Beyond Policy Alignment: Closing the Planning–Learning Loop for Robot Control with Learned World Models

Anonymous Authors

***Abstract*— Planning with learned world models combines online trajectory optimization with learned value and policy functions for high-dimensional control. Because the planner determines the experience used for learning, while the learned critic and actor in turn score and propose future plans, planning and learning form a closed feedback loop. TD-MPC is a prominent instance of this design. Recent policy-constrained variants strengthen one part of the loop by aligning the learned policy with planner behavior. We introduce PL-MPC (*Planning–Learning MPC*), which additionally modifies critic supervision and planner terminal-value estimation. Hybrid multi-step TD targets expose critic updates to more realized rewards before bootstrapping; disagreement-aware terminal estimates reduce the influence of uncertain critic values during MPPI planning; and return-weighted actor distillation emphasizes planner-executed actions from high-return episodes. The world-model architecture and MPPI optimizer are otherwise unchanged. On HumanoidBench, the largest gains occur on `balance-hard`, where Total Average Return (TAR) increases from 98 ± 18 to 387 ± 255, and `hurdle`, from 199 ± 13 to 466 ± 200; performance across the broader benchmark remains task dependent, and PL-MPC remains competitive on DMControl. Controlled ablations show different component interactions across the two tasks. We further demonstrate zero-shot sim-to-real transfer on wrench–nut alignment with a 7-DoF KUKA IIWA14, obtaining higher observed success than TD-M(PC)^2 on the training object size and two unseen sizes. Code and data will be available at: https://icra27-pl-mpc.github.io.**

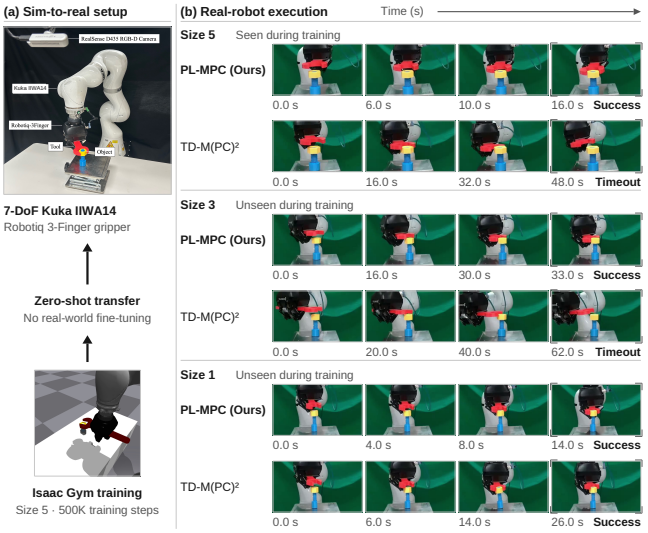


Fig. 1. **Zero-shot real-robot wrench–nut alignment.** (a) PL-MPC is trained entirely in Isaac Gym and deployed on a KUKA IIWA14 without real-world fine-tuning; the deployed checkpoint is selected within the first 500K training steps. (b) Representative trials on the training size (size 5) and unseen sizes 3 and 1. Frames are sampled at different times during each execution and are not consecutive control steps. In the shown size 5 and size 3 trials, PL-MPC completes the task while TD-M(PC)^2 reaches the 128-step limit. On size 1, both methods succeed, with PL-MPC completing the trial in fewer control steps. Successful sequences end with the wrench seated around the nut.

I. Introduction

Planning with learned world models has become a powerful approach to high-dimensional control. A learned dynamics model predicts the consequences of candidate actions, model-predictive control (MPC) optimizes short action sequences online, and a learned value function estimates return beyond the planning horizon. Many recent methods also learn a policy that supplies action proposals and supports value learning. The TD-MPC family [1], [2] is a prominent realization of this architecture, combining latent world-model learning with Model Predictive Path Integral (MPPI) control [3]. This architecture creates a closed planning–learning loop. The online planner generates experience; replayed experience updates the world model, critic, and actor; the critic then returns to the planner as a terminal value estimator; and the actor supplies both planning proposals and actions for TD bootstrapping. Errors or mismatches at one interface can therefore influence the data collected at the next. Recent TD-MPC variants have focused primarily on the planner–policy interface by constraining or distilling the actor toward planner behavior [4]–[7].

Policy alignment, however, addresses only one part of this loop. First, standard one-step TD learning exposes the critic to only the immediate observed reward before bootstrapping, so later rewards affect earlier values only through subsequent updates. Second, the same learned critic supplies terminal values for model-generated MPC rollouts, where uncertain estimates can change trajectory ranking and hence future data collection. Third, planner-generated experience can vary substantially in quality, yet planner-to-policy transfer need not distinguish trajectories by their realized outcomes. These three interfaces-critic supervision, terminal-value evaluation, and planner-to-policy transfer motivate our approach.

We introduce PL-MPC (*Planning–Learning MPC*), a simple extension of the policy-constrained TD-M(PC)^2 [4] backbone that modifies these three interfaces. *Multi-step TD targets* (`MTD`) incorporate observed rewards from planner-generated replay before the terminal bootstrap, using learned-model predictions only when the target extends beyond the sampled replay slice. An *adaptive terminal estimate* (`ATE`) penalizes MPPI terminal values when the target-critic ensemble disagrees, reducing reliance on uncertain value estimates during planning. Finally, *return-weighted actor distillation* (`RAD`) weights imitation of planner-executed

actions by realized episode return rather than by the current critic. These changes leave the latent world model, critic architecture, and MPPI optimization procedure unchanged.

We evaluate PL-MPC on thirteen HumanoidBench locomotion tasks [8] and four high-dimensional DMControl tasks [9]. The largest gains over TD-M(PC)^2 occur on `balance-hard`, where Total Average Return (TAR) increases from 98 ± 18 to 387 ± 255, and `hurdle`, from 199 ± 13 to 466 ± 200, while performance across the broader benchmark remains task dependent. Leave-one-out ablations reveal different component interactions across the two tasks, and a warmup study shows that `RAD` is sensitive to its activation schedule.

We further evaluate zero-shot sim-to-real wrench–nut alignment on a 7-DoF KUKA iiwa14. Policies trained entirely in Isaac Gym [10] are deployed without real-world fine-tuning using RGB-D-based 6D pose tracking [11]. PL-MPC achieves higher observed success than TD-M(PC)^2 on the training object size and two unseen sizes.

Our contributions are:

- We formulate policy-constrained TD-MPC as a coupled planning–learning loop and introduce PL-MPC, which modifies critic supervision, MPC terminal-value evaluation, and planner-to-policy transfer without changing the underlying world model or MPPI optimizer.
- Through controlled ablations and training diagnostics, we show that these mechanisms interact in a task-dependent manner, with the largest gains on the difficult HumanoidBench `balance-hard` and `hurdle` tasks.
- We demonstrate zero-shot sim-to-real wrench–nut alignment on a 7-DoF KUKA IIWA14, including transfer to object sizes unseen during simulation training.

II. Related Work

A. Planning with learned models and value expansion.

Learned world models support policy learning through imagined trajectories [12] and online control through finite-horizon planning with learned terminal values [1], [2], [13]. POLO combines online trajectory optimization and value learning while assuming access to an accurate internal dynamics model [14]. Palenicek et al. [15] find diminishing gains from longer value-expansion horizons even with oracle dynamics; in their experiments, model-free value expansion performs comparably to model-based variants, suggesting that model error is not the sole bottleneck. Our `MTD` update instead uses a hybrid target: replay rewards from the planner-generated trajectory are used where available, and a learned-model tail is introduced only when the target extends beyond the sampled replay slice.

B. Planner–policy alignment.

Online MPC can generate behavior that differs from the actor used for policy updates and value bootstrapping, creating a planner–policy mismatch [13]. Recent TD-MPC variants address this interface in different ways. TD-M(PC)^2 constrains the actor toward planner behavior stored in replay [4]; BMPC imitates an MPC expert and refreshes planner supervision through lazy reanalysis [5]; BOOM combines value maximization with value-weighted imitation of planner behavior [6]; and PO-MPC regularizes policy optimization toward an adaptive planning prior [7]. These methods primarily improve alignment between planner behavior and the learned policy. PL-MPC builds on TD-M(PC)^2 but also changes critic supervision and the terminal values used to rank MPPI trajectories.

C. Outcome-weighted policy learning and uncertainty.

Weighted policy-regression methods such as AWR, AWAC, CRR, and IQL emphasize actions according to estimated value or advantage [16]–[19]. Guided policy search similarly transfers behavior from a trajectory optimizer to a parameterized policy [20]. Most closely related to `RAD`, TDMPBC/SIRL weights self-imitation using trajectory-level realized return for humanoid control [21]. PL-MPC also uses realized episode return, but applies it specifically to planner-executed replay actions within a policy-constrained TD-MPC loop. Ensemble uncertainty has also been used to control value expansion. STEVE weights value-expansion horizons according to model and value uncertainty [22]. In contrast, `ATE` leaves the TD regression target unchanged and instead penalizes the terminal critic value used to rank MPPI trajectories. Thus, the three PL-MPC mechanisms act at distinct points in the same planning–learning loop: critic supervision, terminal trajectory evaluation, and planner-to-policy transfer.

III. Preliminaries

A. Control objective and latent world model

We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ with continuous state and action spaces. A policy $\pi(a \mid s)$ maximizes the expected discounted return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. Its action-value function $Q^\pi(s, a)$ is the expected return after taking a in s and following π thereafter. TD-MPC [1], [2] learns a control-oriented latent world model. An encoder h_θ maps observations to latent states $z_t = h_\theta(s_t)$, a dynamics model $f_\theta(z_t, a_t)$ predicts the next latent state, and a reward model $R_\theta(z_t, a_t)$ predicts immediate reward. These components are trained for control rather than observation reconstruction. A stochastic actor $\pi_\theta(a \mid z)$ and an ensemble of M action-value functions $\{Q_\theta^{(i)}\}_{i=1}^{M}$ are learned alongside the world model.

B. Planning and temporal-difference learning

At each environment step, TD-MPC performs online trajectory optimization with Model Predictive Path Integral (MPPI) control [3]. For a candidate sequence $a_{0:H-1}$, the learned model rolls out a latent trajectory and scores it by

$$\sum_{h=0}^{H-1} \gamma^h R_\theta(z_h, a_h) + \gamma^H V_{\mathrm{term}}(z_H), \qquad z_{h+1} = f_\theta(z_h, a_h).$$

The actor provides proposal trajectories, MPPI refines its sampling distribution according to these scores, and only the first optimized action is executed before replanning. The baseline planner terminal value is $V_{\mathrm{term}}(z_H) = Q_\theta^{\mathrm{ra}}(z_H, \pi_\theta(z_H))$,

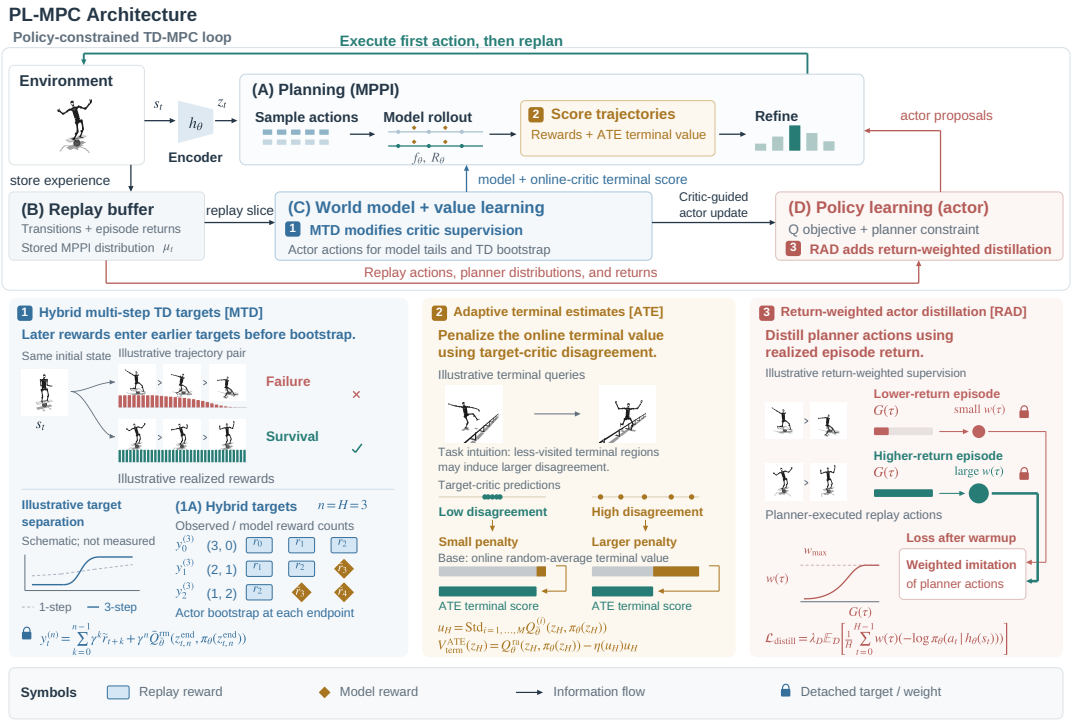


Fig. 2. **PL-MPC architecture.** The upper loop comprises MPPI planning (A), replay (B), world-model and value learning (C), and actor learning (D). Matching numbered badges link each modification to its detailed panel. **1: `MTD`** constructs critic targets in C from replay rewards, using actor-driven model tails beyond the available replay slice; detail 1A shows the observed/model reward counts. **2: `ATE`** penalizes the planner's online random-average terminal value in A using target-critic ensemble disagreement. **3: `RAD`** uses realized episode returns to weight distillation of planner-executed replay actions in D. The actor supplies planning proposals and actions for model tails and TD bootstraps. Arrows show information flow; locks denote detached targets or weights. Dotted and solid target-separation curves illustrate one-step and multi-step TD supervision, respectively; these curves, ensemble spreads, and value/weight diagrams are schematic.

where the random-average operator Q_θ^{ra} uniformly samples two distinct heads from the online critic ensemble and averages their decoded values. For compactness, $\pi_\theta(z)$ denotes an action sampled from the stochastic actor when used as a critic input. The critic is trained from replay. For a nonterminal transition, the standard one-step TD target is

$$y_t^{(1)} = r_t + \gamma Q_{\bar{\theta}}^{\mathrm{rm}}(z_{t+1}, \pi_\theta(z_{t+1}))\,. \tag{1}$$

Here $\bar{\theta}$ denotes target-critic parameters updated by Polyak averaging of the online critic parameters. The random-min operator $Q_{\bar{\theta}}^{\mathrm{rm}}$ uniformly samples two distinct target-critic heads and returns the smaller of their decoded values. Thus, online critics provide the planner terminal score, whereas target critics provide the TD bootstrap.

Figure 2 connects MPPI planning (A), replay (B), world-model and value learning (C), and actor learning (D). The replay action a_t is selected by MPPI, whereas the continuation action in Eq. (1) is sampled from the actor. Replay supplies training transitions through B→C; the learned world model and online critic then provide the rollout predictions and terminal scores used by the planner in A.

C. Policy-constrained TD-MPC

Our backbone is TD-M(PC)^2 [4], which constrains actor updates toward the planner distribution stored in replay. Let $\mu_t(\cdot \mid z_t)$ denote the MPPI proposal associated with a replay transition. Suppressing implementation-specific normalization, the actor objective is

$$\begin{aligned}\mathcal{L}_\pi^{\mathrm{PC}} = &-\mathbb{E}_{a\sim\pi_\theta(\cdot|z)}\left[Q_\theta(z,a)\right] - \alpha\mathcal{H}[\pi_\theta(\cdot \mid z)] \\ &+ \beta_{\mathrm{prior}}\mathbb{E}_{a\sim\pi_\theta(\cdot|z)}\left[-\log\mu_t(a \mid z)\right].\end{aligned} \tag{2}$$

The first term uses the online critic to favor high-value actions (C→D in Fig. 2), while the entropy term regularizes the stochastic actor. The planner-prior term uses μ_t from replay (B→D) to discourage actions outside the stored planner distribution.

IV. Method

As shown in Fig. 2, PL-MPC modifies the critic target in C (`MTD`, panel 1), the MPPI terminal value in A (`ATE`, panel 2), and the actor objective in D (`RAD`, panel 3). The latent world model, critic-ensemble architecture, and MPPI optimization procedure are otherwise unchanged.

A. Hybrid multi-step TD targets (`MTD`)

The one-step target in Eq. (1) incorporates only the immediate observed reward before bootstrapping. We instead use

$$y_t^{(n)} = \sum_{k=0}^{n-1}\gamma^k\tilde{r}_{t+k} + \gamma^n\bar{Q}_{\bar{\theta}}^{\mathrm{rm}}\left(z_{t,n}^{\mathrm{end}}, \pi_\theta(z_{t,n}^{\mathrm{end}})\right). \tag{3}$$

Algorithm 1 PL-MPC training.

Require: replay buffer $\mathcal{D}$, online parameters θ, target parameters $\bar{\theta}$, training counter k

Interact with the environment using MPPI, applying ATE to the terminal score via Eq. (4); execute the first action and store the transition and planner proposal
Upon episode completion, compute $G(\tau)$, associate it with the episode's transitions.
Sample a contiguous replay slice $(s_{0:H}, a_{0:H-1}, r_{0:H-1}, G(\tau), \mu_{0:H-1}) \sim \mathcal{D}$
Compute the multi-step targets $y_t^{(n)}$ using Eq. (3)
Update the encoder, dynamics, reward model, and critics using the standard TD-MPC losses with $y_t^{(n)}$
Compute the backbone actor objective $\mathcal{L}_\pi^{\mathrm{PC}}$ (Eq. (2)) using the stored planner proposals $\mu_{0:H-1}$
Update the return-statistics queue from returns in the sampled minibatch
if $k \geq k_{\mathrm{warmup}}$ and return statistics are available **then**
  Compute $w(\tau)$ using Eq. (5)
  Compute $\mathcal{L}_{\mathrm{distill}}$ using Eq. (6)
  $\mathcal{L}_\pi \leftarrow \mathcal{L}_\pi^{\mathrm{PC}} + \mathcal{L}_{\mathrm{distill}}$
else
  $\mathcal{L}_\pi \leftarrow \mathcal{L}_\pi^{\mathrm{PC}}$
end if
Update the actor using $\mathcal{L}_\pi$
Polyak-update target parameters $\bar{\theta}$

Whenever the corresponding transition lies within the sampled replay slice, $\widetilde{r}_{t+k} = r_{t+k}$ is the observed reward from the planner-executed trajectory. If the target extends beyond the available slice, the remaining rewards and latent states are obtained by rolling the learned dynamics and reward models forward under the current actor. Accordingly, $z_{t,n}^{\mathrm{end}}$ is either an encoded replay state or the endpoint of this model rollout.

The target is therefore hybrid rather than a fully observed n-step return. For the default $n = H = 3$, the targets at replay-slice positions $t = 0, 1, 2$ contain $(3, 0)$, $(2, 1)$, and $(1, 2)$ observed and model-predicted rewards, respectively (Fig. 2, detail 1A). The failure and survival trajectories in panel 1 illustrate how later observed rewards enter earlier critic targets directly, rather than only through successive one-step TD updates.

We substitute $y_t^{(n)}$ for $y_t^{(1)}$ in the standard TD-MPC categorical value loss, retaining its temporal weighting ρ^t with $\rho = 0.5$. The constructed target is stop-gradient: gradients update the critic prediction but do not propagate through the actor, target critic, or world-model computations used to construct the label. All other world-model and critic losses are unchanged. MTD changes the information supplied to value learning, not merely the discount applied to the terminal bootstrap. The target remains off-policy: its observed prefix follows historical planner actions, while its model-predicted tail and terminal bootstrap follow the current actor.

B. Adaptive terminal estimates (ATE)

ATE changes terminal evaluation within the MPPI planner in A (Fig. 2, panel 2), using target-critic disagreement at model-generated terminal latents to modulate the penalty applied to the online-critic terminal score.

For each terminal query, we compute $u_H = \mathrm{Std}_{i=1,\ldots,M} Q_{\bar{\theta}}^{(i)}(z_H, \pi_\theta(z_H))$, and use

$$V_{\mathrm{term}}^{\mathrm{ATE}}(z_H) = Q_\theta^{\mathrm{ra}}(z_H, \pi_\theta(z_H)) - \eta(u_H) u_H, \qquad \eta(u_H) = \eta_{\max} \sigma\left(\frac{u_H - \mu_u}{\sigma_u + \epsilon}\right). \tag{4}$$

Here $\sigma(\cdot)$ is the logistic sigmoid, and μ_u, σ_u are exponentially weighted estimates of the mean and standard deviation of terminal-query disagreement. The normalization scales the penalty relative to recently observed disagreement.

ATE modifies MPPI trajectory scoring only; it does not alter the TD target in Eq. (3). Ensemble disagreement is used as an uncertainty proxy, not as a calibrated estimate of critic error.

C. Return-weighted actor distillation (RAD)

RAD adds a loss to actor learning in D using planner-executed actions and episode returns from replay (B→D in Fig. 2, panel 3). The backbone constrains actor-sampled actions through the stored planner distribution μ_t; RAD additionally fits the executed actions a_t, weighted by their source episode's realized return.

Each replay transition is associated with the undiscounted episodic return $G(\tau) = \sum_{t=0}^{T_\tau - 1} r_t$ of its source trajectory. To normalize returns, we maintain a finite FIFO queue of episode returns encountered in sampled replay minibatches. Returns are deduplicated within each minibatch before being appended, but may re-enter the queue in subsequent updates. We compute G_{med} as the queue median and G_{std} as its population standard deviation, and define

$$w(\tau) = \mathrm{clip}\left[\exp\left(\frac{G(\tau) - G_{\mathrm{med}}}{\max(G_{\mathrm{std}}, \epsilon)}\right), 0, w_{\max}\right]. \tag{5}$$

Distinct finite returns are appended in ascending order within each minibatch; for an even-sized queue, G_{med} is the lower middle value. Higher-return trajectories therefore receive greater weight relative to recent replay experience, while clipping limits the influence of outliers. The additional actor loss is

$$\mathcal{L}_{\mathrm{distill}} = \lambda_D \mathbb{E}_{\mathcal{D}}\left[\frac{1}{H}\sum_{t=0}^{H-1} w(\tau)\left(-\log \pi_\theta(a_t \mid h_\theta(s_t))\right)\right]. \tag{6}$$

The return weight is treated as fixed during this update. Unlike Q-weighted planner imitation [6], the weighting signal is the realized trajectory return rather than the current critic estimate.

The actor is optimized with $\mathcal{L}_\pi = \mathcal{L}_\pi^{\mathrm{PC}} + \mathcal{L}_{\mathrm{distill}}$ once $k \geq k_{\mathrm{warmup}}$ and return statistics are available; otherwise, it is optimized with $\mathcal{L}_\pi^{\mathrm{PC}}$. Algorithm 1 summarizes one training iteration. The only additional replay annotation required by PL-MPC is the episodic return $G(\tau)$; TD-M(PC)^2 already stores the planner proposal used by its policy constraint.

V. Experiments

We evaluate PL-MPC on thirteen HumanoidBench locomotion tasks [8] and four high-dimensional DMControl tasks [9]. We first evaluate whether the proposed changes to value learning, MPC terminal-value estimation, and policy distillation improve tasks on which the TD-M(PC)^2 backbone remains weak, without changing its world-model architecture or MPPI planner. We then analyze the two tasks with the largest improvements to determine how individual components affect performance and training dynamics. Finally, we examine the sensitivity of return-weighted distillation to its

Task	DreamerV3	TD-MPC2	BMPC	BOOM	TD-M(PC)^2	PL-MPC (Ours)	Target
DMControl							
dog-stand	41 ± 9	523 ± 421	932 ± 21	$\mathbf{982 \pm 7}$	832 ± 99	976 ± 14	–
dog-trot	11 ± 3	394 ± 184	$\mathbf{953 \pm 10}$	923 ± 8	899 ± 39	885 ± 78	–
humanoid-stand	5 ± 1	637 ± 71	$\mathbf{955 \pm 8}$	920 ± 15	929 ± 15	954 ± 2	–
humanoid-walk	2 ± 0	643 ± 96	933 ± 20	921 ± 10	867 ± 63	$\mathbf{947 \pm 6}$	–
HumanoidBench Locomotion							
walk	161 ± 45	891 ± 43	651 ± 34	$\mathbf{928 \pm 16}$	922 ± 8	911 ± 7	700
stand	220 ± 74	754 ± 132	801 ± 13	915 ± 26	927 ± 48	$\mathbf{933 \pm 3}$	800
run	56 ± 11	196 ± 122	358 ± 152	596 ± 63	$\mathbf{852 \pm 8}$	658 ± 355	700
crawl	504 ± 90	822 ± 97	$\mathbf{922 \pm 17}$	855 ± 50	899 ± 64	855 ± 31	700
maze	117 ± 6	196 ± 36	348 ± 9	341 ± 2	347 ± 9	$\mathbf{353 \pm 3}$	1200
stair	43 ± 11	66 ± 14	445 ± 201	$\mathbf{462 \pm 75}$	387 ± 110	461 ± 12	700
slide	19 ± 6	210 ± 39	498 ± 18	858 ± 43	902 ± 24	$\mathbf{910 \pm 7}$	700
sit-simple	268 ± 26	359 ± 228	695 ± 105	882 ± 44	$\mathbf{932 \pm 18}$	910 ± 39	750
sit-hard	152 ± 20	744 ± 165	576 ± 44	740 ± 209	$\mathbf{821 \pm 64}$	635 ± 233	750
pole	123 ± 34	156 ± 25	676 ± 45	$\mathbf{879 \pm 33}$	874 ± 124	$\mathbf{879 \pm 49}$	700
balance-simple	15 ± 1	120 ± 24	613 ± 78	$\mathbf{784 \pm 101}$	599 ± 389	765 ± 176	800
hurdle	13 ± 3	86 ± 26	165 ± 64	331 ± 45	199 ± 13	$\mathbf{466 \pm 200}$	700
balance-hard	17 ± 4	102 ± 17	104 ± 23	102 ± 11	98 ± 18	$\mathbf{387 \pm 255}$	800

Fig. 3. **Benchmark performance.** Total Average Return (TAR; mean±std across seeds). Green and gray cells denote the highest and second-highest row means, respectively; ties share the best highlight. Target denotes the HumanoidBench reference-return threshold and is distinct from the environment-defined success metric.

activation time. Zero-shot physical deployment is evaluated separately in Sec. VI.

A. Experimental setup

For HumanoidBench, DreamerV3 and TD-MPC2 results are computed from the publicly released 2M-step evaluation logs, while BMPC, BOOM, TD-M(PC)^2, and PL-MPC are trained locally for 3M environment steps under the same task protocol. For every method, each seed's reported score is the mean return over its final five evaluations, and tables report mean±std across seeds. Local runs are evaluated every 50K steps using ten episodes. Unless otherwise stated, PL-MPC uses $n = H = 3$, ATE uses $\eta_{\max} = 0.5$ with decay 0.99, and RAD uses a 256-entry return-statistics queue with $w_{\max} = 10$, $\epsilon = 10^{-3}$, $\lambda_D = 0.5$, and a 200K-step warmup. This configuration is fixed across the main benchmark rather than tuned per task. Several HumanoidBench configurations exhibit substantial across-seed variability; comparisons below therefore refer to mean performance.

B. Benchmark performance

	balance-hard		hurdle	
Variant	TAR	Succ.	TAR	Succ.
TD-M(PC)$^2{}_{3M}$	98 ± 18	0.00	199 ± 13	0.00
w/o MTD	186 ± 51	0.00	530 ± 275	0.47
w/o ATE	206 ± 63	0.00	296 ± 21	0.00
w/o RAD	272 ± 221	0.15	231 ± 72	0.00
PL-MPC (Ours)	387 ± 255	0.19	466 ± 200	0.19

Fig. 4. **Leave-one-out component ablation.** Total Average Return (TAR) and environment-defined success rate (Succ.), averaged over three seeds. Each variant removes one component from full PL-MPC; when enabled, RAD uses the default 200K warmup.

PL-MPC shows its largest improvements over TD-M(PC)^2 on balance-hard, increasing TAR from 98 ± 18 to 387 ± 255, and on hurdle, from 199 ± 13 to 466 ± 200 (Fig. 3). Both tasks exhibit substantial across-seed variability, so these results reflect improvements in mean performance rather than consistent gains across all seeds.

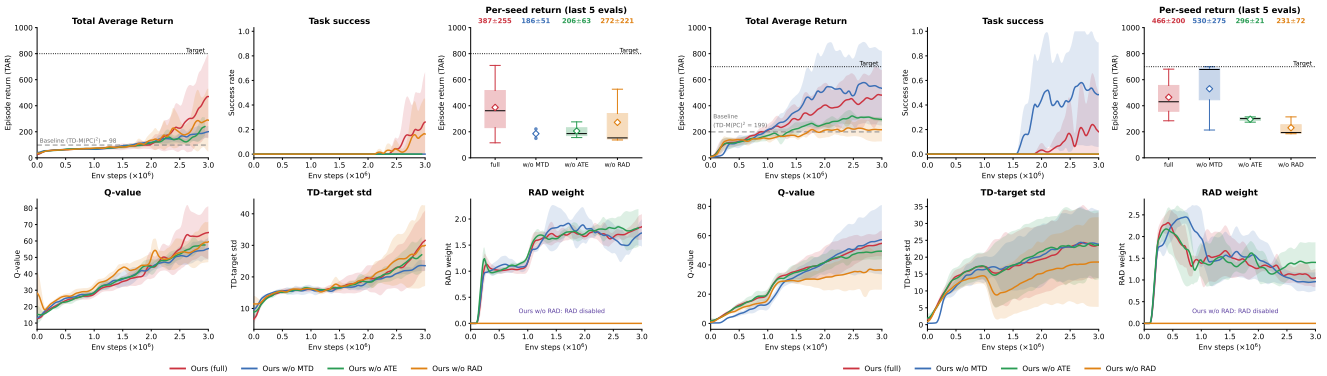
Across the thirteen HumanoidBench locomotion tasks, PL-MPC improves the mean over TD-M(PC)^2 on eight tasks and is lower on five. Of those five, PL-MPC nevertheless exceeds the HumanoidBench reference-return threshold on walk, crawl, and sit-simple; on these tasks, the gap to TD-M(PC)^2 is modest, averaging 26 TAR. The larger degradations are concentrated on run and sit-hard, where PL-MPC also remains below the reference threshold. Among all compared methods, PL-MPC attains the highest mean TAR on stand, maze, slide, hurdle, and balance-hard, and ties BOOM on pole.

On the four DMControl tasks with complete baseline coverage, PL-MPC remains competitive, with the highest mean on humanoid-walk. We therefore focus the analyses below on balance-hard and hurdle, where the changes relative to the backbone are largest.

C. Component ablations and training dynamics

The three modifications affect different parts of the TD-MPC update: MTD changes the critic target, ATE changes the terminal value used for MPC trajectory scoring, and RAD adds return-weighted policy distillation from planner-executed actions. We evaluate their conditional contributions on balance-hard and hurdle by removing one component at a time from full PL-MPC. Fig. 4 summarizes final performance, while Fig. 5 shows the corresponding learning curves and training diagnostics.

On balance-hard, removing any component reduces mean TAR, with the largest reduction occurring without MTD (387 ± 255 to 186 ± 51). As shown in Fig. 5a, the variants remain relatively close early in training and diverge later, when some full-PL-MPC seeds reach substantially higher



(a) `balance-hard`. (b) `hurdle`.

Fig. 5. **Leave-one-out ablations and training dynamics.** Each panel compares full PL-MPC with variants that remove `MTD`, `ATE`, or `RAD`. Top: TAR, environment-defined success, and per-seed return averaged over the final five evaluations; dotted lines denote HumanoidBench reference-return thresholds. Bottom: mean critic value, TD-target standard deviation, and `RAD` weight. Curves show mean±1 standard deviation across seeds. The TD-M(PC)^2 final return is shown for reference.

returns. The accompanying critic and TD-target statistics characterize training dynamics rather than value-estimation accuracy.

The component effects differ on `hurdle`. Removing `ATE` or `RAD` reduces mean TAR to 296 ± 21 and 231 ± 72, respectively, whereas removing `MTD` yields 530 ± 275 and 0.47 success, exceeding the mean performance of full PL-MPC in this experiment. The component effects are therefore task dependent and non-additive: on `hurdle`, removing `MTD` increases the observed mean, whereas removing `ATE` or `RAD` lowers it.

D. Targeted ablations and training diagnostics

The leave-one-out study measures conditional component effects around full PL-MPC. We next perform two targeted ablations that isolate specific component interactions and examine their training dynamics (Fig. 6).

a) `balance-hard`: effect of multi-step TD targets: With `ATE` and `RAD` fixed, replacing the one-step TD target with the hybrid $n = 3$ target increases mean return from 186 ± 51 to 387 ± 255. The difference is also reflected in task success: all seeds without `MTD` remain at 0% success, whereas full PL-MPC averages 19%, with the best seed succeeding on 50.9% of evaluation episodes. On this task, success requires maintaining balance for the full 1000-step episode, so the successful seeds correspond to qualitatively different behavior rather than a modest increase in accumulated return. The learning curves remain similar early in training and separate later, when these successful behaviors emerge. The critic-value and TD-target statistics characterize the accompanying training dynamics rather than value-estimation accuracy.

b) `hurdle`: effect of `ATE` and `RAD` with `MTD` fixed: Both PL-MPC variants use `MTD`. Adding `MTD` alone to the TD-M(PC)^2 backbone yields 288 ± 116 mean return, compared with 199 ± 13 for TD-M(PC)^2. Adding `ATE` and `RAD` jointly raises the mean to 465 ± 201. The behavioral difference is also visible in task success: the matched baseline seeds remain at 0%, whereas full PL-MPC averages 19% success and its best seed reaches 58.0%. Combined with the leave-one-out ablation, these results show that the components interact: `MTD` improves over the backbone in this targeted comparison, whereas removing `MTD` from full PL-MPC does not reduce mean performance on `hurdle`.

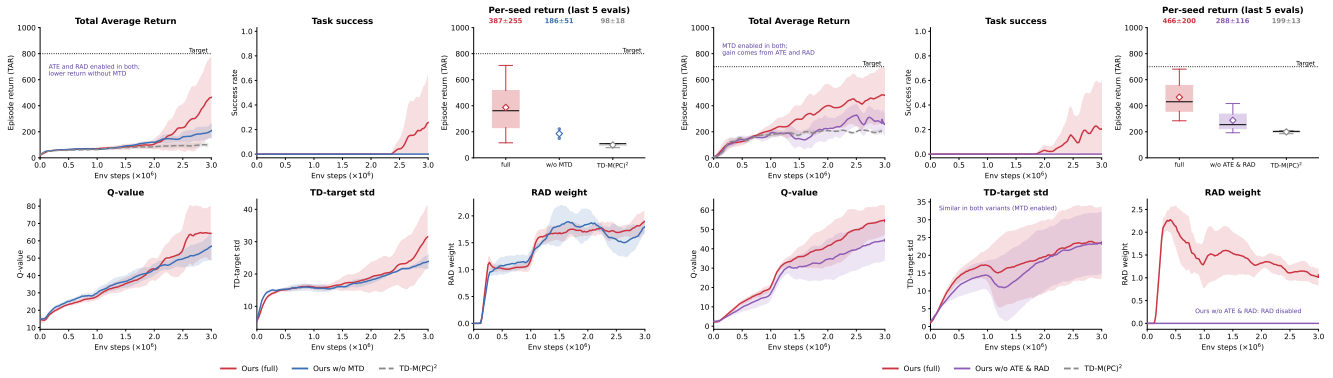
E. Sensitivity to the `RAD` warmup

PL-MPC with the default 200K `RAD` warmup underperforms its TD-M(PC)^2 backbone on five HumanoidBench tasks. Because 200K occurs early in a 3M-step training run, distillation may begin before planner behavior has sufficiently improved. We therefore test whether delaying `RAD` activation changes these performance gaps. Fig. 7 compares the default 200K warmup with 750K on six tasks; the main benchmark retains the fixed 200K setting.

A 750K warmup improves `crawl`, `balance-simple`, and `sit-hard`, while `walk` and `maze` change little. It decreases return and success on `hurdle`. Thus, no fixed warmup dominates across these tasks. Delaying distillation closes the performance gap to TD-M(PC)^2 on `crawl` and substantially narrows it on `sit-hard`. We retain 200K for the main benchmark and treat adaptive scheduling of planner-to-policy distillation as future work.

VI. Real-Robot Wrench–Nut Alignment

a) Task and setup: We evaluate zero-shot sim-to-real transfer on the wrench–nut alignment benchmark introduced in [23]. A 7-DoF KUKA IIWA14 equipped with a Robotiq 3-Finger gripper places a grasped wrench over a nut threaded onto an upright bolt (Fig. 1). Contact can rotate the nut during execution, changing the desired wrench pose and requiring closed-loop realignment. Policies are trained entirely in Isaac Gym [10] and deployed without real-world fine-tuning. FoundationPose [11] provides 6D object-pose estimates from a fixed RealSense D435 RGB-D camera. Observations express the current and target wrench poses in the nut frame, and actions specify an $\mathrm{SE}(3)$ displacement. Simulation randomizes the initial object configuration and observation



(a) `balance-hard`: effect of `MTD`. (b) `hurdle`: effect of adding `ATE` and `RAD` to `MTD`.

Fig. 6. **Targeted ablations on the two analysis tasks.** (a) On `balance-hard`, `ATE` and `RAD` are fixed and only `MTD` is removed. (b) On `hurdle`, both variants use `MTD`; the comparison adds `ATE` and `RAD` jointly. Top: episode return, task success, and per-seed return averaged over the final five evaluations; dotted lines denote HumanoidBench reference-return thresholds. Bottom: mean critic value, standard deviation of the TD target, and `RAD` weight. The TD-M(PC)^2 backbone is shown in the performance panels for reference. Diagnostic panels compare the corresponding PL-MPC variants. Curves show mean$\pm$1 standard deviation across seeds.

Task	**TD-M(PC)^2**	**PL-MPC (Ours)**			
		200K	**750K**	**Δ**	**Succ. 200K/750K**
`crawl`	899 ± 64	855 ± 31	971 ± 6	**+116**	0.99/0.99
`maze`	347 ± 9	353 ± 3	349 ± 8	−4	0.00/0.00
`walk`	922 ± 8	911 ± 7	910 ± 1	−1	0.99/0.99
`balance-simple`	599 ± 389	765 ± 176	809 ± 32	**+44**	0.63/0.65
`hurdle`	199 ± 13	466 ± 200	414 ± 127	−52	0.19/0.09
`sit-hard`	821 ± 64	635 ± 233	805 ± 104	**+170**	0.52/0.84

Fig. 7. **Sensitivity to `RAD` warmup.** Total Average Return (TAR; mean$\pm$std across three seeds) for the TD-M(PC)^2 backbone and PL-MPC with 200K and 750K `RAD` warmups. Δ denotes the change from 200K to 750K; Succ. reports the environment-defined success rates for the two warmups.

noise, while the low-level controller executes the commanded end-effector displacements. We evaluate three matched nut–bolt–wrench asset sizes. Size 5, used during simulation training, has a nut width across flats of 46 mm. Sizes 3 and 1 are unseen during training and measure 36 mm and 30 mm, respectively. Nut height (21.5 mm) and wrench thickness (15.3 mm) are fixed across the three sizes.

b) Simulation training: Training episodes contain 256 control steps and start from the hardest curriculum stage, with no curriculum progression during training. Both methods use the shaped reward

$$r_t = 0.5 \exp\left(-\frac{d_t^2}{2(0.03)^2}\right) + 0.3\,\mathbf{1}[\text{engaged}] + 0.2\,\mathbf{1}[\text{inserted}],$$

where

$$d_t = \left\| p_{\text{head}} - (p_{\text{nut}} + 5\ \text{mm}\ \hat{z}) \right\|_2 .$$

The `engaged` term requires $|\Delta z| < 10.75$ mm and an xy displacement below 50 mm, while the `inserted` term requires $|\Delta z| < 5$ mm and an xy displacement below 50 mm. Simulation success uses a separate pose criterion. At the final step, the wrench-head origin must lie within ± 5 mm of the nut origin in height, and the summed distance between four corresponding keypoints on the wrench-head and nut axes must be below 50 mm. The criterion constrains height, lateral alignment, and tilt while remaining invariant to rotation about the nut axis.

For hardware deployment, we use one independently trained policy per method and select its checkpoint using simulation evaluation within the first 500K training steps. Policies are evaluated for ten MPC episodes every 10K steps, and the checkpoint with the highest evaluation score is retained. For the selected checkpoints, a separate 200-episode simulation evaluation on training size 5 yields terminal-step success rates of 81.0% for PL-MPC and 67.5% for TD-M(PC)^2; these episodes are not used for checkpoint selection.

c) Hardware evaluation: A trial is successful when the wrench head is inserted onto the nut before the time limit; otherwise it terminates after 128 control steps. We conduct 31 trials per method on the training size and 15 trials per method on each unseen size. For successful trials, we additionally report completion steps and final position and rotation errors.

Method	Size	N	SR (%)	95% CI	Steps	Pos. (mm)	Rot. (deg)
Training object size							
TD-M(PC)^2	5	31	61.3	[43.8, 76.3]	53.3 ± 24.4	11.08 ± 4.31	7.28 ± 4.26
PL-MPC (Ours)	5	31	**74.2**	[56.8, 86.3]	55.9 ± 25.9	9.86 ± 4.50	6.14 ± 2.35
Unseen object sizes							
TD-M(PC)^2	3	15	73.3	[48.0, 89.1]	54.1 ± 16.0	10.60 ± 4.54	3.57 ± 1.92
PL-MPC (Ours)	3	15	**80.0**	[54.8, 93.0]	47.9 ± 14.6	11.57 ± 4.41	4.64 ± 1.55
TD-M(PC)^2	1	15	20.0	[7.0, 45.2]	52.0 ± 7.2	7.89 ± 3.00	3.69 ± 1.86
PL-MPC (Ours)	1	15	**33.3**	[15.2, 58.3]	45.2 ± 7.6	9.75 ± 3.78	6.91 ± 3.61
TD-M(PC)^2	Pooled	30	46.7	[30.2, 63.9]	53.6 ± 14.3	10.02 ± 4.31	3.60 ± 1.84
PL-MPC (Ours)	**Pooled**	30	**56.7**	[39.2, 72.6]	47.1 ± 12.8	11.03 ± 4.20	5.31 ± 2.46

Fig. 8. **Zero-shot real-robot wrench–nut alignment.** Success rate (SR) is computed over all trials with 95% Wilson confidence intervals. Steps and final position/rotation errors are mean$\pm$sample standard deviation over successful trials. Size 5 is used during simulation training; sizes 3 and 1 are unseen. Pooled rows combine the two unseen sizes. **Bold green** denotes the higher observed SR.

d) Real-robot results: PL-MPC achieves a higher observed success rate on all three object sizes: 74.2% versus 61.3% on the training size, 80.0% versus 73.3% on unseen

size 3, and 33.3% versus 20.0% on unseen size 1 (Fig. 8). Across the two unseen sizes, pooled success increases from 46.7% to 56.7%. PL-MPC also completes successful trials in fewer control steps on both unseen sizes and in the pooled comparison. Figure 1 shows representative executions, including successful zero-shot transfer to both unseen sizes. Final pose errors are comparable overall and do not favor either method consistently. The hardware study demonstrates zero-shot sim-to-real deployment and transfer to object sizes unseen during training, with the deployed PL-MPC policy attaining a higher observed success rate than the deployed TD-M(PC)^2 policy on each tested size.

VII. Conclusions

We presented PL-MPC, which modifies critic supervision, MPC terminal-value evaluation, and planner-to-policy transfer within policy-constrained TD-MPC. Its largest gains occur on HumanoidBench `balance-hard` and `hurdle`, while improvements are less consistent on tasks such as `maze`, where useful planner trajectories remain rare and return-weighted distillation has little signal to amplify. HumanoidBench's per-step survival reward can further obscure behavioral differences: policies may accumulate substantial return by remaining stable without completing the task, while successful seeds can occupy a distinct higher-return regime. This behavior also contributes to the large across-seed variability observed on the hardest tasks. On `balance-hard`, multi-step TD targets have the largest leave-one-out effect, but outcomes remain strongly seed dependent. On `hurdle`, the components interact differently; the targeted comparison adds `ATE` and `RAD` jointly, while the leave-one-out study provides their conditional effects rather than an additive decomposition. Performance is also sensitive to when `RAD` is activated: delaying distillation improves several tasks but degrades `hurdle`, motivating adaptive rather than fixed warmup schedules. Zero-shot deployment further demonstrates transfer to the physical wrench–nut task and to object sizes unseen during simulation training.

Although the proposed modifications are evaluated only on TD-M(PC)^2, they act at interfaces shared by other TD-MPC-style methods, making systematic transfer across planning–learning backbones a natural direction for future work.

Acknowledgment

OpenAI's ChatGPT was used to assist with improving the text, and presentation of the paper.

References

[1] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 8387–8406.

[2] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations*, 2024.

[3] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[4] H. Lin, P. Wang, J. Schneider, and G. Shi, "TD-M(PC)^2: Improving temporal difference MPC through policy constraint," in *Proceedings of the 8th Annual Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, vol. 331. PMLR, 2026, pp. 705–736.

[5] Y. Wang, H. Guo, S. Wang, L. Qian, and X. Lan, "Bootstrapped model predictive control," in *International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=i7jAYFYDcM

[6] G. Zhan, L. Wang, X. Zhang, J. Gao, M. Tomizuka, and S. E. Li, "Bootstrap off-policy with world model," in *Advances in Neural Information Processing Systems*, vol. 38, 2025, pp. 148 608–148 636.

[7] A. Serra-Gomez, D. Jarne Ornia, D. Tirumala, and T. Moerland, "A KL-regularization framework for learning to plan with adaptive priors," in *Proceedings of the 43rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 306, 2026.

[8] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel, "Humanoid-Bench: Simulated humanoid benchmark for whole-body locomotion and manipulation," in *Robotics: Science and Systems*, 2024.

[9] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[10] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance GPU based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_

[11] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.

[12] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse control tasks through world models," *Nature*, vol. 640, pp. 647–653, 2025.

[13] H. Sikchi, W. Zhou, and D. Held, "Learning off-policy with online planning," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 2022, pp. 1622–1633.

[14] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *International Conference on Learning Representations*, 2019.

[15] D. Palenicek, M. Lutter, J. Carvalho, and J. Peters, "Diminishing return of value expansion methods in model-based reinforcement learning," in *International Conference on Learning Representations*, 2023.

[16] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[17] A. Nair, A. Gupta, M. Dalal, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[18] Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas, "Critic regularized regression," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 7768–7778.

[19] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

[20] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 28, no. 3. PMLR, 2013, pp. 1–9.

[21] Z. Zhuang, D. Shi, R. Suo, X. He, H. Zhang, T. Wang, S. Lyu, and D. Wang, "TDMPBC: Self-imitative reinforcement learning for humanoid robot control," *arXiv preprint arXiv:2502.17322*, 2025.

[22] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] Y. Liu, X. Zhang, H. Chang, and A. Boularias, "Failure forecasting boosts robustness of sim2real rhythmic insertion policies," 2025. [Online]. Available: https://arxiv.org/abs/2507.06519