage and replay-target analysis describe separate steps in TD-MTP; their interaction during learning is tested empirically.

C Replay-target analysis

We first expand the finite replay target, including episode boundaries. We then identify the policy evaluated by the infinite-window target and use one contraction argument for both value-error bounds. All operators below act on bounded measurable action-value functions with the sup norm.

C.1 Finite returns and episode boundaries

Let $n = L - t$ be the number of transitions remaining in the window. The product Γ_k below records whether the return can reach transition $t + k$.

Proposition 3 (Return expansion across episode boundaries)**.** *Let $n \geq 1$, $\Gamma_0 = 1$ and $\Gamma_k = \prod_{j=0}^{k-1} \chi_{t+j}$. With the indicators in Equation 6, the target in Equation 7 is*

$$\begin{aligned} G_t = &\sum_{k=0}^{n-1} (\gamma\lambda)^k \Gamma_k r_{t+k} \\ &+ \sum_{k=1}^{n-1} \gamma^k \lambda^{k-1} \Gamma_{k-1} \nu_{t+k-1} (1 - \lambda\chi_{t+k-1}) q_{t+k} + \gamma^n \lambda^{n-1} \Gamma_{n-1} \nu_{L-1} q_L. \end{aligned} \tag{18}$$

Empty sums are zero, empty products are one, and zero powers equal one.

Proof. Since $\nu_s\chi_s = \chi_s$, the recursion is $G_s = b_s + c_s G_{s+1}$, where $b_s = r_s + \gamma\nu_s(1-\lambda\chi_s)q_{s+1}$ and $c_s = \gamma\lambda\chi_s$. Repeated substitution gives

$$G_t = \sum_{k=0}^{n-1} \left(\prod_{j=0}^{k-1} c_{t+j} \right) b_{t+k} + \left(\prod_{j=0}^{n-1} c_{t+j} \right) q_L.$$

This identity holds for $n = 1$; substituting G_{t+1} proves the induction step. Each product of k coefficients equals $(\gamma\lambda)^k\Gamma_k$. Substitute b_s and combine the two endpoint terms using $\nu_{L-1}\chi_{L-1} = \chi_{L-1}$. Their total coefficient on q_L is

$$\gamma^n\lambda^{n-1}\Gamma_{n-1}\nu_{L-1}\big[(1-\lambda\chi_{L-1}) + \lambda\chi_{L-1}\big],$$

which gives Equation 18. □

Without boundaries, the expansion has a simpler interpretation: the target is a weighted average of returns with different bootstrap times. These weights will also control its sensitivity to bootstrap perturbations.

Corollary 1 (Finite mixture without boundaries)**.** *If $\nu_s = \chi_s = 1$ throughout the window, then $G_t = \sum_{m=1}^{n} w_m G_t^{(m)}$, where*

$$w_m = \begin{cases} (1-\lambda)\lambda^{m-1}, & m < n, \\ \lambda^{n-1}, & m = n, \end{cases} \qquad G_t^{(m)} = \sum_{j=0}^{m-1} \gamma^j r_{t+j} + \gamma^m q_{t+m}.$$

Proof. The geometric sum gives $\sum_{m=j+1}^{n} w_m = \lambda^j$ for $0 \le j < n$. Thus the reward coefficient is $\gamma^j\lambda^j$, and each bootstrap coefficient is $w_m\gamma^m$. These are exactly the coefficients in Equation 18. In particular, the weights are non-negative and sum to one. □

Corollary 2 (One-step identity)**.** *For any boundary flags, $\lambda = 0$ gives $G_t = r_t + \gamma\nu_t q_{t+1}$.*

Proof. Set $\lambda = 0$ in Equation 7; its continuation term vanishes. □

If the first boundary is at $t + j$, then $\Gamma_k = 0$ for $k > j$. The remaining bootstrap at that boundary is multiplied by ν_{t+j}. A termination therefore removes it; an external truncation retains it. This proves the boundary cases in §3.2 without treating a reset observation as a successor.

C.2 Infinite-window fixed point and error bound

Fix stationary policies μ and π, bounded rewards, $0 < \gamma < 1$ and $\lambda \in [0, 1]$. Write $T^\kappa Q = r + \gamma P^\kappa Q$ for the Bellman operator, with P^κ defined in §3.3. Probability averaging is nonexpansive: $\|P^\kappa f\|_\infty \le \|f\|_\infty$. Without boundaries, the expected return obeys

$$T_\lambda Q = r + \gamma\lambda P^\mu T_\lambda Q + \gamma(1-\lambda)P^\pi Q.$$

The series $\sum_{j\ge0}(\gamma\lambda P^\mu)^j$ converges in operator norm. It gives the inverse in Equation 8, bounded by $1/(1-\gamma\lambda)$. For any bounded Q and R,

$$\|T_\lambda Q - T_\lambda R\|_\infty \le \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\|Q-R\|_\infty = \eta_\lambda\|Q-R\|_\infty.$$

Since $\eta_\lambda < 1$, successive iterates converge uniformly to a unique bounded fixed point Q_λ. To identify the policy it evaluates, substitute $T_\lambda Q_\lambda = Q_\lambda$ into the recursion:

$$Q_\lambda = r + \gamma\big[\lambda P^\mu + (1-\lambda)P^\pi\big]Q_\lambda = T^{\lambda\mu+(1-\lambda)\pi}Q_\lambda. \tag{19}$$

Thus $Q_\lambda = Q^{\lambda\mu+(1-\lambda)\pi}$ (Kozuno et al., 2021). The mixture chooses between policies at each successor state. Its value need not equal $\lambda Q^\mu + (1-\lambda)Q^\pi$, since those choices change future state visitation.

A common error inequality. We compare the perturbed fixed point with a reference value by separating the error in one update from the error propagated through later updates. Let T be an η-contraction, with $0 \le \eta < 1$, and let $\delta \ge 0$. Suppose $\|\widehat{T}Q - TQ\|_\infty \le \eta\delta$ for every Q, and $\widehat{Q} = \widehat{T}\widehat{Q}$. For any reference function Q_0, the triangle inequality gives

$$\|\widehat{Q} - Q_0\|_\infty \le \eta\delta + \eta\|\widehat{Q} - Q_0\|_\infty + \|TQ_0 - Q_0\|_\infty.$$

Rearranging yields

$$\|\widehat{Q} - Q_0\|_\infty \le \frac{\eta\delta + \|TQ_0 - Q_0\|_\infty}{1-\eta}. \tag{20}$$

The residual measures how much one target update changes the reference value function. The factor $1/(1-\eta)$ accounts for repeated propagation of these errors. No contraction assumption on $\widehat{T}$ is needed, only the stated bounded fixed point.

Proof of Proposition 1. A bootstrap perturbation of at most δ contributes at most $\eta_\lambda\delta$ to one update. To use Equation 20 with $Q_0 = Q^\pi$, subtract Q^π from Equation 8 and use $r = Q^\pi - \gamma P^\pi Q^\pi$:

$$T_\lambda Q^\pi - Q^\pi = (\mathrm{Id} - \gamma\lambda P^\mu)^{-1}\gamma\lambda(P^\mu - P^\pi)Q^\pi.$$

For bounded measurable f, let $l = \inf f$ and $u = \sup f$. The identity $f(a) = l + \int_l^u \mathbf{1}\{f(a) > x\}\,dx$ gives

$$|\mathbb{E}_\mu f - \mathbb{E}_\pi f| \le \int_l^u |\mu(f > x) - \pi(f > x)|\,dx \le \mathrm{TV}(\mu,\pi)\,\mathrm{span}(f),$$

by the definition of total variation as a supremum over measurable sets. With ε and V_{sp} from Proposition 1, it follows that

$$\|(P^\mu - P^\pi)Q^\pi\|_\infty \le \varepsilon V_{\mathrm{sp}}, \qquad \|T_\lambda Q^\pi - Q^\pi\|_\infty \le \frac{\gamma\lambda}{1-\gamma\lambda}\varepsilon V_{\mathrm{sp}}.$$

Insert this residual and η_λ into Equation 20. Since $1 - \eta_\lambda = (1-\gamma)/(1-\gamma\lambda)$, the result is $\gamma[(1-\lambda)\delta + \lambda\varepsilon V_{\mathrm{sp}}]/(1-\gamma)$. This is Equation 9; its stated slope follows by differentiation. □

C.3 Finite-window error bound

A finite window retains its endpoint bootstrap, including at $\lambda = 1$. We now derive Equation 11 under the same assumptions and with no boundaries inside the window. The expected m-step return and its mixture are

$$U_m Q = \sum_{j=0}^{m-1} \gamma^j (P^\mu)^j r + \gamma^m (P^\mu)^{m-1} P^\pi Q, \qquad T_{\lambda,n} = \sum_{m=1}^{n} w_m U_m, \tag{21}$$

where w_m are the weights in Corollary 1. The first action is fixed; the next $m-1$ actions follow μ, and the endpoint bootstrap averages under π. Only this endpoint term depends on Q. Thus $T_{\lambda,n}$ is a contraction with factor $\eta_{\lambda,n} = \sum_{m=1}^{n} w_m\gamma^m \le \gamma < 1$. Bootstrap perturbations bounded by δ change one update by at most $\eta_{\lambda,n}\delta$.

To compute the residual, write $D = (P^\mu - P^\pi)Q^\pi$ and $R_m = U_m Q^\pi - Q^\pi$. The one-step Bellman identity gives $R_1 = 0$. Since $U_{m+1}Q = r + \gamma P^\mu U_m Q$, we have $R_{m+1} = \gamma D + \gamma P^\mu R_m$. Expanding this recursion gives

$$U_m Q^\pi - Q^\pi = \sum_{j=1}^{m-1} \gamma^j (P^\mu)^{j-1}(P^\mu - P^\pi)Q^\pi. \tag{22}$$

Nonexpansiveness and the total-variation bound above give

$$\|U_m Q^\pi - Q^\pi\|_\infty \le \varepsilon V_{\mathrm{sp}} \sum_{j=1}^{m-1} \gamma^j = \frac{\gamma - \gamma^m}{1-\gamma}\varepsilon V_{\mathrm{sp}},$$

$$\|T_{\lambda,n} Q^\pi - Q^\pi\|_\infty \le \frac{\gamma - \eta_{\lambda,n}}{1-\gamma}\varepsilon V_{\mathrm{sp}}.$$

The second line averages the first with weights w_m, which sum to one. Equation 20 now gives Equation 11 directly. At $n = 1$, $\eta_{\lambda,1} = \gamma$ and the mismatch term vanishes. The geometric sum gives $\eta_{\lambda,n} \to \eta_\lambda$ as $n \to \infty$, recovering the infinite-window bound.

Why the endpoint matters. Let $Q_{\lambda,n}$ be the unperturbed finite-window fixed point. Taking $Q_0 = Q_{\lambda,n}$ in Equation 20 removes the residual and yields

$$\|\widehat{Q}_{\lambda,n} - Q_{\lambda,n}\|_\infty \leq \frac{\eta_{\lambda,n}}{1-\eta_{\lambda,n}}\delta. \tag{23}$$

At $\lambda = 1$, this coefficient is $\gamma^n/(1-\gamma^n)$. It is attained in a one-state, one-action, zero-reward MDP with a constant bootstrap error δ: $\widehat{Q} = \gamma^n(\widehat{Q} + \delta)$, so $\widehat{Q} = \gamma^n\delta/(1-\gamma^n)$. A finite multi-step target therefore remains sensitive to its endpoint value.

Interpreting the bound in an experiment. Each finite-window bound fixes the remaining length n. The implemented loss uses several positions: without boundaries, $L = 13$ gives $n = 13, 12, 11$ at $t = 0, 1, 2$. Their shared-critic update is not an iteration of one fixed-n operator. The stationary-policy assumption also abstracts away changes during training and the planner's warm-start memory; freezing network weights alone does not remove that memory.

The bound separates two error sources, but does not provide measured values for them. Ensemble disagreement is not the uniform bootstrap perturbation δ. A discrepancy measured on sampled states does not certify the worst-state policy distance ε. These diagnostics can test hypotheses about critic error and planner–prior mismatch, but cannot be substituted into the bound without further assumptions.

D Results in full

Summaries in this appendix are computed from unrounded values while the cells they summarize are displayed rounded, so recomputing a mean from printed cells can differ in the last digit.

D.1 Learning curves and sample efficiency

Final performance and sample efficiency answer different questions. The first asks what an agent achieves at the training budget. The second asks how its performance changes as it collects more transitions. Figure 2 shows two tasks from the five-seed P1 comparison. Figure 3 extends the main comparison to `stair` and `pole` at 2M steps. These two panels use the same five registered seeds per method that the endpoint table reports. Figure 4 shows every task in Meta-World protocol A, with the same seed counts as Table 8. All axes count training environment steps; evaluation episodes do not enter replay.

Reading the curves. The P1 and P2 panels average three evaluations per seed, then average across seeds. Their endpoints use the last three evaluations. Bands show one standard error of these seed values, not a confidence interval for a method difference. Meta-World curves use the raw five-episode success rate at each checkpoint, with no temporal smoothing. The evaluation grids and episode protocols are in Table 6. An early lead can disappear, and a higher endpoint need not imply a lead throughout training. Exact first-crossing times require the unsmoothed evaluations. We make no aggregate speedup claim from these selected return curves. The reported figure-generation procedure uses the same run files as the `h1` and Meta-World tables and checks their last-three-evaluation endpoints against the printed cells. Endpoint agreement alone does not verify the run mapping or the seeds behind each band; those checks require the original logs. Full histories also exist for `humanoid-walk` on the five P1 seeds of its Table 1 row, and for MyoSuite `PenTwirl` and `Pose`; they are not plotted, so the sample-efficiency reading here covers the tasks shown and no others.

Average success during training. To summarize each manipulation curve, let $s_{i,j} \in [0,1]$ be the success rate for seed i at checkpoint j. We report

$$\bar{S} = \frac{1}{nJ}\sum_{i=1}^{n}\sum_{j=1}^{J} s_{i,j}, \qquad J = 50, \quad t_j = 40{,}000j. \tag{24}$$

This is the mean success over the evaluation grid up to 2M steps, not a trapezoidal integral. It weights each checkpoint equally and rewards early, sustained success. Table 7 gives all 12 Meta-World tasks, keeping the two episode protocols separate. On `push-back`, the target-only method raises

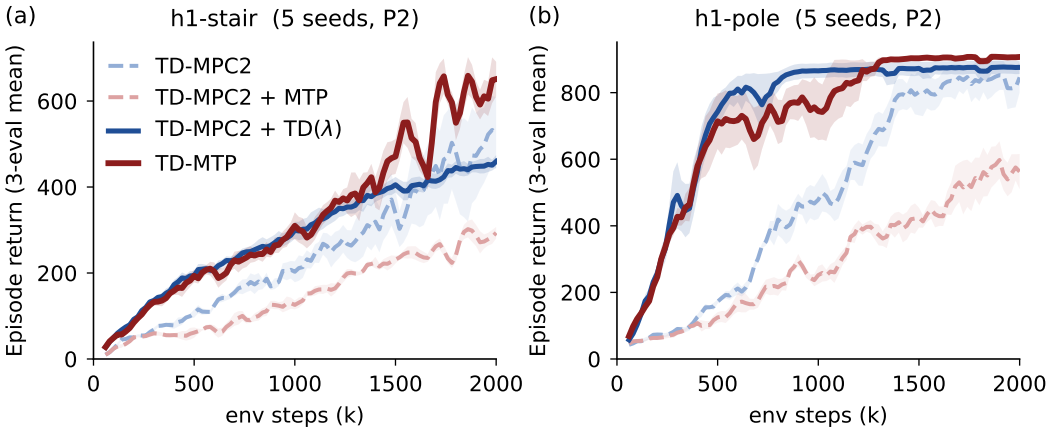

Figure 3: **Longer learning trajectories under both planners and targets.** Protocol P2, $\lambda = 0.8$, the five registered seeds: 1865, 5927, 6108, 3 and 4. The line and band conventions follow Figure 2; all arms use the same MPPI evaluation planner. The last three means are $525.4/293.6/459.6/651.0$ on `stair` and $846.0/562.8/876.0/907.2$ on `pole`, in legend order. These are the means Table 1 reports, read from the same five runs. On `stair`, the combined method separates from the target-only method late in training. On `pole`, the target-only method rises earlier, while TD-MTP ends higher at 2M.

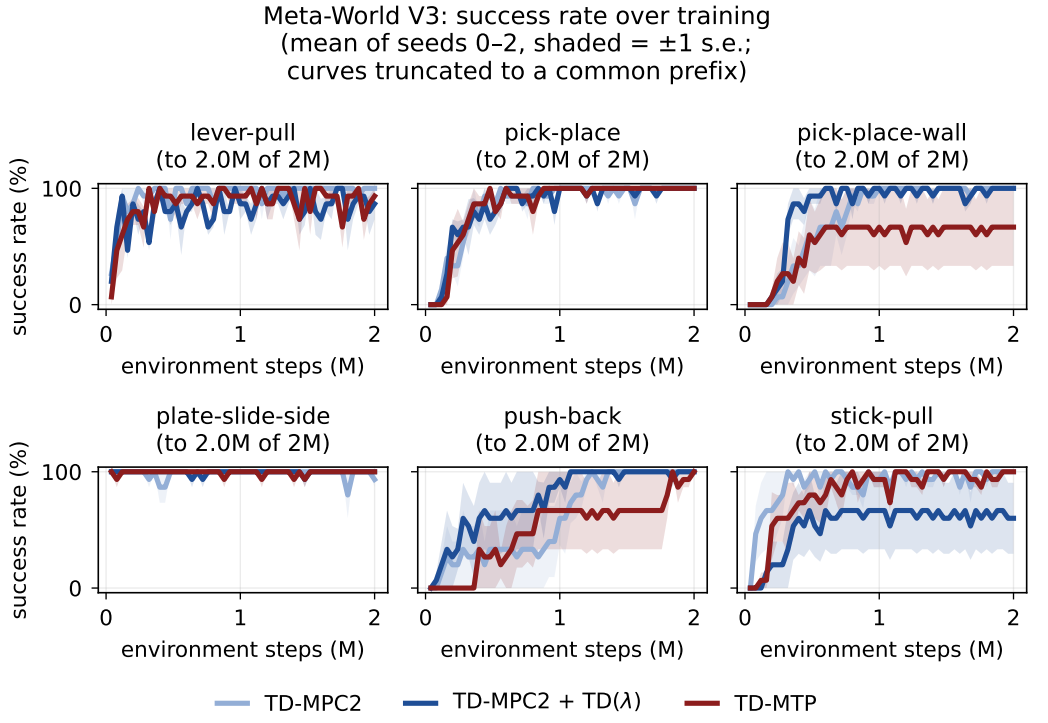

Figure 4: **Manipulation learning curves include both gains and failures.** All six tasks in Meta-World protocol A (S3), $\lambda = 0.8$ for the multi-step methods. Lines show mean success over seeds at each 40k-step checkpoint; bands show ± 1 standard error. The last three means match Table 8; the last plotted point uses one evaluation. The target-only method learns `push-back` earlier, but performs worse on `stick-pull`.

Table 6: **Training and evaluation protocols.** Steps are nominal environment-step budgets; "Eval." is the interval between checkpoints. Each checkpoint uses five evaluation episodes. A run's endpoint score averages its last three evaluations. Run counts are per task and method. Learning-curve summaries are defined in Appendix D.1.

ID	Task set	Steps	Eval.	λ	Runs
Planner-by-target study: both planners under both targets					
P1	DMControl, 3 tasks	300k	20k	0 / 0.6 / 0.8	5
P2	HumanoidBench h1, 7 tasks	2M	20k	0 / 0.8	5[a]
Benchmark suite: TD-MTP versus TD-MPC2					
S1	HumanoidBench h1, 14 tasks	2M	20k / 40k	0.5 / 0.8	3–7[b]
S2	DMControl, 8 tasks	1M	20k	0.8	5–10[c]
S3	Meta-World A, 6 tasks	2M	40k	0.8	3
S4	Meta-World B, 6 tasks	2M	40k	0.8	3
S5	MyoSuite, 4 -hard tasks	2M	40k	0.8	3[e]
Target and training-budget sensitivity					
E5	Acrobot, multi-step arms	1M	20k	0.6	5
P3	HumanoidBench h1-pole	2M	20k / 40k	0.8	7
E1	HumanoidBench h1-reach	2M	40k	0 / 0.5 / 0.8	3
E2	DMControl dog-run	1M / 4M	40k	0 / 0.8	4 unpaired
E3	HumanoidBench h1-stair	4M	40k	0 / 0.5 / 0.8	3

P1 and P2 use the tasks in Table 1; P1 uses $\lambda = 0.6$ only on acrobot-swingup. S1 uses 0.5 only on stair. Suite λ values describe the multi-step methods; baseline targets are one-step. Meta-World A has a 100-decision-step cap; B has a 500-step cap. MyoSuite uses the -hard variants.

[a] Complete seed-level data are available for stair and pole only. Learning curves use the same five (Figure 3). [b] The balance_simple baseline reads all three of its runs. stair has seven baseline runs and four TD-MTP runs. Two S1 records have incomplete evaluation coverage (Appendix F). [c] The PyTorch baseline has five seeds and 51 checkpoints. TD-MTP has five runs, or 9–10 from two pooled sets with overlapping seed identifiers. [e] Every MyoSuite cell reads three runs, matching Table 9.

Paired JAX arms use the same MPPI evaluation settings and share initialization within each target. The interaction is seed-matched across targets rather than paired within one run. S2 uses the reference PyTorch evaluation configuration. E1 pairs $\lambda = 0$ with 0.5; the 0.8 run set is separate. E3 uses paired seeds 0–2. E5 uses acrobot seeds 0–4, separate from P1's 1–5. P3's seven-seed pool differs from P2. E2 fixes the structured planner and changes the target.

mean training success from 0.616 to 0.772; adding the planner lowers it to 0.504. On `disassemble`, TD-MTP raises it from 0.207 to 0.595. On `stick-pull`, TD-MTP ties the baseline at the endpoint but has lower mean training success (0.817 versus 0.900). These comparisons distinguish earlier learning from final success. They do not isolate whether a gain comes from exploring new states or learning better values from visited states.

D.2 Endpoint and seed-level comparisons

Figure 5 shows every suite outcome, including losses. Figure 6 separates the two planner effects within each paired seed. Figures 7 and 8 show individual success rates, so means do not hide failed runs. These figures summarize final performance; they do not establish learning speed.

D.3 HumanoidBench `h1` and DMControl

The HumanoidBench suite uses $\lambda = 0.5$ on `stair` and 0.8 elsewhere; DMControl uses 0.8 throughout. These settings differ from the paired study in Table 1, which uses 0.6 on `acrobot` and 0.8 on `stair`. Results for those tasks come from separate runs. The DMControl baseline uses the reference PyTorch implementation with five seeds. We report the SD across runs and the run count for each DMControl entry. TD-MTP entries with 9–10 runs pool two sets with the same λ and code version.

Table 7: **Mean success during training on Meta-World.** Equation 24, from 40k to 2M steps; higher is better. Seed counts match Table 8. Bold marks the largest mean, without a significance claim. These descriptive means complement the endpoint scores and include every task in each protocol.

Task	TD-MPC2	+Target	TD-MTP
Protocol A (S3): 100-decision-step cap			
lever-pull	**0.949**	0.831	0.873
pick-place	0.860	0.857	**0.861**
pick-place-wall	0.748	**0.832**	0.537
plate-slide-side	0.988	**0.996**	0.993
push-back	0.616	**0.772**	0.504
stick-pull	**0.900**	0.547	0.817
Protocol B (S4): 500-step cap			
assembly	**0.621**	0.540	0.416
basketball	**0.740**	0.651	0.612
button-press	0.940	**0.956**	0.951
coffee-push	**0.743**	0.565	0.627
dial-turn	0.875	**0.913**	**0.913**
disassemble	0.207	0.401	**0.595**

Table 8: **Success rates at 2M steps.** Cells give the mean and seed count (n). Bold marks the largest mean when methods differ; tied maxima are both bold. Δ is TD-MTP minus TD-MPC2. Outcomes use the ± 0.05 tie tolerance, not a significance test.

Task	TD-MPC2	+Target	TD-MTP	Δ	Outcome
Meta-World A (S3)					
lever-pull	**1.000** (3)	0.867 (3)	0.844 (3)	-0.156	loss
pick-place	1.000 (3)	1.000 (3)	1.000 (3)	0.000	tie
pick-place-wall	**1.000** (3)	**1.000** (3)	0.667 (3)	-0.333	loss
plate-slide-side	0.978 (3)	**1.000** (3)	**1.000** (3)	$+0.022$	tie
push-back	**1.000** (3)	**1.000** (3)	0.956 (3)	-0.044	tie
stick-pull	**1.000** (3)	0.622 (3)	**1.000** (3)	0.000	tie
Meta-World B (S4)					
assembly	**1.000** (3)	0.822 (3)	0.667 (3)	-0.333	loss
basketball	**0.978** (3)	0.933 (3)	0.844 (3)	-0.133	loss
button-press	**1.000** (3)	0.933 (3)	**1.000** (3)	0.000	tie
coffee-push	**0.844** (3)	0.622 (3)	0.800 (3)	-0.044	tie
dial-turn	0.756 (3)	0.867 (3)	**1.000** (3)	$+0.244$	win
disassemble	0.267 (3)	0.778 (3)	**0.889** (3)	$+0.622$	win
MyoSuite -hard **variants (S5)**					
KeyTurn	0.000 (3)	0.800 (3)	**0.844** (3)	$+0.844$	win
ObjHold	1.000 (3)	1.000 (3)	1.000 (3)	0.000	tie
PenTwirl	0.933 (3)	0.689 (3)	0.933 (3)	0.000	tie
Pose	0.000 (3)	0.000 (3)	0.000 (3)	0.000	tie

+Target denotes TD-MPC2 with the λ-return. Individual scores are in Table 9.

Table 9: **Individual success rates behind Table 8.** Each number is one seed's mean over the last three evaluations at 2M steps. Groups use different episode protocols and remain separate.

Task	TD-MPC2	+Target	TD-MTP
Meta-World A (S3)			
lever-pull	1.00, 1.00, 1.00	0.93, 1.00, 0.67	0.93, 0.67, 0.93
pick-place	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00
pick-place-wall	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 0.00, 1.00
plate-slide-side	1.00, 1.00, 0.93	1.00, 1.00, 1.00	1.00, 1.00, 1.00
push-back	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 0.87
stick-pull	1.00, 1.00, 1.00	0.00, 0.87, 1.00	1.00, 1.00, 1.00
Meta-World B (S4)			
assembly	1.00, 1.00, 1.00	1.00, 0.47, 1.00	1.00, 0.00, 1.00
basketball	0.93, 1.00, 1.00	0.87, 0.93, 1.00	1.00, 0.60, 0.93
button-press	1.00, 1.00, 1.00	0.87, 0.93, 1.00	1.00, 1.00, 1.00
coffee-push	0.93, 0.67, 0.93	0.67, 0.67, 0.53	0.93, 0.80, 0.67
dial-turn	0.60, 0.93, 0.73	0.87, 0.73, 1.00	1.00, 1.00, 1.00
disassemble	0.00, 0.00, 0.80	0.93, 0.40, 1.00	0.93, 0.93, 0.80
MyoSuite -hard **variants (S5)**			
KeyTurn	0.00, 0.00, 0.00	0.60, 0.80, 1.00	0.87, 0.80, 0.87
ObjHold	1.00, 1.00, 1.00	1.00, 1.00, 1.00	1.00, 1.00, 1.00
PenTwirl	1.00, 0.87, 0.93	0.47, 0.60, 1.00	1.00, 0.87, 0.93
Pose	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.00, 0.00, 0.00

Their seed identifiers overlap (Table 18). This is a comparison across implementations, whereas P1 and P2 share one implementation. For example, our port's one-step `dog-run` baseline scores 244 ± 103, against 574 ± 43 for reference PyTorch at 1M. Both use five seeds. This gap limits attribution in the suite comparison.

The `reach` comparison in §4 uses separate runs: returns are 9053, 7422 and 5240 at $\lambda = 0, 0.5, 0.8$, with three seeds per setting. The reported 57% recovery is calculated within that comparison, whose scores differ from the 9172.1 and 5259.8 in Table 10.

The multi-step target scored on its own. Table 11 compares every available target-only arm with TD-MPC2 under the same tolerances as above, keeping the paired study and manipulation suites separate. The target alone wins on 4 of the ten paired tasks and 3 of the sixteen manipulation tasks. Counting any increase in mean gives 8 of ten and 4 of sixteen, respectively. S1 and S2 have no target-only arm and are omitted.

D.4 Paired results and component controls

The main text reports the four methods and their interaction on ten tasks. Here we give seed-level values for the two largest interactions, then controls for replay context, fitting and proposals.

Per-seed cells on `stair` and `pole`. The two largest interactions in Table 1 use the five seeds listed in Table 12, protocol P2, which gives all four scores per seed at 2M under the last-three rule. Within each job, the two planner arms share a model initialization. Across jobs, the shared training script derives that initialization from the seed alone, so a seed's one-step and λ runs also begin from the same parameters. The seed means reproduce the main-text interactions.

A separate `stair` experiment uses seeds 0–4 at 2M and evaluates each arm with its own planner. It gives a mean interaction of $+355.4$, a 95% interval of $[+181, +530]$, and five positive seed effects. This is a separate replication with a different evaluation protocol.

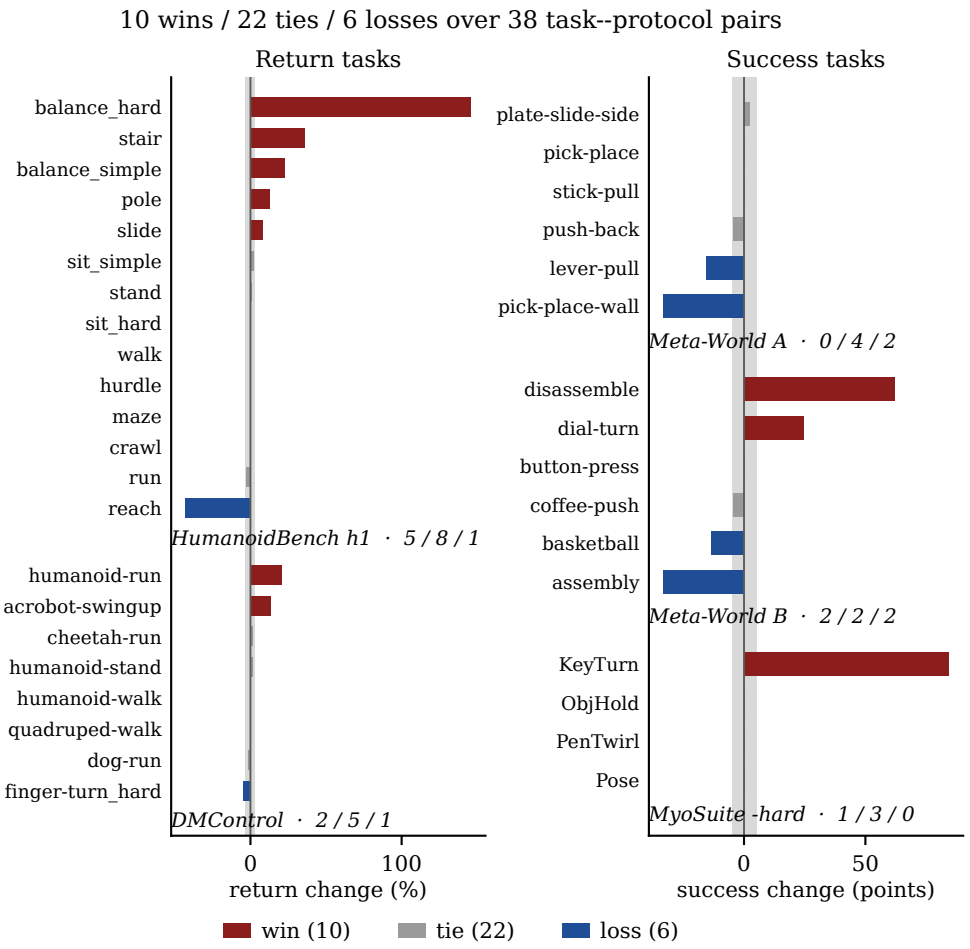

Figure 5: **All 38 suite outcomes.** Values come from Tables 10 and 8, protocols S1–S5. Left: relative return change, $100(R_{\text{TD-MTP}}/R_{\text{TD-MPC2}} - 1)$. Right: success-rate change in percentage points. Grey bands mark the descriptive tie tolerances: $\pm 3\%$ return and ± 5 success points. Labels report wins/ties/losses. Scales differ between panels and must not be pooled. These are method means; no uncertainty interval is implied.

Controls for replay windows and planner components. Our main one-step-versus-λ comparisons change two things at once: the λ target draws a longer replay window, so target construction and replay sampling move together. An experiment with seven settings separates these changes on `stair` at 2M and `humanoid-run` at 300k. Each condition uses five paired seeds. Multi-step targets use $\lambda = 0.8$; extended replay windows add $\ell = 10$ transitions. The seeds are 1865, 3, 4, 5927, 6108 on `stair` and 1–5 on `humanoid-run`. **R** is stock MPPI with the ordinary window H and the one-step target. **A0** widens the window to $L = H + \ell$ and still computes the one-step target at the first H positions, without using the tail in the loss; it is the matched-window control. **AI** is stock MPPI with window L and the λ target. **B** is our planner with the tensor candidates switched off ($\beta = 0$; free candidate slots use local Gaussian draws), **C** the full planner, each at window L under the one-step target (`0`) and under λ (`1`). Table 14 gives the conditions and cell means; Table 15 gives the contrasts. These controls use 95% normal-approximation intervals over five seeds. The main comparison uses Student-t intervals.

The controls distinguish replay-window, proposal, and fitting changes. Each interval applies to its stated budget and run set. The isolated tensor-proposal contrast is $\mathcal{I}_s$ in Table 15.

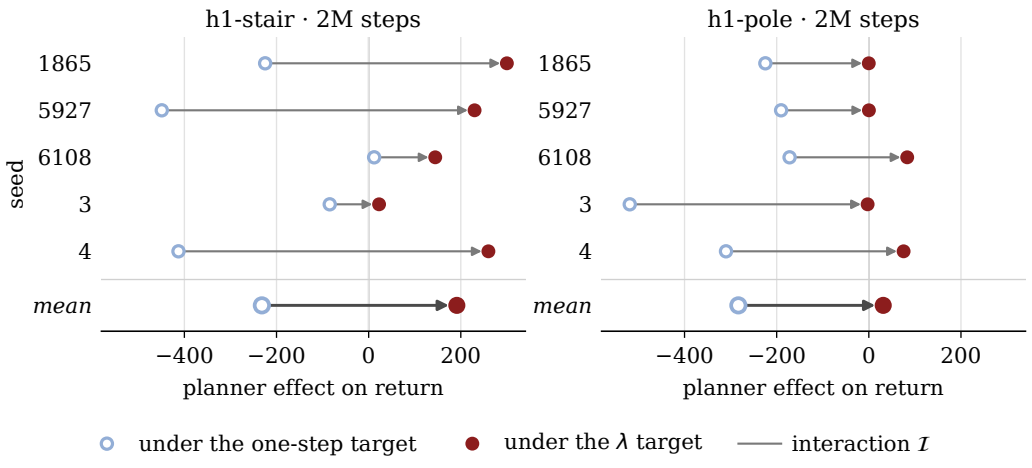

Figure 6: **Planner effects within each paired seed.** Return differences use Table 12, protocol P2 at 2M steps. Open circles show $R_{1,0} - R_{0,0}$; filled circles show $R_{1,1} - R_{0,1}$. A line moving right indicates a positive interaction. The planner effect under the multi-step target remains negative for some seeds, even when the interaction is positive. The paired interpretation reads each seed across both targets; a run's initialization is derived from its seed alone, so the two targets start from the same parameters.

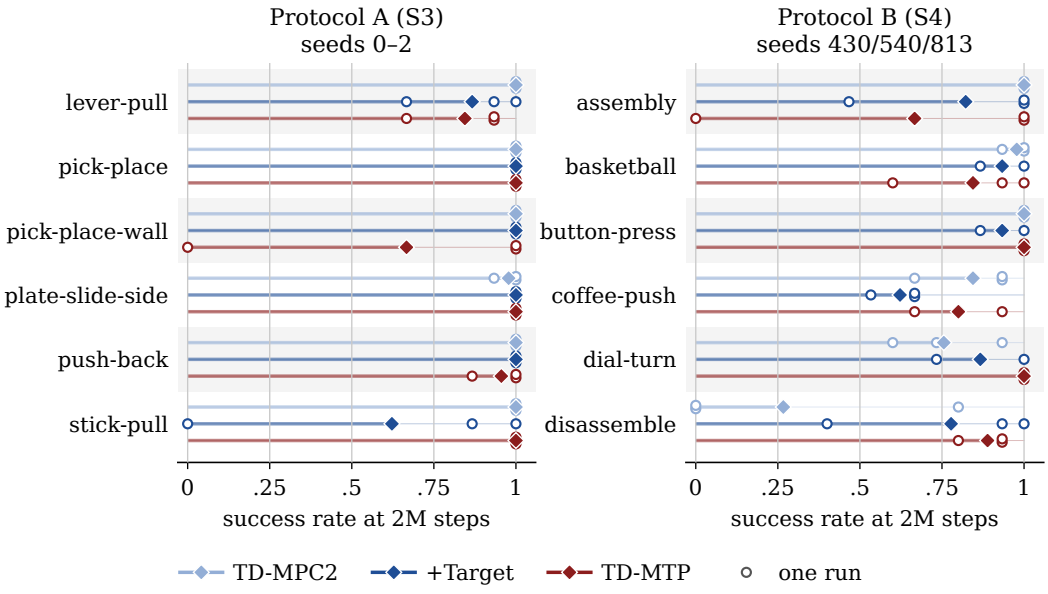

Figure 7: **Meta-World success across individual runs.** Each small circle is a seed's last-three-evaluation mean at 2M. Diamonds mark the method means in Table 8; individual scores are in Table 9. Protocols A and B have different episode caps and remain separate (S3 and S4). Each method has its own lane inside a task's band, and seeds sharing a value are fanned vertically; neither vertical position carries uncertainty. The failed `pick-place-wall` and `assembly` runs explain why their method means fall below the baseline.

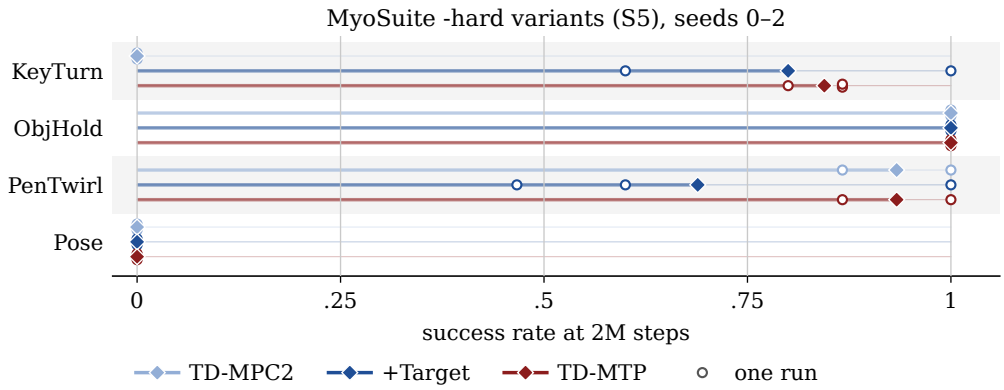

Figure 8: **MyoSuite success across individual runs.** All four tasks use the `-hard` variant and protocol S5 at 2M. Marks follow Figure 7; values come from Table 9. Each `PenTwirl` lane shows three runs, seeds 0–2. Their means give a tie between TD-MTP and TD-MPC2; the TD-MTP runs include two recovered after the filename collision. `ObjHold` is solved by every method, while `Pose` remains unsolved.

Replay-window control. On `stair` at 2M, A0−R gives $+138.5$ ($[-7, +284]$) on the five seeds in Table 15. Five further seeds reverse that ordering. Across all ten seeds, the reported effect is $+66.4$ ($[-37, +169]$). The original five-seed estimate therefore overstates the gain across all ten seeds.

Fitting and execution controls. Conditions D and E use the same five `stair` seeds at 2M, $\beta = 0$ and window L. D fits a softmax over all candidates instead of truncating to the top K. E draws the executed action from the refit first-step distribution. E changes execution stochasticity, but does not reproduce TD-MPC2's execution rule exactly. For each seed, define

$$\mathcal{I}_X = (R_{X,\lambda} - R_{X,\text{1-step}}) - (R_{\text{Al}} - R_{\text{A0}}), \qquad \text{share}_X = 1 - \mathcal{I}_X/\mathcal{I}_B.$$

This isolates each control's planner–target interaction against the matched-window reference. The $\beta = 0$ configuration gives $\mathcal{I}_B = 496.8$ with interval $[289, 705]$, positive on all five seeds. The additional means are $D0 = 486.7 \pm 53$, $Dl = 565.9 \pm 102$, $E0 = 325.0 \pm 11$ and $El = 634.3 \pm 107$. They give $\mathcal{I}_D = 264.8$ ($[165, 364]$) and $\mathcal{I}_E = 494.9$ ($[329, 661]$). The mean within-seed fitting share is 46.1% ($[39, 52]\%$), positive on all five seeds. The execution share is -6.8% ($[-42, +28]\%$), which is inconclusive. Mean within-seed shares need not equal ratios of mean interactions. This experiment uses a different reference from Table 1; its 496.8 must not be substituted for the main 423.2 interaction.

Table 10: **Final return by task.** Last-three-evaluation means under protocols S1 and S2. Δ is the relative change from TD-MPC2 to TD-MTP. Bold marks the higher mean only when the difference exceeds the descriptive 3% tie tolerance. S1 uses a nominal 2M budget; incomplete evaluation records are described in Appendix F.

HumanoidBench h1: 2M steps (S1)

Task	TD-MPC2	TD-MTP	Δ	Outcome
balance_hard	175.0	**430.0**	+145.7%	win
stair	489.6	**664.1**	+35.6%	win
balance_simple	479.1	**677.7**	+41.5%	win
pole	804.9	**908.1**	+12.8%	win
slide	560.5	**605.4**	+8.0%	win
sit_simple	852.9	866.2	+1.6%	tie
stand	846.6	853.0	+0.8%	tie
sit_hard	820.5	823.3	+0.3%	tie
walk	816.8	818.3	+0.2%	tie
hurdle	593.1	593.4	+0.1%	tie
maze	365.8	365.7	0.0%	tie
crawl	975.6	973.5	−0.2%	tie
run	782.5	762.2	−2.6%	tie
reach	**9172.1**	5259.8	−42.7%	loss

DMControl: 1M steps (S2)

	TD-MPC2 (PyTorch)		TD-MTP (JAX)			
Task	Mean ± SD	n	Mean ± SD	n	Δ	Outcome
acrobot-swingup	540 ± 230	5	**612** ± 49	10	+13.3%	win
humanoid-run	394 ± 49	5	**475** ± 35	9	+20.6%	win
cheetah-run	898 ± 30	5	915 ± 4	5	+1.9%	tie
humanoid-stand	916 ± 11	5	933 ± 10	5	+1.9%	tie
humanoid-walk	905 ± 15	5	910 ± 15	10	+0.6%	tie
quadruped-walk	972 ± 6	5	974 ± 5	5	+0.2%	tie
dog-run	574 ± 43	5	564 ± 49	9	−1.7%	tie
finger-turn_hard	**969** ± 27	5	921 ± 59	10	−5.0%	loss

HumanoidBench provides method means; cross-seed SDs are unavailable here. For DMControl, n counts included runs; entries with 9–10 runs pool two sets at the same λ. The S2 comparison uses different implementations, which limits attribution. Table 18 gives the task-level seed sets for both suites.

Table 11: **The multi-step target alone against TD-MPC2.** Cells with a trained target-only arm, read from Tables 1 and 8. Return rows use the 3% tie tolerance, success rows ± 0.05. Outcomes are descriptive; they are not equivalence tests. The 14-task HumanoidBench suite (S1) and the DMControl suite (S2) have no target-only arm and are not listed.

Task set	Tasks	Wins	Ties	Losses
Paired study, episode return				
DMControl, 300k (P1)	3	3	0	0
HumanoidBench h1, 2M (P2)	7	1	4	2
both, all	*10*	*4*	*4*	*2*
Manipulation suites, success rate				
Meta-World A (S3)	6	0	4	2
Meta-World B (S4)	6	2	1	3
MyoSuite (S5)	4	1	2	1
all three, all	*16*	*3*	*7*	*6*

Meta-World A and B use different episode caps and are not pooled.

Table 12: **Per-seed results for the main comparison.** Final return at 2M steps, using the last-three rule. $\mathcal{I}$ is the interaction in Equation 12. Bold marks the largest method mean and the mean interaction for each task. Intervals use Student-t across the five registered seeds; the interaction is matched by seed across targets, not paired within one run.

	One-step target		λ-return target		
Seed	TD-MPC2	+Planner	+Target	TD-MTP	$\mathcal{I}$
HumanoidBench h1-stair: 2M steps					
1865	484.4	259.7	419.6	719.6	+524.7
5927	756.7	307.7	427.3	657.2	+678.9
6108	302.9	314.8	483.0	627.5	+132.7
3	408.1	323.6	488.6	511.2	+107.0
4	674.8	262.0	479.4	739.3	+672.8
Mean	525.4	293.6	459.6	**651.0**	**+423.2**
Interaction: 5/5 seeds positive; 95% interval					[+71, +776]
HumanoidBench h1-pole: 2M steps					
1865	908.0	683.4	908.1	908.0	+224.5
5927	834.4	644.0	908.1	908.3	+190.5
6108	672.4	500.4	824.8	908.1	+255.2
3	908.7	389.8	909.8	907.2	+516.3
4	906.3	596.3	829.0	904.5	+385.6
Mean	846.0	562.8	876.0	**907.2**	**+314.4**
Interaction: 5/5 seeds positive; 95% interval					[+147, +482]

Table 13: **Planner–target interaction on every paired task.** $\mathcal{I}$ is Equation 12 computed per seed and averaged, using the last-three rule of Table 1. Intervals are unadjusted paired Student-t over the five registered seeds; * marks an interval excluding zero. The final column counts seeds with $\mathcal{I} > 0$. `h1` tasks run to 2M, DMControl to 300k. Values are computed from unrounded seed scores, so they can differ by up to 0.2 from the difference of the rounded cells printed in Table 1.

Task	Budget	$\mathcal{I}$	95% interval	Seeds > 0
stair	2M	$+423.2$	$[+71, +776]^*$	$5/5$
pole	2M	$+314.4$	$[+147, +482]^*$	$5/5$
hurdle	2M	$+69.6$	$[-58, +197]$	$3/5$
humanoid-walk	300k	$+61.7$	$[-221, +345]$	$4/5$
humanoid-run	300k	$+49.4$	$[-71, +169]$	$3/5$
acrobot-swingup	300k	$+25.9$	$[-150, +202]$	$3/5$
stand	2M	$+8.1$	$[-15, +31]$	$4/5$
maze	2M	$+7.0$	$[-16, +30]$	$3/5$
run	2M	-11.4	$[-65, +42]$	$2/5$
walk	2M	-53.8	$[-140, +32]$	$2/5$

Eight of ten interactions are positive; two intervals exclude zero. Reward scales differ across tasks, so the magnitudes are not pooled.

Table 14: **Seven conditions for replay and proposal controls.** Mean $\pm$ sample SD over five seeds. $H = 3$, $L = 13$ and $\lambda = 0.8$. Bold marks the largest mean within each task, without a significance claim. Table 15 gives the component effects.

ID	Planner	Window	Target	Stair (2M)	Humanoid-run (300k)
R	MPPI	H	1-step	566.6 ± 168	79.2 ± 38
A0	MPPI	L	1-step	$\mathbf{705.1} \pm 69$	46.0 ± 43
Al	MPPI	L	λ	519.4 ± 60	$\mathbf{197.3} \pm 52$
B0	Structured, no tensor	L	1-step	278.7 ± 80	52.6 ± 57
Bl	Structured, no tensor	L	λ	589.9 ± 118	163.8 ± 9
C0	Structured, full	L	1-step	252.3 ± 41	68.4 ± 55
Cl	Structured, full	L	λ	548.9 ± 165	187.4 ± 49

Table 15: **Component effects at a fixed training budget.** Contrasts use the conditions in Table 14 and five paired seeds. Bold marks the tensor–target interaction $\mathcal{I}_s$. The reported 95% intervals use a normal approximation, which is less conservative than Student-t at five seeds.

Contrast	Return change	95% interval	Positive seeds
HumanoidBench h1-stair: 2M steps			
Replay window: A0 − R	$+138.5$	$[-7, +284]$	$4/5$
Target, matched window: Al − A0	-185.7	$[-280, -91]$	$0/5$
Tensor proposals, λ: Cl − Bl	-41.0	$[-105, +23]$	$1/5$
Tensor proposals, 1-step: C0 − B0	-26.4	$[-87, +35]$	$2/5$
Full planner, λ: Cl − Al	$+29.5$	$[-114, +173]$	$3/5$
Tensor–target interaction $\mathcal{I}_s$	$\mathbf{-14.6}$	$[-107, +78]$	$2/5$
DMControl humanoid-run: 300k steps			
Replay window: A0 − R	-33.2	$[-104, +38]$	$2/5$
Target, matched window: Al − A0	$+151.3$	$[+78, +225]$	$5/5$
Tensor proposals, λ: Cl − Bl	$+23.7$	$[-24, +71]$	$4/5$
Tensor proposals, 1-step: C0 − B0	$+15.8$	$[-67, +99]$	$3/5$
Full planner, λ: Cl − Al	-9.9	$[-87, +67]$	$2/5$
Tensor–target interaction $\mathcal{I}_s$	$\mathbf{+7.9}$	$[-72, +88]$	$3/5$

$\mathcal{I}_s = (\mathrm{Cl} - \mathrm{Bl}) - (\mathrm{C0} - \mathrm{B0})$. Its intervals include zero on both tasks. Positive seeds count return changes above zero; bold does not indicate significance.

E SENSITIVITY TO THE TARGET AND TRAINING BUDGET

These comparisons test the choice of λ, the training budget and threshold reliability. Endpoint scores use the primary last-three-evaluation rule. The separate first-crossing comparison uses raw checkpoint returns.

`acrobot-swingup` at a longer budget. A separate 1M comparison uses the two multi-step methods, five seeds, and $\lambda = 0.6$ (E5). The target-only method scores 625.1 ± 35 and TD-MTP scores 652.7 ± 46 (mean $\pm$ cross-seed SD). The planner's mean gain is about 28, compared with 105.1 at 300k. It improves three of five seeds at 1M, compared with all five at 300k. The 1M comparison uses seeds 0–4; P1 uses 1–5. This difference limits attribution to the budget alone. Only the two multi-step methods are compared, so these results do not measure the planner–target interaction at 1M.

Target effects on `dog-run` at two budgets. Protocol E2 fixes the structured planner and compares one-step and $\lambda = 0.8$ targets. The multi-step method has higher mean return at both budgets (Table 16). The gap narrows as three of the four one-step runs improve; one remains near zero at 4M. Thus the mean gap combines earlier learning with a difference in failed runs. The arms use four unpaired seeds each. This is a target comparison within our JAX implementation, not a planner ablation or an improvement over the reference PyTorch result. The cross-implementation gap on this task is reported in Appendix D.

Table 16: **Target comparison at 1M and 4M on `dog-run`.** Protocol E2, structured planner fixed; four unpaired seeds per target. Each entry averages a run's last three evaluations at the stated budget. SD is the sample SD across runs. Bold marks the larger method mean at each budget.

Steps	Target	Seed 0	Seed 1	Seed 2	Seed 3	Mean $\pm$ SD
1M	One-step	177	291	366	37	218 ± 143
	$\lambda = 0.8$	600	499	587	562	$\mathbf{562} \pm 45$
4M	One-step	724	748	734	28	558 ± 354
	$\lambda = 0.8$	788	691	756	668	$\mathbf{726} \pm 56$

Threshold reliability on `pole`. Protocol P3 compares the two planners at $\lambda = 0.8$ on seven seeds, paired within each job. It combines seeds 1865, 5927, and 6108 on a 20k evaluation grid with four runs on a 40k grid. This pool differs from P2; run 3 differs from P2's run 3. Table 17 gives the first raw evaluation with return at least 890. TD-MTP reaches this threshold on all seven seeds; the target-only method does so on five. Treating non-crossings as infinite, both median first-crossing times are 0.76M. The observed difference is the fraction of runs that reach the threshold, not a faster median crossing. A first crossing does not imply sustained performance. Under P2's endpoint criterion on its five seeds, TD-MTP exceeds 890 on five of five and the target-only method on three of five.

Target sensitivity on `reach`. With the reference planner fixed, returns are 9053, 7422 and 5240 at $\lambda = 0, 0.5, 0.8$. Each setting uses three seeds (protocol E1). The middle setting recovers $2182/3813 = 57.2\%$ of the endpoint gap. The structured-planner arm gives the same ordering: 8164, 7689 and 5260. Thus, the decline is not specific to either planner. However, the 0 and 0.5 settings use one paired experiment, while 0.8 comes from a separate experiment. The one-step experiment baselines differ by 1.3% (9172 versus 9053). These data show target sensitivity, not a validated rule based on action dimension or the theoretical bound.

Longer training on `stair`. Protocol E3 compares the four methods at 4M on three seeds per condition. At $\lambda = 0.8$, the means are 754.7, 495.3, 777.1 and 705.0 in the order of Table 1. The interaction is $+187$, positive on all three seeds. Yet the planner effect under the multi-step target is -72, with only one seed improving. At $\lambda = 0.5$, the interaction is $+240$, again positive on all three seeds. The corresponding planner effect is -20, with no seed improving. The 0.8 result uses the same λ as the main 2M study, so a target-setting change alone cannot explain the reversal. The budgets use

Table 17: **First threshold crossing on `h1-pole`.** Protocol P3, $\lambda = 0.8$, common MPPI evaluation planner. Entries give the first evaluation with return ≥ 890, in millions of environment steps. A dash means no crossing by 2M. No uncertainty interval is implied.

Seed	+Target	TD-MTP
1865	0.88	1.32
5927	0.60	0.52
6108	—	0.46
0	—	1.00
1	0.76	0.76
2	0.56	0.48
3	0.48	1.04
Reached by 2M	5/7	7/7
Median, non-crossings $= \infty$	0.76	0.76

different run sets; three seeds leave the budget effect uncertain. A positive interaction is compatible with a lower final return. The interaction measures how the planner effect changes across targets.

F EXPERIMENTAL DETAILS

Scoring and pairing. A run's score averages its last three evaluations; method scores average the included runs. Evaluation uses five episodes at the cadence in Table 6. Paired JAX comparisons use a common MPPI evaluation planner; S2 uses the reference PyTorch baseline. SDs describe variability across the included runs. The suite tie bands are descriptive tolerances, not equivalence tests. The two Meta-World episode protocols remain separate.

Planner arms within each target share their model initialization. Across targets the match is structural: the training script keys a run's initialization on its seed alone, and the target enters only the value-target computation, so a seed's one-step and λ runs start from identical parameters. Protocol P2 reports the five seeds registered for it before dispatch, the same five in every cell. All five registered seeds have run files on all seven `h1` tasks, and every cell of the `h1` block of Table 1 reconstructs from them. Per-seed values are printed for `stair` and `pole` only (Table 12). Evaluation episodes reset with seeds $0, 1000, 2000, 3000, 4000$: the same five start states for every run and every method, so evaluation-state variance does not enter a between-method comparison. A run trained with seed 0 shares the first of these with its own first training episode. The five-seed control intervals and fitting shares also lack complete seed-level values.

Seed coverage in the return-task suites. Table 18 lists the training seeds for S1 and S2. S1 uses three runs per method on most tasks, with unequal counts on `stair`, `crawl` and `slide`. The `crawl` and `slide` suite comparisons are unpaired. S2 uses separate PyTorch baseline runs and JAX TD-MTP runs. Some TD-MTP cells pool two run sets with overlapping seed identifiers. A pooled run count does not establish an equal number of independent training seeds. The pooled means and SDs describe the included runs; independence requires confirmation of their initialization and random streams. Seed sets for the success tasks are given in Table 9.

Table 18: **Training-seed coverage for the return-task suites.** Parentheses give the number of included runs per task and method. Within each S2 task, A and B label separate run sets; repeated seed identifiers are retained explicitly. This table specifies coverage and does not establish pairing or independence.

Task	TD-MPC2 seeds (n)	TD-MTP seeds (n)
HumanoidBench h1 (S1)		
stair	1865, 5927, 6108; 0–3 (7)	0–3 (4)
pole, stand, hurdle, run, walk, maze	1865, 5927, 6108 (3)	1865, 5927, 6108 (3)
crawl, slide	0–2 (3)	0–3 (4)
balance_hard	0–2 (3)	0–2 (3)
reach, sit_simple, sit_hard	430, 540, 813 (3)	430, 540, 813 (3)
balance_simple	430, 540, 813 (3)	430, 540, 813 (3)
DMControl (S2): PyTorch baseline; JAX TD-MTP		
acrobot-swingup, humanoid-walk, finger-turn_hard	1–5 (5)	A: 1–5; B: 1–5 (10)
humanoid-run, dog-run	1–5 (5)	A: 1–5; B: 0–3 (9)
cheetah-run	1–5 (5)	1, 3, 4, 5, 10 (5)
humanoid-stand	1–5 (5)	1, 2, 6, 7, 10 (5)
quadruped-walk	1–5 (5)	1–5 (5)

Software and hardware. The reported JAX configuration uses Python 3.12.13, JAX/JAXlib 0.10.2, Flax 0.12.7 and Optax 0.2.8. The base MuJoCo/MJX version is 3.8.1, with DMControl 1.0.41 and HumanoidBench 0.2. Meta-World uses version 3.1.1 with MuJoCo 3.3.0; MyoSuite uses MuJoCo 3.6.0. The four MyoSuite tasks are the `-hard` (randomised-goal) variants: `myoHandKeyTurnRandom-v0`, `myoHandObjHoldRandom-v0`, `myoHandPenTwirlRandom-v0` and `myoHandPoseRandom-v0`. The `Fixed-v0` counterparts are not used. The MyoSuite package version is unspecified, which limits exact reproduction. The reference PyTorch baseline uses Python 3.9.25, PyTorch 2.6.0, Mu-

JoCo 3.1.2 and DMControl 1.0.16. Hardware consists of NVIDIA H100 80GB GPUs and 3g.40gb MIG slices, with two to four jobs sharing a device. Repeated Meta-World runs with the same seed and action sequence can produce different rewards. Seed identifiers alone therefore do not guarantee identical trajectories. Exact reproduction also requires the training code, complete per-run results and scoring commands.

Planning cost. With the checkpoint, device, and candidate budget fixed, reported planning times are 4.2 ms for MPPI and 5.1 ms for the structured planner, both with and without tensor candidates. These are 10th percentiles over 300 calls, measured in three to five sessions per condition. They describe fast calls, not median or worst-case latency. The reported ratio is about 1.2 on an idle device and 1.8 on a shared device. Disabling tensor candidates retains the overhead from the other planner operations. These measurements do not establish end-to-end training cost or wall-clock efficiency.