BACK TO STATE: EXPLICIT PHYSICAL STATE WORLD MODELS FOR CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal difference learning for model predictive control (TD-MPC) and its recent variants achieve strong continuous control performance by learning task-oriented latent dynamics and performing short-horizon planning entirely in latent space. For state-based control, however, this removes the physical semantics from imagined trajectories and makes physical state predictions unavailable for deployment-time objectives. Prior work has attempted to recover explicit state predictions from latent world models, but reconstruction-based approaches can suffer substantial performance degradation on challenging high-dimensional continuous control tasks. In this work, we introduce State TD-MPC (S-TD-MPC), which directly propagates the normalized physical state while augmenting it with learned latent features recomputed at each imagined step, providing explicit physical state predictions while retaining strong representation capacity. We further align value learning with MPC through MPC-based Bellman targets and introduce *reanalysis via Bellman target bounds* to refresh stale planner targets selectively. Beyond the training objective, we study three capabilities enabled by explicit state predictions: zero-shot transfer across tasks with shared dynamics, deployment-time state constraints, and analytic reward evaluation during training. Experiments on challenging continuous control tasks show that S-TD-MPC matches the performance of state-of-the-art methods in the TD-MPC family, achieving an average return 2.2% higher than the strongest baseline, while enabling these constraint-aware and transferable control capabilities. Videos and supplementary materials are available here.

1 INTRODUCTION

World models have become a powerful tool for sample-efficient continuous control by allowing agents to reason about future actions before interacting with the environment (Ha & Schmidhuber, 2018; M. Moerland et al., 2023). TD-MPC (Hansen et al., 2022) established an effective framework that combines short-horizon model predictive control (MPC) with temporal-difference value learning, and TD-MPC2 (Hansen et al., 2024) substantially improved its robustness and scalability across challenging tasks. A central design choice in this line of work is to encode observations into a task-oriented latent representation and perform imagined rollouts entirely in that latent space.

Latent dynamics are natural for high-dimensional and unstructured observations such as images (Hafner et al., 2019b;a; Watter et al., 2015; Lee et al., 2020). Their necessity is less clear in state-based control, where observations are already compact, Markovian, and physically meaningful, such as joint configurations, velocities, and contacts. Encoding these quantities into an unconstrained latent representation can be highly effective for control, but removes their direct physical semantics from imagined trajectories. Previous work has attempted to recover the original state through an additional decoder. Such reconstruction has achieved comparable performance on relatively simple control tasks (Matthies et al., 2023), but has been reported to degrade performance on more challenging high-dimensional control domains and, in some cases, hinder learning altogether (Hansen et al., 2022; Zhao et al., 2023; Scannell et al., 2025). This leaves an important practical gap within the TD-MPC family: a world model that provides explicit physical-state predictions while retaining the strong control performance achieved by state-of-the-art methods in this family.

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.

We introduce State TD-MPC (S-TD-MPC), a world-modeling approach built on the TD-MPC family of algorithms that explicitly predicts the environment state while retaining learned latent features. Instead of replacing the environment state with a learned latent variable, our world model directly propagates the physical state and augments it with nonlinear latent features that are recomputed from the predicted state at every imagined step. Reward, value, and policy models jointly use the physical state and these learned features, retaining expressive representations while keeping imagined trajectories explicitly grounded in the environment state. We further align value and policy learning with MPC through MPC-based Bellman targets and Reanalysis via Bellman Target Bounds, together with theoretical analyses of the resulting value backups.

Beyond control performance, explicit state prediction also enables quantities defined in the physical state space to be used directly during planning. We study this through analytic rewards, state-dependent constraints introduced at test time, and transfer between tasks that share the same dynamics but differ in their objectives. Such downstream uses of explicit state predictions have been comparatively underexplored in reconstruction-based extensions of TD-MPC (Matthies et al., 2023; Zhao et al., 2023; Scannell et al., 2025). Our focus is whether these capabilities can be achieved while retaining the strong control performance of modern TD-MPC.

2 Related work

TD-MPC and search-based continuous control. TD-MPC combines latent world model learning with short-horizon MPC and temporal-difference value learning, forming a strong family of methods for continuous control (Hansen et al., 2022; 2024). Several recent works improve the interaction between planning and learning. BMPC trains the policy to imitate an MPC expert and uses the resulting policy to guide subsequent planning (Wang et al., 2025); TD-M(PC)^2 regularizes the learned policy to reduce the mismatch between planner-generated data and policy-based value learning (Lin et al., 2025); and BOOM introduces explicit behavior alignment between the policy and online planner (Zhan et al., 2026). Other work focuses on the planning procedure itself, including gradient-based trajectory optimization in Dream-MPC (Spieler & Behnke, 2026) and uncertainty- and rollout-aware planning objectives in EfficientTDMPC (Evers et al., 2026). The TD-MPC framework has also been extended to settings such as massively multitask learning (Hansen et al., 2026), time-aware dynamics (Nhu et al., 2025), and visual generalization (Yang et al., 2023). Of particular relevance to our experiments, MRS.Q (Chang et al., 2026) builds on a TD-MPC2-style latent dynamics model with short-horizon MPPI, but focuses on search-induced value overestimation and mitigates it using aggressive minimization over a Q-function ensemble. We therefore include MRS.Q as a strong search-based baseline.

World model representations and state reconstruction. Model-based RL methods differ in the representation used for predicting dynamics. Latent world models such as PlaNet and Dreamer learn stochastic latent dynamics for planning or policy learning (Hafner et al., 2019b;a), while TD-MPC learns a task-oriented latent representation through temporal consistency (Hansen et al., 2022). Several TD-MPC-related works have additionally studied reconstructing the original state or observation from the latent representation, but such reconstruction has been reported to perform poorly on more challenging high-dimensional control tasks (Matthies et al., 2023; Zhao et al., 2023; Scannell et al., 2025). In contrast, another line of model-based RL directly learns transitions in the environment state space, including PILCO (Deisenroth & Rasmussen, 2011), neural-network dynamics with MPC (Nagabandi et al., 2018), PETS (Chua et al., 2018), MBPO (Janner et al., 2019), and recent stochastic multimodal models such as WIMLE (Aghabozorgi et al., 2026). Our approach lies between these two designs: it directly propagates the physical state while retaining learned latent features for nonlinear representation capacity.

Downstream uses of learned dynamics. A key advantage of model-based control, particularly MPC, is the ability to modify the cost function and impose state or action constraints directly at planning time (Garcia et al., 1989; Mayne et al., 2000). Learned dynamics models extend this flexibility by allowing such objectives and constraints to be evaluated on predicted trajectories (Andersson et al., 2015; Sikchi et al., 2022; Berkenkamp et al., 2017; Hewing et al., 2020). Another related direction studies transfer across tasks that share common dynamics. Prior work has considered reinforcement learning transfer when the transition dynamics remain fixed and only the reward changes

(Laroche & Barlier, 2017; Barreto et al., 2017; 2018), as well as representations that explicitly decouple dynamics and reward to facilitate transfer (Zhang et al., 2018). More recently, neurosymbolic world models have been used for zero-shot adaptation to new reward functions defined over a shared structured state space (Tamassia et al., 2026), while WOMBET (Kim & Sreenath, 2026) uses a learned source-task world model to generate experience for subsequent target-task adaptation. These works demonstrate the broader utility of semantically meaningful dynamics models. We investigate complementary capabilities within the TD-MPC setting, asking whether explicit state predictions can support analytic reward evaluation, deployment-time state constraints, and transfer between reward-defined tasks without sacrificing the control performance of latent TD-MPC methods.

3 Preliminaries

Continuous control. We consider a discounted Markov decision process $\mathcal{M}=(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ denote the state and action, $s_{t+1} \sim P(\cdot|s_t, a_t)$ is the environment transition, $r_t = r(s_t, a_t)$ is the reward, and $\gamma \in (0, 1)$ is the discount factor. Transitions are stored in a replay buffer $\mathcal{D}$. We use $Q(s, a)$ to denote an action-value function and Q^- its target network; when an ensemble is used, Q_i denotes its i^{th} member. H denotes the finite rollout horizon.

TD-MPC and MPPI. TD-MPC performs short-horizon MPC using a learned world model together with a terminal value estimate. At each state, MPPI (Williams et al., 2017) maintains a factorized Gaussian distribution over an H-step action sequence,

$$q_{\text{MPC}}(\mathbf{a}|s) = \prod_{j=0}^{H-1} \mathcal{N}\left(a_j; \mu_j^{\text{MPC}}, \text{diag}\left((\sigma_j^{\text{MPC}})^2\right)\right), \tag{1}$$

where $\mathbf{a}=(a_0, \ldots, a_{H-1})$. Candidate action sequences are evaluated by rolling them out through the learned model using predicted rewards and a terminal critic value. MPPI iteratively updates the search mean μ_j^{MPC} and standard deviation σ_j^{MPC}, and the first action of the optimized sequence is executed.

4 Method

Our goal is to retain the strong control performance and sample-efficient learning of the TD-MPC2 family while making imagined trajectories explicitly available in the environment state space. To achieve this, S-TD-MPC replaces recursive latent state prediction with direct propagation of the normalized physical state, augmented by latent features recomputed from each predicted state, as shown in Fig. 1. We first describe the resulting world model architecture, then present the training objectives for the world model and policy, including MPC-based value targets and selective reanalysis, and finally analyze the resulting value backups and stale-target error.

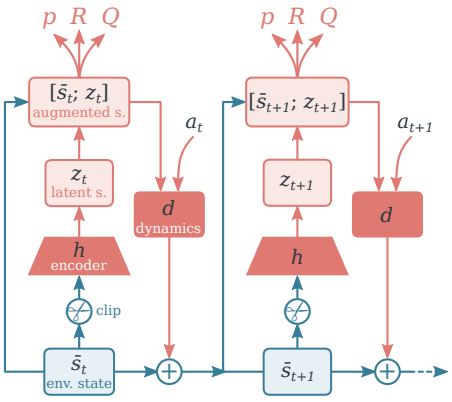


Figure 1: **S-TD-MPC architecture.**

4.1 S-TD-MPC World Model Architecture

Components. Our world model consists of five learnable components together with a running state normalizer. Below, $\bar{s} = (s - \mu_s)/\sigma_s$ denotes the normalized environment state, $\tilde{s} = \text{clip}(\bar{s}, -c, c)$ is its clipped version used as neural network input, and $z = h(\tilde{s})$ is a learned auxiliary latent feature:

Encoder	$z = h(\tilde{s})$	Extracts nonlinear features from the normalized state
State dynamics	$\bar{s}' = \bar{s} + d(\bar{s}, z, a)$	Predicts the next normalized state
Reward	$\hat{r} = R(\bar{s}, z, a)$	Predicts the reward of a transition
Terminal value	$\hat{q} = Q(\bar{s}, z, a)$	Predicts discounted return
Policy prior	$\hat{a} \sim p(\bar{s}, z)$	Provides an action prior for planning

Design. Unlike latent world models such as TD-MPC2, which recursively predict a learned latent state, we directly propagate the normalized environment state $\bar{s}$ through the model. The learned feature z augments the state rather than replacing it, such that all prediction heads operate on the combined representation $[\tilde{s}; z]$. The dynamics model uses a residual transition, $d(\bar{s}, z, a) = \delta_{\max} \tanh(f([\tilde{s}; z], a)/\delta_{\max})$, which bounds the predicted state change while retaining an explicit identity path. Importantly, clipping is used only for neural-network inputs, whereas the recursively propagated state remains unclipped. This design preserves an explicit physical state-space representation throughout planning, while learned latent features provide additional nonlinear representation capacity.

4.2 Training Objectives

Joint model and policy learning. We train the world model from trajectory segments sampled from the replay buffer. Starting from $\hat{\bar{s}}_0 = \bar{s}_0$, the dynamics model is recursively unrolled under the recorded action sequence, producing predicted states $\hat{\bar{s}}_{1:H}$ and corresponding latent features $z_t = h(\tilde{s}_t)$. The encoder, dynamics, reward, and value models are jointly optimized with

$$\mathcal{L}_{\mathrm{WM}} = \mathbb{E}_{(s,a,r,s')_{0:H} \sim \mathcal{D}} \left[\sum_{t=0}^{H-1} \rho^t \left(\lambda_s \mathcal{L}_s^t + \lambda_r \mathcal{L}_r^t + \lambda_Q \mathcal{L}_Q^t \right) \right], \tag{2}$$

where $\rho \in (0, 1]$ downweights temporally distant predictions. The individual objectives are

$$\mathcal{L}_s^t = \ell_{\mathrm{Huber}} \left(\hat{\bar{s}}_{t+1}, \bar{s}_{t+1} \right), \quad \mathcal{L}_r^t = \mathrm{CE} \left(\hat{r}_t, r_t \right), \quad \mathcal{L}_Q^t = \frac{1}{N_Q} \sum_{i=1}^{N_Q} \mathrm{CE} \left(\hat{q}_t^i, y_t \right), \tag{3}$$

where CE denotes categorical cross-entropy. Unlike the latent consistency objective used in TD-MPC2, our dynamics model is directly supervised against the observed normalized state, thereby constraining recursive model predictions in the physical state space. Following TD-MPC2 (Hansen et al., 2024), reward and value prediction are formulated as distributional regression: scalar reward and value targets are two-hot encoded and optimized with categorical cross-entropy (Bellemare et al., 2017). The planner-based value target y_t is introduced in the following paragraph.

The policy prior is trained separately using the action distribution produced by MPC (Wang et al., 2025). Let $q_{\mathrm{MPC}}(\cdot|s) = \mathcal{N}\left(\mu_{\mathrm{MPC}}, \mathrm{diag}(\sigma^2_{\mathrm{MPC}})\right)$ denote the first-step Gaussian of the final MPPI search distribution, and let $p_\theta(\cdot|s) = \mathcal{N}\left(\mu_\theta, \mathrm{diag}(\sigma^2_\theta)\right)$ denote the learned policy prior. For states with valid planner targets, we optimize

$$\mathcal{L}_\pi = \mathbb{E}_{s \sim \mathcal{D}} \left[\lambda_{\mathrm{KL}} \frac{D_{\mathrm{KL}} \left(p_\theta(\cdot|s) \,\|\, q_{\mathrm{MPC}}(\cdot|s) \right)}{S_{\mathrm{KL}}} - \lambda_{\mathcal{H}} \mathcal{H} \left(p_\theta(\cdot|s) \right) \right], \tag{4}$$

where D_{KL} denotes KL divergence, $\mathcal{H}$ denotes entropy, and S_{KL} is an online running scale of the KL objective used to stabilize optimization. Rather than imitating only the selected MPC action, this objective distills the full search distribution, allowing the policy prior to capture both the preferred action and the uncertainty of the planner.

MPC-based value targets. A key distinction lies in how the value target y_t is constructed. In TD-MPC2, the critic is bootstrapped using actions sampled from the learned policy prior,

$$y_t^\pi = r_t + \gamma Q^- \left(\bar{s}_{t+1}, a_{t+1}^\pi \right), \qquad a_{t+1}^\pi \sim p_\theta(\cdot|\bar{s}_{t+1}), \tag{5}$$

where Q^- denotes the target critic. However, the resulting value estimates are subsequently used to bootstrap MPC, creating a planner–value mismatch when MPC and the policy favor different actions. BMPC reduces this gap by training the policy to imitate the MPC expert (Wang et al., 2025), while BOOM introduces an explicit behavior-alignment objective between the policy and planner (Zhan et al., 2026). TD-M(PC)^2 likewise reduces the discrepancy by regularizing the learned policy toward replay behavior (Lin et al., 2025) that is more consistent with planner-induced actions. These approaches mitigate the MPC–policy mismatch by bringing the policy closer to planner behavior. We instead additionally address this mismatch at the value-learning level by defining the Bellman continuation directly using actions produced by MPC.

More generally, we introduce a k-step planner backup, where k controls how far the cached MPC trajectory is rolled out before value bootstrapping. Let $\{a_j^{\mathrm{MPC}}, \sigma_j^{\mathrm{MPC}}\}_{j=0}^{H-1}$ denote the cached MPC

solution starting from $\bar{s}_{t+1}$, and let $\hat{\bar{s}}^{(j)}$ denote the state obtained after j model transitions along this action sequence, with $\hat{\bar{s}}^{(0)} = \bar{s}_{t+1}$. For $0 \le k < H$, the target is

$$y_t = r_t + \gamma \left[\sum_{j=0}^{k-1} \gamma^j R\left(\hat{\bar{s}}^{(j)}, z^{(j)}, a_j^{\mathrm{MPC}}\right) + \gamma^k \min_{i \in \mathcal{I}} Q_i^-\left(\hat{\bar{s}}^{(k)}, z^{(k)}, \tilde{a}_k^{\mathrm{MPC}}\right) \right], \tag{6}$$

where $\mathcal{I}$ indexes the five target critics used in our experiments and

$$\tilde{a}_k^{\mathrm{MPC}} = a_k^{\mathrm{MPC}} + \eta\, \sigma_k^{\mathrm{MPC}} \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{7}$$

introduces local action perturbations according to the final MPPI search variance. For $k = H$, the terminal estimate follows the terminal bootstrap used by the planner. The special case $k = 0$ requires no imagined rollout and directly replaces the policy action in the one-step TD target with the current MPC action. Increasing k progressively incorporates model-predicted rewards along the MPC trajectory. Thus, k explicitly controls the trade-off between a purely value-based planner target and a longer model-based planner backup.

Reanalysis via Bellman Target Bounds. Planner targets stored in the replay buffer are generated by earlier versions of the world model and critic and therefore become increasingly stale as training progresses. Reanalysis addresses this issue by rerunning MPC on replay states to refresh outdated planner targets, as adopted in BMPC (Wang et al., 2025) and more recent TD-MPC variants such as EfficientTDMPC (Evers et al., 2026). However, reanalyzing all sampled transitions is computationally expensive and ignores that stale planner targets can have substantially different effects on value learning. We therefore perform budgeted reanalysis and selectively refresh transitions according to their estimated influence on the Bellman target.

For a replay transition i, let A_i denote the age of its cached planner target and define

$$g_i = \left\| \nabla_a Q^-\left(\bar{s}_i', a\right) \big|_{a=a_i^{\mathrm{MPC}}} \right\|_2, \tag{8}$$

which measures the local action sensitivity of the target critic around the cached MPC action. We assign each planner target the priority

$$S_i = A_i^2 + \beta A_i g_i, \tag{9}$$

where the first term prioritizes older planner targets, while the second emphasizes stale targets whose value estimates are more sensitive to changes in the planner action. As shown in the following section, this form is motivated by an upper bound on the Bellman-target error induced by stale planner actions.

At each reanalysis event, a fixed computation budget is allocated to the transitions with the largest priorities S_i. MPC is rerun in parallel on the selected replay states using the current world model and critic, and the corresponding cached action sequences, search means, search standard deviations, and planner versions are replaced by the newly computed solutions. The refreshed planner targets are then immediately used for value and policy learning. For a fair comparison with BMPC (Wang et al., 2025), we use the same reanalysis schedule, performing reanalysis every 10 network updates and refreshing 20 replay transitions per event, with a network-update batch size of 256.

4.3 Theoretical Analysis

MPC-Based Value Backup Analysis. We first characterize the mismatch induced by using a policy-trained critic to bootstrap MPC. Consider the one-step continuation terms $\mathcal{T}_\pi Q = r + \gamma Q(s', a^\pi)$ and $\mathcal{T}_{\mathrm{MPC}} Q = r + \gamma Q(s', a^{\mathrm{MPC}})$. If $Q(s', \cdot)$ is L_Q-Lipschitz in action, then

$$|\mathcal{T}_{\mathrm{MPC}} Q - \mathcal{T}_\pi Q| \le \gamma L_Q \left\| a^{\mathrm{MPC}} - a^\pi \right\|.$$

Thus, the discrepancy between the value target used for critic learning and the continuation actually preferred by MPC grows directly with the MPC–policy action mismatch. Our MPC-based target removes this source of mismatch by evaluating the critic on MPC actions.

We next study the effect of the MPC-backup depth k. For the analysis, we fix the learned model and policy prior and consider an exact H-step MPC optimizer; finite-sample MPPI is an approximation

of this idealized operator. Let $\mathcal{J}_k(Q; x, \xi)$ denote the k-step continuation return evaluated along MPC action sequence ξ, where $x = (\bar{s}, z)$, and let $\Psi_H(Q; x)$ denote the solution of the H-step MPC problem. The induced MPC-based Bellman operator is

$$(\mathcal{T}_{k|H}Q)(x, a) = r(x, a) + \gamma\, \mathbb{E}_{x'} \left[\mathcal{J}_k \left(Q; x', \Phi_k \Psi_H(Q; x') \right) \right], \tag{10}$$

where Φ_k extracts the MPC actions required by the k-step backup, including its bootstrap action. The $k = H$ case recovers the standard multi-step Bellman contraction (Efroni et al., 2018; 2019), while we additionally characterize the truncated case $k < H$.

Theorem 1 (Contractivity of MPC-Based Value Backups)**.** *Assume $\gamma \in (0, 1)$, a fixed learned model and policy prior, and that the terminal value operator is non-expansive in Q under the supremum norm. If the full-horizon MPC exactly optimizes the same return used by the H-step target, then*

$$\left\| \mathcal{T}_{H|H}Q - \mathcal{T}_{H|H}Q' \right\|_\infty \leq \gamma^{H+1} \left\| Q - Q' \right\|_\infty. \tag{11}$$

For $k < H$, suppose the extracted MPC solution is $L_{\Psi,k}$-Lipschitz in Q, and $\mathcal{J}_k$ is $L_{J,k}$-Lipschitz in the MPC solution. Then

$$\left\| \mathcal{T}_{k|H}Q - \mathcal{T}_{k|H}Q' \right\|_\infty \leq \left(\gamma^{k+1} + \gamma L_{J,k} L_{\Psi,k} \right) \left\| Q - Q' \right\|_\infty. \tag{12}$$

Hence the truncated backup is contractive whenever $\gamma^{k+1} + \gamma L_{J,k} L_{\Psi,k} < 1$.

Proof. See Appendix A.1. □

The theorem reveals two regimes. When $k = H$, MPC trajectory selection and value evaluation are fully matched, recovering the standard γ^{H+1} contraction. For $k < H$, the truncated backup introduces an additional sensitivity term $\gamma L_{J,k} L_{\Psi,k}$. Although the full-horizon backup has the cleanest contraction guarantee, it relies on a longer learned-model rollout and is therefore more exposed to model error. In practice, we adopt $k = 0$, which provides the most stable empirical performance.

Bellman Target Error from Stale MPC Actions. We now specialize to the practical case $k = 0$. Let a_i^{cached} denote the MPC action stored in the replay buffer, and let a_i^{current} denote the action obtained by rerunning the current MPC at $\bar{s}_i'$. We define the Bellman-target error induced by the cached MPC action as

$$\mathcal{E}_i^{\text{BT}} = \left| \left[r_i + \gamma Q^- \left(\bar{s}_i', a_i^{\text{cached}} \right) \right] - \left[r_i + \gamma Q^- \left(\bar{s}_i', a_i^{\text{current}} \right) \right] \right|, \tag{13}$$

which measures the discrepancy between the target computed using the cached MPC action and the target obtained by replanning with the current MPC.

Theorem 2 (Bellman Target Error Bound under Stale MPC Actions)**.** *Assume that, at $\bar{s}_i'$, the ℓ_2 change in the MPC action is at most ρ_i per planner update, and that $Q^-(\bar{s}_i', \cdot)$ has an L_i-Lipschitz gradient with respect to the action. Then*

$$\mathcal{E}_i^{\text{BT}} \leq \gamma \left(\rho_i A_i g_i + \frac{L_i \rho_i^2}{2} A_i^2 \right). \tag{14}$$

Proof. See Appendix A.2. □

The bound shows that the Bellman-target error induced by a stale MPC action grows with both the target age A_i and the local action sensitivity g_i of the critic. Treating the unknown local constants ρ_i and L_i as approximately shared within a reanalysis batch, removing common positive factors, and absorbing the remaining relative weighting into β motivates the priority in Eq. equation 9.

5 Experiments

5.1 Experimental Setup

Baselines. We compare S-TD-MPC against four strong continuous-control baselines: (1) TD-MPC2 (Hansen et al., 2024) learns task-oriented latent dynamics and performs short-horizon MPC with learned reward and value models. (2) BMPC (Wang et al., 2025) trains the policy to imitate an MPC expert and uses the learned policy to improve subsequent planning. (3) BOOM (Zhan et al., 2026) further couples policy learning and online planning through explicit behavior alignment. (4) MRS.Q (Chang et al., 2026) combines a learned latent world model with short-horizon MPPI search and serves as a strong recent search-based baseline.

Benchmarks and Implementation. We evaluate S-TD-MPC on 7 challenging high-dimensional tasks from the DMC (DeepMind Control) (Tassa et al., 2018), covering the Humanoid and Dog domains. We use a single S-TD-MPC configuration across all tasks; detailed implementation settings, baseline sources, computational setup, and hyperparameters are provided in Appendix B.

5.2 Experimental Results

Overall performance. Figure 2 and Table 1 compare S-TD-MPC with the four baselines on seven challenging DMC Suite tasks. S-TD-MPC achieves the highest overall average return of 886.9, compared with 867.7 for BOOM, 864.2 for MRS.Q, 831.8 for BMPC, and 728.6 for TD-MPC2. It obtains the best final performance on six of the seven tasks, with Humanoid-Walk being the only exception. In particular, S-TD-MPC reaches 571.9 on Humanoid-Run and 845.3 on Dog-Run, the two most challenging tasks in our benchmark, while also maintaining near-saturated performance on the remaining Dog and Humanoid tasks. These results show that explicitly propagating the physical state, while retaining learned latent features for additional representation capacity, does not require sacrificing the strong control performance of latent TD-MPC methods.

Learning performance. The learning curves in Figure 2 further show that S-TD-MPC remains competitive throughout training rather than improving only at convergence. The average return rises rapidly and remains above the baselines for most of training. On Humanoid-Stand, Humanoid-Walk, Dog-Stand, Dog-Walk, and Dog-Trot, S-TD-MPC reaches high returns within roughly the first million environment steps, while on the more difficult Humanoid-Run and Dog-Run tasks it continues to improve steadily throughout training and attains the highest final return. Overall, these results indicate that maintaining explicit physical-state predictions preserves both final performance and sample efficiency on challenging high-dimensional continuous-control tasks.

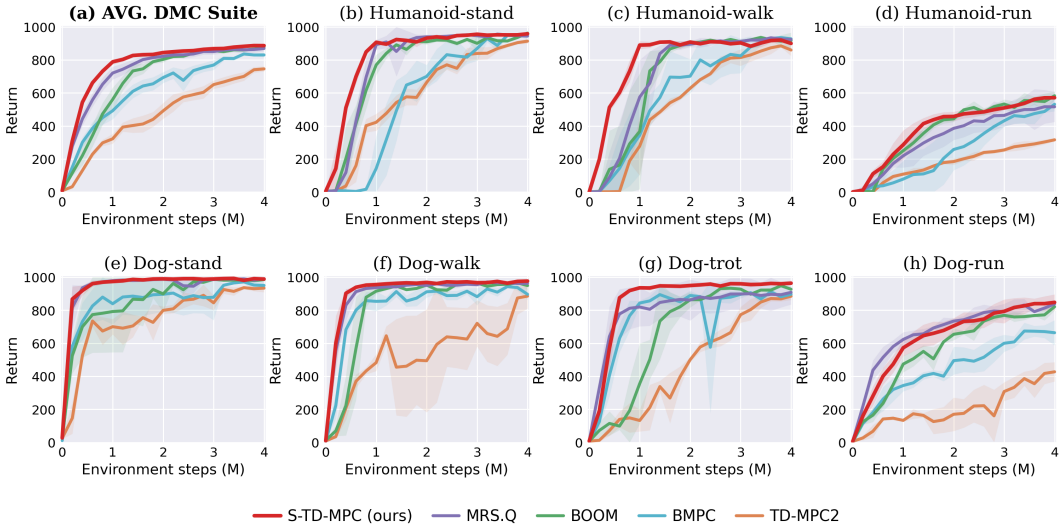


Figure 2: **Training curves.** Solid lines denote mean episodic returns across three seeds, while shaded regions show ±1 standard deviation. The AVG. DMC Suite curve averages the seven task-wise curves.

5.3 Ablation Study

We ablate two key training design choices of S-TD-MPC on Dog-Run and Humanoid-Run, and additionally compare S-TD-MPC against its latent-decoder variant on four Dog tasks.

Reanalysis priority. Figure 3a studies the sensitivity weight β in the reanalysis priority defined in Eq. equation 9. We compare our default setting $\beta = 10\text{K}$ against age-only prioritization ($\beta = 0$), larger sensitivity weights, and random reanalysis as used in BMPC (Wang et al., 2025), under the same reanalysis budget. In both tasks, incorporating critic action sensitivity improves over using target age alone or selecting samples randomly. A moderate weight of $\beta = 10\text{K}$ provides the

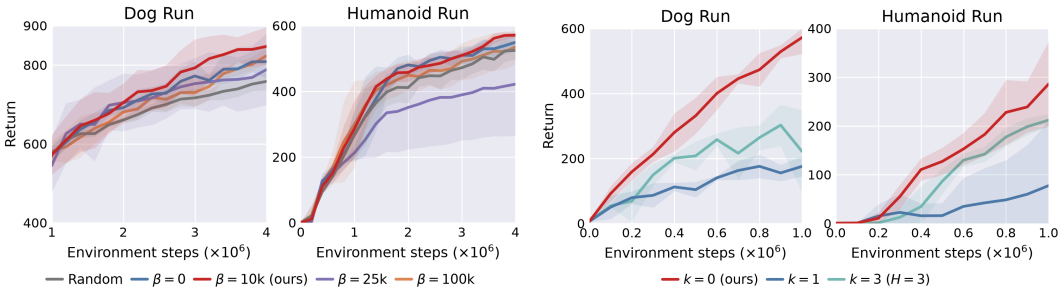
Table 1: Total average return on 7 challenging DMC Suite tasks at 4M environment steps. Each entry reports the mean ± std across three independent seeds, where the score for each seed is averaged over its final three evaluation points. Higher is better.

DMC Suite	H-Stand	H-Walk	H-Run	D-Stand	D-Walk	D-Trot	D-Run	Average
TD-MPC2	901.5 ± 11.8	872.5 ± 27.1	303.5 ± 13.2	933.3 ± 15.8	815.3 ± 110.3	874.5 ± 17.7	399.7 ± 69.0	728.6 ± 19.5
BMPC	946.0 ± 5.6	**930.7** ± 4.1	497.9 ± 37.0	957.7 ± 20.1	925.2 ± 20.3	896.2 ± 12.7	668.8 ± 49.0	831.8 ± 9.9
BOOM	937.4 ± 9.2	917.6 ± 28.0	561.0 ± 32.9	982.2 ± 6.0	956.9 ± 9.7	932.1 ± 9.3	786.6 ± 20.0	867.7 ± 7.2
MRS.Q	942.9 ± 16.9	928.9 ± 10.8	516.9 ± 96.5	986.9 ± 5.9	962.7 ± 7.9	904.5 ± 70.8	835.8 ± 23.5	864.2 ± 19.5
S-TD-MPC	**956.0** ± 5.5	911.7 ± 13.4	**571.9** ± 9.9	**988.1** ± 6.9	**974.5** ± 5.5	**961.0** ± 8.9	**845.3** ± 45.8	**886.9** ± 7.2

Task Abbreviation: H- denotes the DMC *Humanoid* domain (e.g., H-Stand is Humanoid-Stand); **D-** denotes the DMC *Dog* domain (e.g., D-Trot is Dog-Trot). **Bold** = best, Light Shade = second best.

most consistent performance, whereas excessively large values increasingly emphasize local critic sensitivity and can degrade learning. These results support using both target age and value sensitivity when allocating the limited reanalysis budget.

MPC backup depth. Figure 3b compares planner-backup depths $k \in \{0, 1, 3\}$, where $H = 3$. The one-step MPC-action target ($k = 0$) substantially outperforms the longer model-based backups on both Dog-Run and Humanoid-Run. While increasing k incorporates more rewards along the planned trajectory, it also relies on increasingly long recursive model predictions, making the target more susceptible to model error. This empirical result is consistent with the trade-off discussed in Section 4.3: although longer backups can provide a stronger contraction guarantee, the additional imagined rollout can introduce model error. We therefore use $k = 0$ in all main experiments.



(a) Ablation on reanalysis weight β.

(b) Ablation on MPC backup depth k.

Figure 3: **Ablation studies curves.** Solid lines show mean return over three seeds; shaded regions indicate ±1 std.

Latent-decoder variant comparison. We further compare S-TD-MPC with a latent-decoder variant that recursively propagates latent states and reconstructs the physical state through an auxiliary decoder. Relative to S-TD-MPC, the latent-decoder baseline reduces the average return across the four Dog tasks by 22.4% (943.8 ± 11.5 vs. 732.8 ± 43.2), with larger degradation on Dog-Trot (28.3%) and Dog-Run (57.6%). This degradation becomes more pronounced on dynamic tasks, supporting our design choice of directly propagating the physical state rather than reconstructing it from latent rollouts. Full training curves and implementation details are provided in Appendix C.1.

5.4 Downstream Capabilities

Explicit physical-state rollouts make quantities defined in physical coordinates directly accessible to the planner. We highlight zero-shot task transfer and deployment-time state constraints in the main text, while analytic reward evaluation is provided in Appendix C.4. Detailed setups and implementation details are provided in Appendix C.

Zero-shot task transfer. A central advantage of explicit state prediction is that the learned dynamics can be reused when the physical system remains unchanged but the task objective changes. We train S-TD-MPC on each of Dog-Run, Dog-Trot, Dog-Walk, and Dog-Stand, then freeze the

learned model and replace the planning reward with the analytic reward of a target task. Because the source-task critic is no longer valid after the objective changes, we remove the terminal Q bootstrap and increase the planning horizon from $H = 3$ to $H = 15$.

Table 2 shows all source–target combinations, with the off-diagonal entries corresponding to the 12 directed zero-shot transfers. S-TD-MPC transfers effectively across objectives, particularly when the target behavior is close to that encountered during source-task training. Moreover, transfer is notably asymmetric: a model trained on Dog-Run transfers well to Trot, Walk, and Stand, whereas Dog-Stand transfers poorly to the more dynamic locomotion tasks. This is consistent with running exposing the dynamics model to a broader range of joint configurations, velocities, contacts, and actions, while standing occupies a narrower region of the state-action space. Compared with a latent-decoder baseline introduced in Appendix C.1, S-TD-MPC achieves higher return on 10 of the 12 transfer directions, with an average off-diagonal return of 701.8 versus 533.9. Full baseline results, implementation details, and further analysis are provided in Appendix C.2.

Table 2: Zero-shot task transfer with S-TD-MPC.

	Target			
Source	Run	Trot	Walk	Stand
Run	891.5	911.2	676.2	942.2
Trot	471.2	972.2	910.2	982.4
Walk	324.9	626.8	985.2	989.2
Stand	274.1	432.5	880.8	999.0

Deployment-time state constraints. Explicit state predictions also allow physical constraints to be introduced directly into MPC without retraining the learned dynamics. We first impose spatial constraints on Dog-Trot by penalizing predicted trajectories that leave a prescribed corridor. Figure 4 shows that S-TD-MPC can follow both straight and sinusoidal feasible regions while maintaining strong locomotion performance: the constrained controller achieves 947.3 ± 44.8 in the straight corridor and 834.8 ± 76.7 in the sinusoidal corridor, compared with 965.9 ± 9.8 for unconstrained Dog-Trot. We further evaluate joint-space constraints on Dog-Run, where target configurations are imposed on either the back-left leg or the neck. These experiments show that constraint difficulty depends strongly on the task relevance of the constrained state: modifying the neck preserves substantially more running performance than constraining the locomotion-critical leg joints. Full state-constraint experiments, baseline comparisons, further analysis, and implementation details are provided in Appendix C.3.

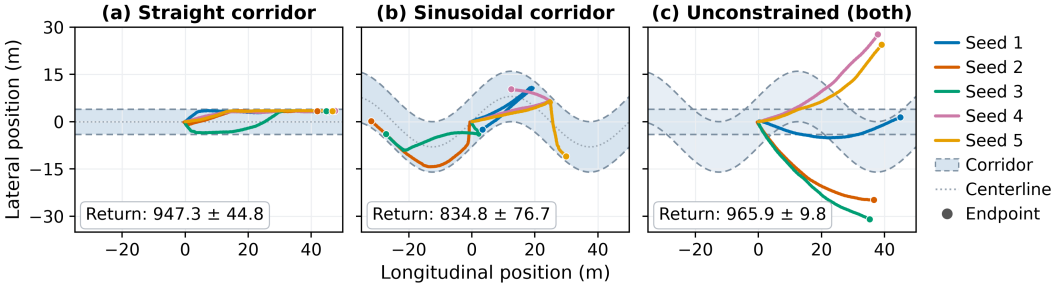


Figure 4: **Deployment-time spatial constraints on Dog-Trot.**

6 Conclusion

We introduced S-TD-MPC, a world model for continuous control that directly propagates the physical state while augmenting it with learned latent features. This preserves physical semantics in imagined trajectories without sacrificing the strong performance of modern TD-MPC methods. We further align value learning with planning through MPC-based Bellman targets and selective reanalysis. Across challenging DMC tasks, S-TD-MPC achieves strong control performance while enabling capabilities unavailable from purely latent rollouts: zero-shot transfer across reward-defined tasks, deployment-time state constraints, and direct use of known analytic rewards. These results show that explicit state prediction can provide a practical interface between learned dynamics and downstream control objectives, combining the flexibility of state-space model-based control with the representation power of modern world models.

AI USE STATEMENT

We used generative AI tools to polish the language of the manuscript and to assist with code generation and implementation; all AI-assisted code was manually reviewed and tested by the authors. We also used AI tools for literature retrieval and discovery, including searching for related work and checking whether closely related ideas or implementations had previously been reported. In addition, AI tools were used to inspect mathematical derivations and proofs for potential gaps or errors, with all final arguments independently verified by the authors. We take responsibility for the final content of this work, including all text, code, mathematical claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed proofs of the theoretical results in Appendix A, and report implementation details, computational setup, baseline sources, and all hyperparameters in Appendix B. Appendix C provides the complete experimental setups and implementation details for the downstream evaluations. Random seeds and evaluation protocols are reported for the corresponding experiments. Code, videos, and supplementary materials are available in the anonymous repository.

REFERENCES

Mehran Aghabozorgi, Alireza Moazeni, Yanshu Zhang, and Ke Li. Wimle: Uncertainty-aware world models with imle for sample-efficient continuous control. *arXiv preprint arXiv:2602.14351*, 2026.

Olov Andersson, Fredrik Heintz, and Patrick Doherty. Model-based reinforcement learning in continuous environments using real-time constrained optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International conference on machine learning*, pp. 501–510. PMLR, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. *arXiv preprint arXiv:2601.21306*, 2026.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one-step greedy approach in reinforcement learning. In *International Conference on Machine Learning*, pp. 1387–1396. PMLR, 2018.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. How to combine tree-search methods in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3494–3501, 2019.

Thomas Evers, Cristian Meo, Wendelin Bohmer, Justin Dauwels, and Yaniv Oren. Efficienttdmpc: Improved mpc objectives for sample-efficient continuous control. *arXiv preprint arXiv:2605.16692*, 2026.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nick Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *International Conference on Learning Representations*, volume 2026, pp. 5048–5091, 2026.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Mintae Kim and Koushil Sreenath. Wombet: World model-based experience transfer for robust and sample-efficient reinforcement learning. *arXiv preprint arXiv:2604.08958*, 2026.

Romain Laroche and Merwan Barlier. Transfer reinforcement learning with shared dynamics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in neural information processing systems*, 33:741–752, 2020.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. Td-m (pc)ˆ2: Improving temporal difference mpc through policy constraint. *arXiv preprint arXiv:2502.03550*, 2025.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

Jonas Matthies, Muhammad Burhan Hafez, Mostafa Kotb, and Stefan Wermter. Model predictive control with self-supervised representation learning. *arXiv preprint arXiv:2304.07219*, 2023.

David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Anh N Nhu, Sanghyun Son, and Ming Lin. Time-aware world model for adaptive prediction and control. *arXiv preprint arXiv:2506.08441*, 2025.

Aidan Scannell, Mohammadreza Nakhaeinezhadfard, Kalle Kujanpää, Yi Zhao, Kevin Luck, Arno Solin, and Joni Pajarinen. Discrete codebook world models for continuous control. In *International Conference on Learning Representations*, volume 2025, pp. 11111–11148, 2025.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

Jonathan Spieler and Sven Behnke. Dream-mpc: Gradient-based model predictive control with latent imagination. *arXiv preprint arXiv:2605.04568*, 2026.

Isidoro Tamassia, Lennert De Smet, and Giuseppe Marra. Towards zero-shot task transfer with neurosymbolic world models. *arXiv preprint arXiv:2608.17959*, 2026.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *International conference on learning representations*, volume 2025, pp. 54241–54259, 2025.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017.

Sizhe Yang, Yanjie Ze, and Huazhe Xu. Movie: Visual model-based policy adaptation for view generalization. *Advances in Neural Information Processing Systems*, 36:21507–21523, 2023.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *Advances in Neural Information Processing Systems*, 38:134093–134121, 2026.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018.

Yi Zhao, Wenshuai Zhao, Rinu Boney, Juho Kannala, and Joni Pajarinen. Simplified temporal consistency reinforcement learning. In *International Conference on Machine Learning*, pp. 42227–42246. PMLR, 2023.

A Theoretical Analysis

A.1 Proof of Theorem 1

Lemma 1 (Fixed-sequence sensitivity)**.** *For any fixed MPC action sequence ξ,*

$$|\mathcal{J}_k(Q; x, \xi) - \mathcal{J}_k(Q'; x, \xi)| \leq \gamma^k \|Q - Q'\|_\infty . \tag{15}$$

Proof. For a fixed MPC action sequence, the model rollout and predicted rewards are identical under Q and Q', and only the terminal bootstrap depends on the critic. By the assumed non-expansiveness of the terminal value operator, its difference is bounded by $\|Q - Q'\|_\infty$. Since the terminal value appears after k discounts, Eq. equation 15 follows. □

The $k = H$ case follows the standard multi-step lookahead contraction argument (Efroni et al., 2018; 2019). We reproduce the argument for completeness and to establish the basis for our $k < H$ extension, where the MPC selection horizon and value-backup horizon differ.

Proof of Theorem 1. For $k = H$, following the standard multi-step contraction argument, define the optimal H-step MPC value as

$$V_H(Q; x) = \sup_\xi \mathcal{J}_H(Q; x, \xi).$$

Because the MPC objective and the H-step target use the same return,

$$\begin{aligned} |V_H(Q; x) - V_H(Q'; x)| &\leq \sup_\xi |\mathcal{J}_H(Q; x, \xi) - \mathcal{J}_H(Q'; x, \xi)| \\ &\leq \gamma^H \|Q - Q'\|_\infty , \end{aligned}$$

where the first inequality follows from $|\sup f - \sup g| \leq \sup |f - g|$ and the second from Lemma 1. Applying the outer Bellman discount and taking the supremum over (x, a) gives

$$\left\|\mathcal{T}_{H|H}Q - \mathcal{T}_{H|H}Q'\right\|_\infty \leq \gamma^{H+1} \|Q - Q'\|_\infty .$$

For $k < H$, let

$$\xi = \Phi_k \Psi_H(Q; x), \qquad \xi' = \Phi_k \Psi_H(Q'; x).$$

The $L_{\Psi,k}$-Lipschitz assumption means that, for a metric d_k on the extracted MPC solutions,

$$d_k(\xi, \xi') \leq L_{\Psi,k} \|Q - Q'\|_\infty ,$$

while the $L_{J,k}$-Lipschitz assumption gives

$$|\mathcal{J}_k(Q; x, \xi) - \mathcal{J}_k(Q; x, \xi')| \leq L_{J,k} d_k(\xi, \xi').$$

Adding and subtracting $\mathcal{J}_k(Q'; x, \xi)$ yields

$$\begin{aligned} &|\mathcal{J}_k(Q; x, \xi) - \mathcal{J}_k(Q'; x, \xi')| \\ &\leq |\mathcal{J}_k(Q; x, \xi) - \mathcal{J}_k(Q'; x, \xi)| + |\mathcal{J}_k(Q'; x, \xi) - \mathcal{J}_k(Q'; x, \xi')| \\ &\leq \left(\gamma^k + L_{J,k} L_{\Psi,k}\right) \|Q - Q'\|_\infty , \end{aligned}$$

where the first term follows from Lemma 1 and the second from the two Lipschitz assumptions above. Applying the outer discount γ and taking the supremum gives

$$\left\|\mathcal{T}_{k|H}Q - \mathcal{T}_{k|H}Q'\right\|_\infty \leq \left(\gamma^{k+1} + \gamma L_{J,k} L_{\Psi,k}\right) \|Q - Q'\|_\infty .$$

Hence $\mathcal{T}_{k|H}$ is a contraction whenever $\gamma^{k+1} + \gamma L_{J,k} L_{\Psi,k} < 1$. □

A.2 Proof of Theorem 2

Proof. For the fixed replay state $\bar{s}'_i$, let $a_i^{(0)}, \ldots, a_i^{(A_i)}$ denote the MPC actions across the A_i planner updates between the cached and current versions, such that

$$a_i^{(0)} = a_i^{\text{cached}}, \qquad a_i^{(A_i)} = a_i^{\text{current}}.$$

The bounded-drift assumption means that

$$\left\| a_i^{(m+1)} - a_i^{(m)} \right\|_2 \le \rho_i, \qquad m = 0, \ldots, A_i - 1.$$

Therefore, by the triangle inequality,

$$\left\| a_i^{\text{current}} - a_i^{\text{cached}} \right\|_2 \le \rho_i A_i. \tag{16}$$

For notational convenience, define $q_i(a) = Q^-(\bar{s}'_i, a)$. The L_i-Lipschitz gradient assumption means

$$\|\nabla_a q_i(a) - \nabla_a q_i(a')\|_2 \le L_i \|a - a'\|_2 .$$

Let $\Delta a_i = a_i^{\text{current}} - a_i^{\text{cached}}$. By the standard smoothness inequality,

$$\begin{aligned} \left| q_i(a_i^{\text{current}}) - q_i(a_i^{\text{cached}}) \right| &\le \left| \nabla_a q_i(a_i^{\text{cached}})^\top \Delta a_i \right| + \frac{L_i}{2} \|\Delta a_i\|_2^2 \\ &\le g_i \|\Delta a_i\|_2 + \frac{L_i}{2} \|\Delta a_i\|_2^2 , \end{aligned}$$

where the second inequality follows from Cauchy–Schwarz and the definition of g_i. Using Eq. equation 16 gives

$$\left| q_i(a_i^{\text{current}}) - q_i(a_i^{\text{cached}}) \right| \le \rho_i A_i g_i + \frac{L_i \rho_i^2}{2} A_i^2.$$

Finally, from the definition of $\mathcal{E}_i^{\text{BT}}$, the reward terms cancel, yielding

$$\mathcal{E}_i^{\text{BT}} = \gamma \left| q_i(a_i^{\text{cached}}) - q_i(a_i^{\text{current}}) \right| .$$

Combining the above inequalities gives

$$\mathcal{E}_i^{\text{BT}} \le \gamma \left(\rho_i A_i g_i + \frac{L_i \rho_i^2}{2} A_i^2 \right),$$

which proves Eq. equation 14. □

From the bound to the practical reanalysis score. Consider a reanalysis batch $\mathcal{B}$ and define the batch-wise bounds

$$\rho = \max_{i \in \mathcal{B}} \rho_i, \qquad L = \max_{i \in \mathcal{B}} L_i.$$

Since $\rho_i \le \rho$ and $L_i \le L$, Theorem 2 implies

$$\mathcal{E}_i^{\text{BT}} \le \gamma \left(\rho A_i g_i + \frac{L \rho^2}{2} A_i^2 \right), \qquad i \in \mathcal{B}.$$

For ranking samples within the same batch, the common positive factor $\gamma L \rho^2 / 2$ does not affect their ordering. Dividing by this factor gives

$$A_i^2 + \frac{2}{L\rho} A_i g_i.$$

The quantities ρ and L can in principle be estimated from repeated MPC solutions and local critic curvature. For example, ρ may be approximated from successive MPC actions as $\hat{\rho} \approx \max \|a^{(m+1)} - a^{(m)}\|_2$, while L can be estimated locally from critic gradients using $\hat{L} \approx \max \|\nabla_a Q(a + \delta a) - \nabla_a Q(a)\|_2 / \|\delta a\|_2$, or from the spectral norm of the action Hessian when second-order information is available. However, these estimates require additional MPC evaluations or gradient/curvature computation. We therefore avoid estimating them explicitly and absorb their relative weighting into a single tunable coefficient β, yielding

$$S_i = A_i^2 + \beta A_i g_i,$$

which is the priority used in Eq. equation 9.

A.3 Bellman Target Error Bound for Multi-Step Backups

For $k > 0$, let $\mathbf{a}_{i,k} = (a_{i,0}, \ldots, a_{i,k})$ denote the action sequence used by the backup. Starting from $x_{i,0} = (\bar{s}'_i, h(\tilde{s}'_i))$, where $\bar{s}'_i$ is the normalized next state, let $x_{i,j+1} = F(x_{i,j}, a_{i,j})$, with F denoting one complete world-model transition. Define

$$G_{i,k}(\mathbf{a}_{i,k}) = \sum_{j=0}^{k-1} \gamma^j R(x_{i,j}, a_{i,j}) + \gamma^k Q^-(x_{i,k}, a_{i,k}), \tag{17}$$

so that $y_{i,k} = r_i + \gamma G_{i,k}(\mathbf{a}_{i,k})$. Let $\mathbf{a}_{i,k}^{\text{cached}}$ and $\mathbf{a}_{i,k}^{\text{current}}$ denote the cached and current sequences, and define

$$\mathcal{E}_{i,k}^{\text{BT}} = \gamma \left| G_{i,k}(\mathbf{a}_{i,k}^{\text{cached}}) - G_{i,k}(\mathbf{a}_{i,k}^{\text{current}}) \right|.$$

Let ν_i denote the MPC version at which the cached target was generated. For each action position j, assume $\|a_{i,j}^{(m+1)} - a_{i,j}^{(m)}\|_2 \le \rho_{i,j}$ for $m = \nu_i, \ldots, \nu_i + A_i - 1$. Since $a_{i,j}^{(\nu_i)} = a_{i,j}^{\text{cached}}$ and $a_{i,j}^{(\nu_i + A_i)} = a_{i,j}^{\text{current}}$, the triangle inequality gives

$$\|a_{i,j}^{\text{current}} - a_{i,j}^{\text{cached}}\|_2 \le \rho_{i,j} A_i. \tag{18}$$

Define the local sensitivity

$$g_{i,j}^{(k)} = \left\| \nabla_{a_j} G_{i,k}(\mathbf{a}) \big|_{\mathbf{a}=\mathbf{a}_{i,k}^{\text{cached}}} \right\|_2. \tag{19}$$

Theorem 3 (Bellman Target Error Bound for $k > 0$)**.** *Suppose the block gradients satisfy*

$$\left\| \nabla_{a_j} G_{i,k}(\mathbf{a}) - \nabla_{a_j} G_{i,k}(\mathbf{b}) \right\|_2 \le \sum_{\ell=0}^{k} L_{i,j\ell}^{(k)} \|a_\ell - b_\ell\|_2.$$

Then

$$\mathcal{E}_{i,k}^{\text{BT}} \le \gamma \left[A_i \sum_{j=0}^{k} \rho_{i,j} g_{i,j}^{(k)} + \frac{A_i^2}{2} \sum_{j=0}^{k} \sum_{\ell=0}^{k} L_{i,j\ell}^{(k)} \rho_{i,j} \rho_{i,\ell} \right]. \tag{20}$$

Proof. Let $\Delta a_{i,j} = a_{i,j}^{\text{current}} - a_{i,j}^{\text{cached}}$ and $\Delta \mathbf{a}_i = (\Delta a_{i,0}, \ldots, \Delta a_{i,k})$. Define $\mathbf{a}(t) = \mathbf{a}_{i,k}^{\text{cached}} + t\Delta \mathbf{a}_i$ and $\phi(t) = G_{i,k}(\mathbf{a}(t))$. By the chain rule and the fundamental theorem of calculus,

$$G_{i,k}(\mathbf{a}_{i,k}^{\text{current}}) - G_{i,k}(\mathbf{a}_{i,k}^{\text{cached}}) = \int_0^1 \sum_{j=0}^{k} \nabla_{a_j} G_{i,k}(\mathbf{a}(t))^\top \Delta a_{i,j} \, dt.$$

Adding and subtracting the gradient at the cached sequence, applying Cauchy–Schwarz, and using the block-gradient assumption gives

$$\left| G_{i,k}(\mathbf{a}_{i,k}^{\text{current}}) - G_{i,k}(\mathbf{a}_{i,k}^{\text{cached}}) \right| \le \sum_{j=0}^{k} g_{i,j}^{(k)} \|\Delta a_{i,j}\|_2 + \int_0^1 t \sum_{j=0}^{k} \sum_{\ell=0}^{k} L_{i,j\ell}^{(k)} \|\Delta a_{i,j}\|_2 \|\Delta a_{i,\ell}\|_2 \, dt.$$

Since $\int_0^1 t \, dt = 1/2$ and $\|\Delta a_{i,j}\|_2 \le \rho_{i,j} A_i$ from Eq. equation 18,

$$\left| G_{i,k}(\mathbf{a}_{i,k}^{\text{current}}) - G_{i,k}(\mathbf{a}_{i,k}^{\text{cached}}) \right| \le A_i \sum_{j=0}^{k} \rho_{i,j} g_{i,j}^{(k)} + \frac{A_i^2}{2} \sum_{j=0}^{k} \sum_{\ell=0}^{k} L_{i,j\ell}^{(k)} \rho_{i,j} \rho_{i,\ell}.$$

Multiplying by the outer Bellman discount γ proves Eq. equation 20. □

When $k = 0$, $G_{i,0}(a) = Q^-(x_{i,0}, a)$ and the sums contain only $j = \ell = 0$. With $\rho_{i,0} = \rho_i$, $g_{i,0}^{(0)} = g_i$, and $L_{i,00}^{(0)} = L_i$, the bound reduces exactly to

$$\mathcal{E}_i^{\text{BT}} \le \gamma \left(\rho_i A_i g_i + \frac{L_i \rho_i^2}{2} A_i^2 \right),$$

recovering Theorem 2. For $k > 0$, the bound requires sensitivities for all actions and cross-step smoothness terms, which require differentiation through the multi-step world-model rollout. We therefore use the $k = 0$ bound for practical reanalysis.

B Implementation Details

This section provides the implementation details for all experiments reported in Section 5.2, including S-TD-MPC and all baseline methods. S-TD-MPC is implemented based on the official TD-MPC2 codebase[1] and retains the default TD-MPC2 hyperparameters unless otherwise specified. The same configuration is used across all previously demonstrated tasks. The complete hyperparameter settings of S-TD-MPC are summarized in Table 4. All experiments are conducted on a single NVIDIA A100-SXM-64GB GPU with an Intel Xeon Platinum 8358 CPU @ 2.60 GHz. Taking Dog-Run as a representative example, training S-TD-MPC for 4M environment steps requires approximately 20 hours.

For MRS.Q, we rerun the official implementation[2] and extend training from the originally reported 1M environment steps to 4M environment steps, using exactly the released hyperparameters without any modification. For BOOM, BMPC, and TD-MPC2, we report the baseline results released by BOOM, which are obtained using the official repositories of BOOM[3], BMPC[4], and TD-MPC2[1], respectively.

Table 3 further compares the model size and training time of S-TD-MPC and MRS.Q on Dog-Run and Humanoid-Run. Both methods are trained for 4M environment steps, and the reported training time is averaged over three seeds.

Table 3: Model size and wall-clock training time on Dog-Run and Humanoid-Run. Training time is averaged over three runs of 4M environment steps each.

	Dog-Run		Humanoid-Run	
Method	Params (M)	Time (h)	Params (M)	Time (h)
S-TD-MPC	**5.92**	**19.9**	**5.09**	**16.2**
MRS.Q	10.30	31.9	10.20	31.6

[1] https://github.com/nicklashansen/tdmpc2
[2] https://github.com/facebookresearch/MRSQ
[3] https://github.com/molumitu/BOOM_MBRL
[4] https://github.com/wertyuilife2/bmpc

Table 4: Hyperparameter settings of S-TD-MPC.

Hyperparameter	Value	Hyperparameter	Value
Training			
Learning rate	3×10^{-4}	Replay batch size	256
Replay buffer size	1,000,000	Environment steps	4,000,000
Optimizer	Adam	Gradient clipping norm	20
Target network update rate τ	0.01	Temporal weighting ρ	0.5
Evaluation frequency	50,000	Evaluation episodes	10
Random seeds	3		
World Model			
Reward loss coefficient λ_r	0.1	Value loss coefficient λ_Q	0.1
State loss coefficient λ_s	0.25	State loss temporal weight ρ_s	0.5
State input clip c	100	Residual bound $\delta_{\max}$	100
Number of value bins	101		
Planner			
Planning horizon H	3	MPPI iterations	6 (8 if $\lvert\mathcal{A}\rvert \geq 20$)
Population size	512	Number of elites	64
Policy-prior samples	24	Temperature	0.5
Minimum planner std	0.05	Maximum planner std	2
MPC backup depth k	0	Target-action noise η	1.0
Policy Distillation			
KL coefficient λ_{KL}	1.0	Entropy coefficient $\lambda_{\mathcal{H}}$	10^{-4}
Reanalysis			
Reanalysis interval	10	Reanalysis batch size	20
Sensitivity weight β	10^4		
Architecture			
Encoder layers	2	Feature dimension	512
Encoder hidden dimension	256	MLP hidden dimension	512
Dynamics layers	2	Q-function dropout rate	0.01
MLP activation	Mish	MLP normalization	LayerNorm
SimNorm dimension	8	Number of Q-functions	5

C Downstream Task Details

C.1 Latent-Decoder Baseline

To provide a direct reconstruction-based baseline for the downstream experiments, we construct a latent-decoder variant of S-TD-MPC. The purpose of this baseline is twofold: it provides physical-state predictions that can be used by the same downstream objectives and constraints as S-TD-MPC, while also representing the alternative strategy of recovering physical semantics from a latent world model through an auxiliary decoder.

Baseline design and comparison. The baseline retains the same planner-consistent training framework, MPC-based value learning, policy training, and reanalysis procedure as S-TD-MPC, but replaces explicit state propagation with a TD-MPC2-style recursive latent dynamics model. An additional decoder maps each imagined latent state back to the physical state space as shown in Fig. 5. Thus, the key difference is how state-space predictions are obtained: S-TD-MPC uses the physical state itself as the rollout variable, whereas the baseline first propagates the latent state and subsequently reconstructs the corresponding physical state.

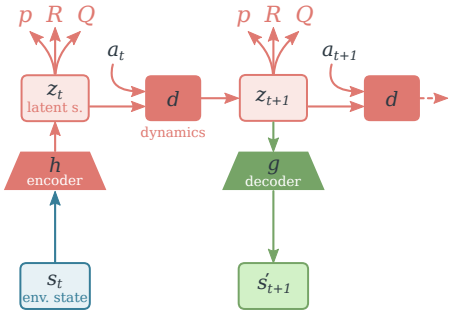


Figure 5: **Latent-decoder baseline.**

Figure 6 compares the standard training performance of S-TD-MPC and the latent-decoder baseline on the four Dog tasks. The latent-decoder baseline remains competitive on Dog-Stand and performs reasonably well on Dog-Walk, but its performance degrades substantially on the more challenging Dog-Trot and Dog-Run tasks. In contrast, S-TD-MPC maintains strong performance across all four tasks. This trend echoes the observations summarized in the Introduction: reconstruction-based approaches can remain effective on relatively simple control problems, but their performance can deteriorate substantially in more challenging high-dimensional control settings (Matthies et al., 2023; Hansen et al., 2022; Zhao et al., 2023; Scannell et al., 2025). Here, the degradation becomes particularly pronounced as the required locomotion behavior becomes more dynamic. These results further motivate directly propagating the physical state rather than recovering it afterwards through an auxiliary decoder. We nevertheless retain the latent-decoder model as the main comparison in the downstream experiments, since both approaches expose physical-state predictions to the planner and therefore support the same downstream objectives.

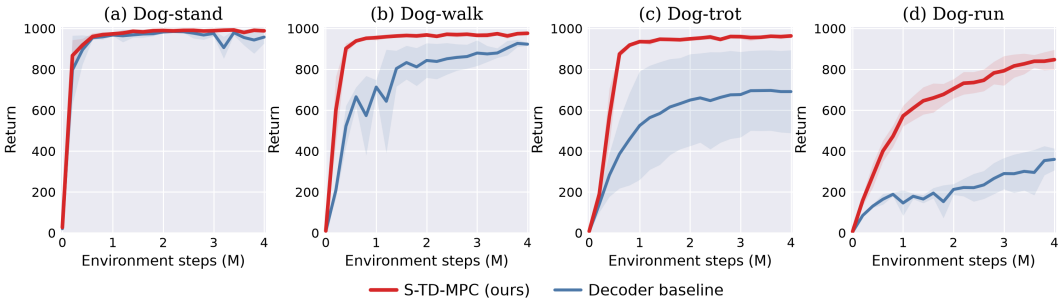


Figure 6: **Training curves for S-TD-MPC and the latent-decoder baseline on the four standard Dog tasks.** The baseline uses recursive latent dynamics and reconstructs physical states with an auxiliary decoder, whereas S-TD-MPC directly propagates the physical state. Solid lines denote mean episodic return over three seeds, and shaded regions show ± 1 standard deviation.

Implementation details. The latent-decoder baseline is implemented within the same training and planning framework as S-TD-MPC. The explicit state dynamics of S-TD-MPC are replaced by TD-MPC2-style recursive latent dynamics, while an additional MLP decoder maps each imagined latent state to the normalized physical state. The decoder consists of two hidden layers with the same

hidden dimension as the other prediction heads and is supervised on recursively imagined latent states produced by the learned dynamics, rather than only on latent representations encoded directly from ground-truth states. All planner, MPC-based value learning, policy distillation, reanalysis, and optimization settings follow S-TD-MPC unless otherwise specified. The latent-decoder-specific hyperparameters are summarized in Table 5.

Table 5: Latent-decoder-specific hyperparameters. All remaining settings follow the S-TD-MPC configuration in Table 4.

Hyperparameter	**Value**
Latent consistency coefficient	20
Decoder state-loss coefficient	0.25
Decoder temporal weight ρ_{dec}	0.5
State-normalization freeze updates	10,000
State-normalization clipping threshold	100
State-normalization epsilon	10^{-5}
State-normalization minimum std	10^{-3}

C.2 Zero-Shot Task Transfer

A central advantage of model-based control is the separation between system dynamics and task objectives: once a sufficiently accurate dynamics model is available, MPC can in principle solve a new task by replacing the cost function and replanning, without relearning the dynamics. This modularity is largely lost in latent TD-MPC methods, where imagined trajectories have no direct physical-state semantics and the learned reward and value functions are tied to the training objective. Consequently, even when two tasks share exactly the same physical system, changing the objective typically requires additional learning. Explicit state prediction restores this interface between the learned dynamics and the task objective. We therefore study whether a model trained for one Dog locomotion objective can be directly reused for another by replacing the planning reward, without any target-task training or parameter updates.

Transfer setup and results. We consider Dog-Run, Dog-Trot, Dog-Walk, and Dog-Stand, which share the same physical dynamics but differ in their reward functions. For each source–target pair, we freeze the source-task model and perform no additional training on the target task. Because the learned critic Q is tied to the source-task objective and is therefore no longer valid after the reward changes, we remove the terminal Q bootstrap during transfer. MPC instead evaluates the analytic reward of the target task directly along the predicted physical-state trajectory. To compensate for the absence of terminal value prediction, we increase the planning horizon from $H = 3$ during training to $H = 15$ during transfer. We evaluate all $4 \times 3 = 12$ directed source–target transfers. As a comparison, we use the latent-decoder baseline introduced above, which reconstructs physical states from imagined latent trajectories and therefore permits the same analytic target reward to be evaluated during planning.

Table 6 shows that S-TD-MPC achieves higher return on 10 of the 12 transfer directions, with an average off-diagonal return of 701.8 compared with 533.9 for the latent-decoder baseline. The results also reveal a pronounced asymmetry across source tasks. Models trained on more dynamic behaviors generally transfer more successfully to slower objectives than the reverse; for example, models trained on Dog-Run transfer substantially better to Trot, Walk, and Stand than a Dog-Stand model transfers to locomotion objectives. This asymmetry is consistent with the state–action coverage induced by the source task: running exposes the learned dynamics to a broader range of joint configurations, velocities, contacts, and actions, whereas standing occupies a much narrower region of the state–action space. Consequently, transfer from a narrow source distribution to a more dynamic target requires planning increasingly far outside the data distribution on which the dynamics were learned. The strong Run-to-Walk result of the latent-decoder baseline is largely incidental: because its Dog-Run model fails to learn fast running and instead produces much slower locomotion, its behavior happens to match the Dog-Walk objective particularly well. Overall, the results demonstrate that the learned dynamics can support zero-shot reuse across new objectives, especially when the target behavior is not too far from the source task. Performance degrades as the behav-

ioral gap increases, showing that shared dynamics alone are not sufficient for reliable transfer: the source-task experience must also provide adequate coverage of the state–action regions required by the new objective. This further highlights the importance of broad exploration during training, since richer behavioral coverage directly expands the set of downstream objectives that a learned dynamics model can support without retraining.

Table 6: **Cross-task transfer results on Dog locomotion tasks.** Source denotes the training task, and Target denotes the evaluation task. Shaded cells report performance on the original task; unshaded cells report transfer performance. Bold indicates the higher mean return between the two methods for each transfer pair.

		Target			
Source	Method	Run	Trot	Walk	Stand
Run	S-TD-MPC	891.51 ± 10.06	**911.21** ± 10.61	676.19 ± 19.63	942.18 ± 1.09
	Baseline	322.68 ± 4.61	644.53 ± 2.89	**922.49** ± 15.73	**947.62** ± 5.74
Trot	S-TD-MPC	**471.21** ± 4.05	972.19 ± 4.56	**910.23** ± 5.71	**982.35** ± 5.39
	Baseline	455.76 ± 14.77	931.42 ± 11.17	524.50 ± 75.62	912.54 ± 32.28
Walk	S-TD-MPC	**324.90** ± 1.86	**626.83** ± 6.61	985.16 ± 6.95	**989.24** ± 0.59
	Baseline	275.91 ± 22.73	410.15 ± 38.80	941.76 ± 39.91	884.77 ± 93.13
Stand	S-TD-MPC	**274.11** ± 4.90	**432.53** ± 42.67	**880.76** ± 31.07	998.96 ± 0.41
	Baseline	198.87 ± 5.31	146.60 ± 8.37	82.77 ± 27.28	980.19 ± 1.50

Implementation details. We use a single transfer configuration for all source–target pairs and for both S-TD-MPC and the latent-decoder baseline. During transfer, MPC uses a planning horizon of 15, 6 MPPI iterations, 256 sampled trajectories, 32 elites, and 24 policy-prior trajectories, with temperature 0.5, maximum action-distribution standard deviation 2.0, and action scale 1.0. The terminal critic is disabled and candidate trajectories are ranked solely by the accumulated analytic reward of the target task. We evaluate each transfer over three episodes. All model parameters remain frozen throughout evaluation.

C.3 Deployment-Time State Constraints

Constraint handling is another central advantage of MPC and is particularly important in robotics, where controllers may need to avoid unsafe regions, respect workspace boundaries, or maintain the robot within prescribed physical configurations. Such constraints are naturally defined in the physical state space and can be modified at deployment time without retraining the underlying dynamics model. This capability is difficult to exploit with purely latent rollouts whose coordinates have no direct physical meaning. We evaluate whether the explicit state predictions of S-TD-MPC can be used directly for deployment-time constraints without any additional learning. We consider two complementary settings: spatial constraints on the robot trajectory and joint-pose constraints on the robot itself.

C.3.1 Spatial Constraints

Setup and results. We impose deployment-time spatial constraints on the predicted Dog-Trot trajectory without modifying or retraining the learned model. We consider two corridor geometries. The straight corridor is an infinite strip with a half-width of 4 m, while the sinusoidal corridor has a half-width of 8 m around a sinusoidal centerline with amplitude 8 m, wavelength 50 m, and zero phase. During MPC, candidate trajectories are evaluated using the original locomotion objective together with penalties defined on their predicted spatial positions, with increasingly large penalties applied near and beyond the corridor boundary. We disable the terminal critic because it is unconstrained and does not account for the deployment-time spatial objective. To compensate for the resulting loss of terminal value guidance, we extend the planning horizon from the training horizon $H = 3$ to $H = 15$ for the straight corridor and $H = 25$ for the more demanding sinusoidal corridor.

Figure 7 shows that S-TD-MPC can reshape its behavior at deployment time to satisfy both spatial constraints. In the straight corridor, the constrained controller keeps all trajectories within the narrow strip while retaining a return of 947.3 ± 44.8, close to the unconstrained return of 965.9 ± 9.8. The sinusoidal corridor requires a substantially larger deviation from the nominal Dog-Trot behavior, yet the constrained controller follows the curved feasible region across seeds and achieves 834.8 ± 76.7. In contrast, the unconstrained controller frequently leaves both corridors despite achieving high task return, confirming that the observed behavior is induced by the deployment-time constraint rather than by the original policy. The latent-decoder baseline is considerably less reliable: although reconstructed states allow the same constraints to be evaluated, it achieves only 237.0 ± 124.1 in the straight corridor and 548.4 ± 427.1 in the sinusoidal corridor, with substantially less consistent forward progress. These results show that explicit access to physical-state predictions is useful not only for specifying constraints, but also for enforcing them while preserving the original control objective.

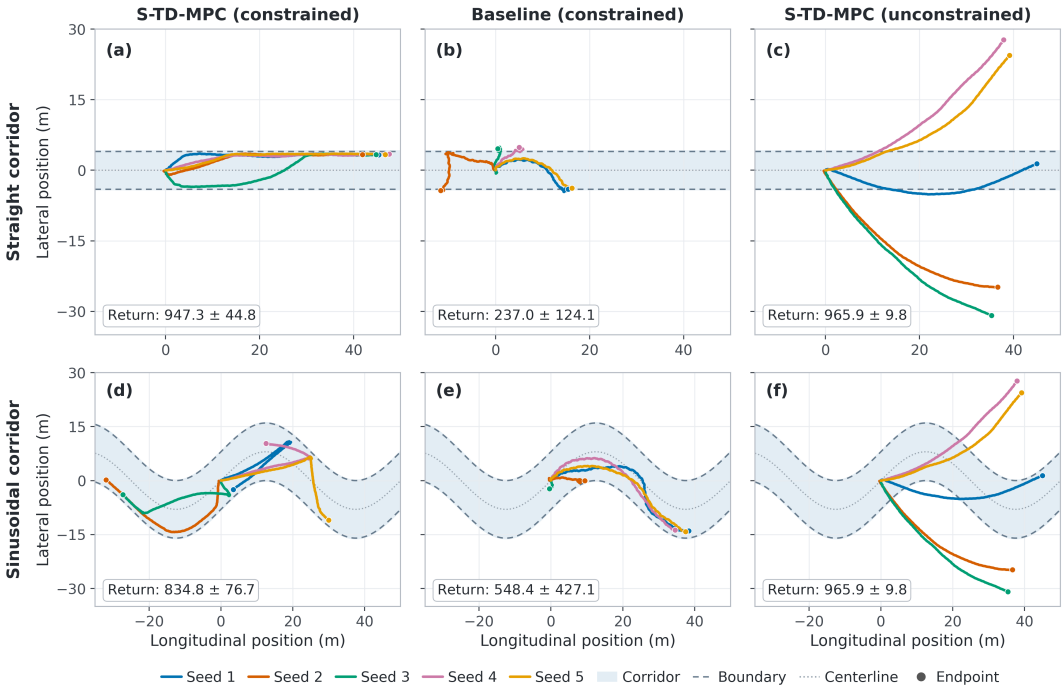


Figure 7: **Deployment-time spatial constraints on Dog-Trot.** Trajectories from five evaluation seeds are shown for a straight corridor (top) and a sinusoidal corridor (bottom). S-TD-MPC directly evaluates the corridor constraints on predicted physical states during planning. The latent-decoder baseline applies the same constraints to reconstructed states, while the unconstrained S-TD-MPC controller uses the original objective only. Shaded regions denote the feasible corridors, dashed lines their boundaries, and dotted lines their centerlines.

Implementation details. The original DMC Dog observation does not contain the global planar position of the robot. To introduce spatial constraints without changing the observation space or retraining the world model, we obtain the current global (x, y) position directly from the MuJoCo simulator at each real environment step and use it only as an anchor for planning. During each imagined MPC rollout, future planar positions are reconstructed by integrating the torso-velocity components predicted by the world model from this current position. Corridor penalties are then evaluated on these reconstructed positions at every imagined step. Thus, the spatial information required by the constraint is introduced entirely at deployment time: the learned model remains unchanged and predicts the same physical state variables as during training, while global position is recovered deterministically from its predicted velocities. The same trained Dog-Trot checkpoint is used for all spatial-constraint experiments, with no parameter updates during evaluation. Table 7 summarizes the corridor geometry and MPC settings.

Table 7: Deployment-time spatial-constraint settings on Dog-Trot.

Parameter	Straight	Sinusoidal
Corridor half-width (m)	4	8
Centerline amplitude (m)	–	8
Centerline wavelength (m)	–	50
Centerline phase	–	0
Planning horizon	15	25
Samples	512	512
Centerline penalty	0	0.03
Lateral-velocity penalty	2.0	0
Action-prior weight	2.0	0.3
Boundary margin (m)	0.5	2.0
Boundary penalty	1000	10
Outside quadratic penalty	20,000	100,000
Outside fixed penalty	100	10,000
Terminal Q	No	No

C.3.2 Joint-Space Constraints

Joint-space constraints provide a complementary test of deployment-time state control. Such constraints are important in robotics for safety and posture regulation. Note that the actions are joint torques rather than direct joint-angle commands. We evaluate this on Dog-Run by constraining the back-left leg and neck while retaining the original locomotion objective.

Leg constraint. We constrain the back-left hip, knee, and ankle to target angles of $42°$, $0°$, and $22°$, respectively, with a tolerance of $\pm 16°$ for each joint. We use a planning horizon of $H = 4$ with the terminal critic Q enabled. Figure 8 visualizes the target configuration relative to the neutral pose. During MPC, the predicted joint angles are read directly from the imagined physical-state trajectory, and a penalty is applied whenever a candidate trajectory violates the prescribed tolerance band. No model parameter is updated and the constraint is introduced only through the deployment-time planning objective.

Figure 9 shows that the unconstrained controller largely ignores the prescribed configuration, with the back-left-leg joints repeatedly deviating far beyond the $\pm 16°$ tolerance. In contrast, constrained S-TD-MPC keeps the leg substantially closer to the target throughout the episode, with only occasional transient deviations during dynamic running motions. The per-joint statistics further show that the hip, knee, and ankle remain centered near their respective targets under constrained planning. The latent-decoder baseline can also enforce the constraint reasonably well, confirming that decoded states can in principle support joint-space constraints. It achieves a *lower mean joint error* of $4.57°$ than S-TD-MPC ($7.72°$), but at a *substantially lower return* (303.4 versus 513.0). This highlights the task–constraint trade-off: stricter constraint satisfaction can come at the cost of degrading the original locomotion objective, while S-TD-MPC preserves considerably stronger running performance while still keeping the constrained joints close to the prescribed configuration.

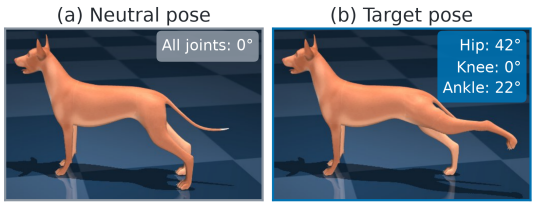


Figure 8: **Target back-left-leg configuration.** The neutral configuration sets all three joints to $0°$, while the deployment-time target specifies hip, knee, and ankle angles of $42°$, $0°$, and $22°$, respectively.

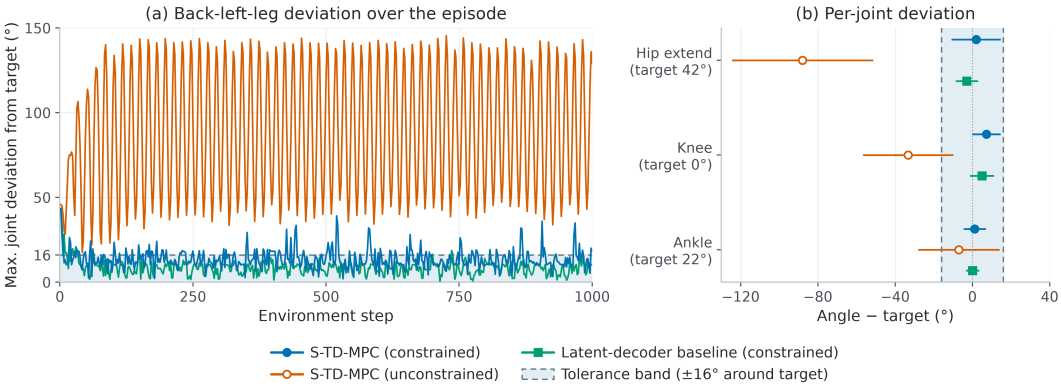


Figure 9: **Deployment-time back-left-leg joint constraint on Dog-Run.** Left: maximum deviation from the prescribed joint targets over the episode. The shaded region indicates the $\pm 16^\circ$ tolerance. Right: per-joint deviation from the target configuration. Constrained S-TD-MPC and the latent-decoder baseline evaluate the joint constraint during MPC, while the unconstrained controller uses the original Dog-Run objective only.

Neck constraint. An interesting behavior emerges in the unconstrained Dog-Run controller: despite achieving high return, the dog's neck remains persistently bent to the right during running. We therefore test whether this posture can be corrected purely at deployment time by constraining the seven neck joints toward a neutral configuration, without retraining the model or changing the original locomotion objective. We use a planning horizon of $H = 3$ with the terminal critic Q enabled.

Figure 10 shows that the unconstrained controller maintains a large neck deviation of roughly 25–30° throughout most of the episode, whereas constrained S-TD-MPC keeps the neck close to the prescribed $\pm 8^\circ$ tolerance for most of the trajectory, with only occasional transient violations. The per-joint statistics show the same pattern: the unconstrained controller exhibits large systematic offsets across several neck joints, while constrained planning shifts them substantially closer to the target configuration. The constrained controller achieves a mean joint error of only $3.92 \pm 0.11^\circ$, while the return decreases from 891.51 ± 10.06 for the unconstrained controller to 726.9 ± 25.3. The qualitative frames in Fig. 11 further confirm that the robot can maintain a visibly straighter neck while continuing to run.

The smaller performance loss compared with the leg constraint suggests that the effect of a deployment-time constraint depends strongly on which state variables are constrained and how tightly they are coupled to the original task. Neck configuration plays a relatively minor role in generating the running motion, so it can be altered while largely preserving the learned locomotion behavior. In contrast, constraining task-critical joints such as the legs changes the mechanics required for propulsion and can force planning toward behaviors that are poorly represented in the training distribution. More generally, deployment-time constraints are likely to be easier to satisfy when they act on state dimensions that are weakly coupled to the task objective, whereas constraints on task-critical states can induce larger performance degradation and stronger out-of-distribution model use.

Implementation details. Both joint-space constraints are imposed only through the MPC objective, without retraining either world model. At every imagined step, a quadratic penalty is applied to deviations of the predicted joint angles from the prescribed targets, while the original learned Dog-Run reward and terminal critic remain active. S-TD-MPC evaluates this penalty directly on its predicted physical states, whereas the latent-decoder baseline first reconstructs the corresponding joint states from its latent rollout. Both settings use 512 candidate trajectories, including 24 policy-prior rollouts, 64 elites, temperature 0.5, and 10 configured MPPI iterations. For the leg constraint, we use $H = 4$, planner std range $[0.02, 0.70]$, penalty coefficient 1.2, and a $\pm 16^\circ$ tolerance. For the neck constraint, we use $H = 3$, std range $[0.05, 0.85]$, coefficients 13 for bending joints and 8 for twisting joints, and a $\pm 8^\circ$ tolerance. The terminal critic Q is enabled in both joint-space constraint settings.

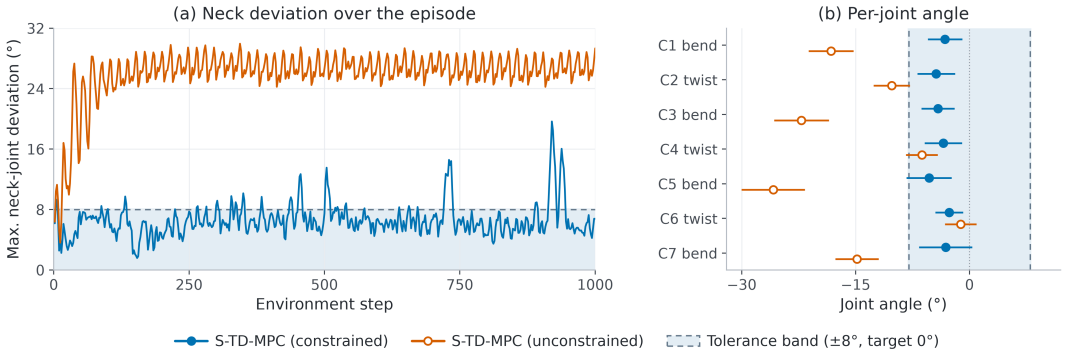


Figure 10: **Deployment-time neck constraint on Dog-Run.** Left: maximum neck-joint deviation over the episode for constrained and unconstrained S-TD-MPC. The shaded region denotes the $\pm 8^\circ$ tolerance around the neutral configuration. Right: per-joint neck angles, showing that constrained planning keeps all seven neck joints close to the target pose.



Figure 11: **Qualitative comparison under the neck constraint.** Representative frames from constrained and unconstrained Dog-Run trajectories. Constrained S-TD-MPC maintains a substantially straighter neck throughout running, while the unconstrained controller exhibits the characteristic bent-neck posture. The badges report the maximum neck-joint deviation at each frame.

C.4 Analytic Reward Evaluation

Explicit physical-state prediction also allows known task rewards to be evaluated directly on imagined trajectories. This recovers another standard advantage of model-based control: when the task objective is analytically specified, the controller can use it directly rather than learning a separate reward model. We therefore construct a known-reward variant of S-TD-MPC in which the learned reward network is removed and the original DMC reward function is evaluated directly on predicted physical states during planning and training. We study this variant on Dog-Run and Humanoid-Run.

Results and analysis. Figure 12 shows that replacing the learned reward model with the analytic reward does not yield a clear improvement in control performance on either task. Although the analytic reward provides exact knowledge of the task objective, the learned reward model in S-TD-MPC is already highly accurate: its relative reward prediction MAE is approximately 0.77% on Dog-Run and 0.27% on Humanoid-Run. Consequently, reward-model error is not a significant performance bottleneck in these experiments, leaving little room for the analytic reward to improve return. The main benefit is instead computational. As shown in Table 8, removing the reward network reduces the model size from 5.92M to 5.21M parameters on Dog-Run and from 5.09M to 4.46M on Humanoid-Run, while reducing training time from 19.9 to 17.7 hours and from 16.2 to 14.9 hours, respectively. Thus, explicit state prediction makes it possible to eliminate reward learning when the objective is known, reducing model complexity and training cost without sacrificing control performance.

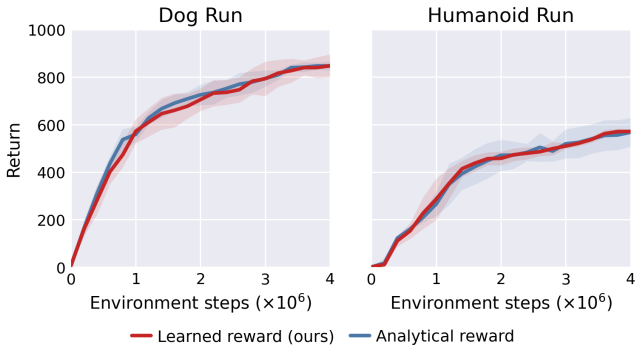


Figure 12: **Analytic versus learned reward models.** Training curves on Dog-Run and Humanoid-Run when S-TD-MPC uses either its learned reward model or the analytic DMC reward evaluated directly on predicted physical states. Solid lines denote the mean over three seeds and shaded regions indicate ± 1 standard deviation.

Table 8: Model size and wall-clock training time with learned and analytic rewards. Training time is averaged over three runs of 4M environment steps each.

	Dog-Run		Humanoid-Run	
Method	Params (M)	Time (h)	Params (M)	Time (h)
S-TD-MPC	5.92	19.9	5.09	16.2
Known-reward	5.21	17.7	4.46	14.9

Implementation details. The known-reward variant uses the same S-TD-MPC architecture, training procedure, planner, MPC-based value targets, policy distillation, and reanalysis settings, except that the learned reward head is removed and replaced by a fixed differentiable implementation of the corresponding DMC reward. During imagined rollouts, the analytic reward is evaluated directly from the predicted post-transition physical state and action. The same analytic reward is also used as reward-consistency supervision for the predicted states during training. All remaining hyperparameters are identical to the S-TD-MPC configuration in Table 4. Each method is trained for 4M environment steps over three seeds.