Fac-TDMPC: A Factored World Model for Robot Planning

Anonymous authors
Paper under double-blind review

Abstract

Model-based reinforcement learning (MBRL) has shown strong sample efficiency in robotics by learning predictive world models and planning with them, but existing methods suffer from high planning latency due to the combination of centralized world models and model predictive control (MPC) as planners, thus limiting the real-time deployment in high-dimensional action spaces. We introduce **Fac-TDMPC**, a factored latent-space world model that decomposes transition, reward, and value functions on the latent space and learns the factorization via model distillation. The factored design enables decentralized planning across action dimensions. Empirically, Fac-TDMPC achieves substantial planning speedups while preserving the control performance across a suite of continuous-control robotic tasks; it also demonstrates improved robustness to action perturbations, interpretable joint-level latent structure, and enhanced multi-task data efficiency.

1 Introduction

Reinforcement learning (RL) (Sutton & Barto, 1998) is a powerful paradigm for solving sequential decision-making problems, achieving remarkable success across diverse domains, such as mastering the game of Go (Silver et al., 2017) and controlling quadruped robots (Kostrikov et al., 2023). Model-based reinforcement learning (MBRL) (Moerland et al., 2023) is a prominent branch of RL that offers superior sample efficiency and improved interpretability compared to the model-free approaches (Wang et al., 2019; Ye et al., 2021). TDMPC (Hansen et al., 2022) and its successor TDMPC2 (Hansen et al., 2023) represent state-of-the-art MBRL methods in robotics. These frameworks learn a world model for representing robotic systems in latent spaces and perform trajectory optimization using model predictive control (MPC) in the latent space, enabling long-horizon planning and constraint handling.

Despite their successes, TDMPC-based frameworks typically employ a centralized world model with densely connected neural networks as dynamics, reward and value functions, resulting in an MPC optimization space of $\mathcal{O}(|\mathcal{A}|^H)$, where $\mathcal{A}$ denotes the action space and H is the planning horizon. For the high-dimensional action space, this leads to substantial computational overhead during planning, often referred to as *the curse of dimensionality* (Bertsekas, 2019), which restricts their scalability to more complex robotic scenarios (e.g. humanoid benchmark (Sferrazza et al., 2024)) and real-time deployment. Notably, in many real-world scenarios, planning over the entire action space is unnecessary, since the effects of different actions on the environment are often largely uncorrelated. For instance, a common approach to humanoid whole-body control is to decompose the upper and lower body, assigning balance and manipulation tasks to the respective parts (Lu et al., 2025).

To overcome the planning latency caused by the high-dimensional action space, we propose **Fac-TDMPC**, a novel extension of TDMPC that introduces **a factored world model** in the latent space. Instead of the densely connected neural networks used in TDMPC processing all action dimensions simultaneously, we employ factored dynamics, reward, and value functions for each action dimension i, following the factored decentralized Markov decision processes (fDec-MDPs) framework (Oliehoek & Amato, 2016). Given enough computational resources, this latent world model enables decentralized planning, reducing the optimization space to $\mathcal{O}(\max_i |\mathcal{A}^i|^H)$, where $\mathcal{A}^i$ denotes the action space of dimension i, thereby substantially reducing optimization complexity. To learn the factored world model, we employ a model distillation technique (Hinton et al., 2015) to transfer reward and value predictions from a centralized expert model. The objective of

distilling models for planning is to make the student model preserve the expert model's prediction ordering over actions (Rusu et al., 2016), i.e., $f_s(s, a_1) < f_s(s, a_2)$ if $f_e(s, a_1) < f_e(s, a_2), \forall a_1, a_2 \in \mathcal{A}, s \in \mathcal{S}$, where $\mathcal{S}$ is the state space and f_e, f_s are corresponding expert and student reward/value functions. We achieve this by incorporating Gaussian noise into action sequences and minimizing the KL divergence between the distributions predicted by their respective reward and value functions.

Beyond reducing planning latency, the factored latent model confers several practical advantages demonstrated in our experiments. (1) By enabling independent optimization per action dimension, Fac-TDMPC improves **robustness** to action perturbations: when individual actuators are disabled, or systematically scaled, the factored planner is less affected compared with the global planner. (2) The latent states and values of each agent can be separately visualized, which improves **interpretability** of the complete robotic system and helps to debug the failure cases. (3) Factored models naturally support **multi-task learning** across different robot embodiments: the shared joint-level structure allows data-efficient transfer, and in our multi-task learning experiments, Fac-TDMPC matches the centralized expert's average performance with a faster speed.

In summary, our contributions are as follows: (1) Fac-TDMPC—a novel MBRL framework that learns a factored world model to enable efficient planning in robotic tasks; (2) A model distillation procedure that transfers knowledge from a centralized expert world model (e.g., TDMPC) to a factored student world model preserving the prediction ordering over actions; (3) Empirical evidence showing that the proposed architecture achieves significant planning speedups, while maintaining the control performance and supporting robustness to action perturbations, explainability of robot behaviours and data-efficient multi-task learning.

2 Related Work

Model-based reinforcement learning (MBRL) (Moerland et al., 2023) integrates planning and learning by building a predictive model of the environment's dynamics and using it for decision-making, often achieving superior sample efficiency compared to model-free methods (Wang et al., 2019; Ye et al., 2021; Hansen et al., 2022; Hafner et al., 2023). Most existing approaches learn either reconstruction-based models (Hafner et al., 2023) or latent-space models with state abstractions (Hansen et al., 2022; 2023), which are used for planning at deployment. However, when unstructured latent-space models are combined with computationally heavy planners such as model predictive control (MPC), inference becomes slow, limiting real-time applicability in robotics. Our work, Fac-TDMPC, addresses this issue by learning factored world models and using decentralized planning to accelerate decision-making.

Our method requires learning a factored world model on which the latent state and action spaces are factored. The factorization idea has been explored in model-based planning settings. Some works (Balaji et al., 2020) assumed knowing the ground-truth factorization of the system before performing the planning process, which is limited to certain domains. Jiang et al. (2022) mapped the large action space into the compact latent space and planned on that latent space, but required a complex variational encoder to reconstruct the original actions. In contrast, our method learns the factored world model in the latent space, without the prior knowledge of the system, and speeds up planning without action reconstruction.

Our paper utilizes a decentralized model predictive path integral (MPPI) (Anderson & Milutinović, 2013) algorithm to efficiently plan on the factored world model. More efficient planning methods designed for different special structures exist. Variable elimination (Guestrin et al., 2001) and max-plus (Kok & Vlassis, 2006) are exact and approximated decentralized planning methods on the coordination graph (Guestrin et al., 2001). Liu et al. (2023) adopted mixed-integer programming for sparsified neural networks. Dec-MCTS (Claes et al., 2017; Czechowski & Oliehoek, 2020) is similar to our setup that allows each robot to optimize its own actions, but with additional effort on estimating the team plan. Our method can be extended with these advanced planning methods when considering a more complex framework in the latent space. However, since we enforce the fDec-MDP framework in the latent space as introduced in Section 4, the decentralized MPPI is sufficient for efficient planning.

Our approach is also related to multi-agent reinforcement learning (MARL) (Bertsekas, 2019), where high-dimensional action spaces present similar challenges. In robotics, some methods decompose a single robot into

multiple agents and apply MARL for control (Peng et al., 2021; Yan et al., 2024). However, these approaches require specifying factored state and action spaces *a priori*, which can be restrictive for generating the optimal actions. We instead learn the factored latent world model automatically, serving as an abstraction for efficient planning. In this work, we adopt reward and value decomposition from VDN (Sunehag et al., 2018) as a proof of concept, though more advanced MARL decompositions, such as QMIX (Rashid et al., 2018) or QTRAN (Son et al., 2019), could be integrated in future work.

3 Preliminaries

3.1 Factored Dec-MDP

A Markov decision process (MDP) (Bellman, 1957) is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mu_0, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\Delta_{\mathcal{S}}$ and $\Delta_{\mathcal{A}}$ are the space of probability measure, $T : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mu_0 \in \Delta_{\mathcal{S}}$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. A factored decentralized MDP (fDec-MDP) (Oliehoek & Amato, 2016) is a special case of the MDP to model multi-agent systems, where $\mathcal{S} = \times_{i=1}^{N} \mathcal{S}^i$ and $\mathcal{A} = \times_{i=1}^{N} \mathcal{A}^i$ partition state and action spaces into N disjoint subspaces. The fDec-MDP is said to be transition independent if $T(s_{h+1}|s_h, a_h) = \prod_{i=1}^{N} T_i(s_{h+1}^i|s_h^i, a_h^i)$, and reward independent if $R(s_h, a_h) = f(R_1(s_h^1, a_h^1) \dots R_N(s_h^N, a_h^N))$, where f is a *monotonic function*, $s_h^i, s_{h+1}^i \in \mathcal{S}^i, a_h^i \in \mathcal{A}^i$.

3.2 TDMPC

TDMPC (Hansen et al., 2022) falls into a class of model-based RL algorithms that learn a world model in the latent space (Schrittwieser et al., 2020; Hafner et al., 2023), which is aimed at facilitating real-time planning. Given that the robot receives image or proprioceptive state as global information g_t at step t, it internally maintains an MDP in the latent space $\mathcal{S}$, which consists of the following components: (1) an encoder $s = f_E(g)$ maps observations to latent states; (2) a transition function $s' = f_T(s, a)$ models the dynamics in the latent space; (3) a reward function $\hat{r} = f_R(s, a)$ predicts the reward on the latent space; (4) a terminal value function $\hat{q} = f_Q(s, a)$ predicts the discounted sum of rewards, where $s, s' \in \mathcal{S}, a \in \mathcal{A}$.

TDMPC maintains a replay buffer $\mathcal{B}$ with collecting trajectories of horizon H through interaction with the environment, and iteratively updates the world model using the data sampled from $\mathcal{B}$. The f_E, f_T, f_R, f_Q components are jointly optimized to minimize the following objective function:

$$\mathcal{L}(f_E, f_T, f_R, f_Q) = \mathbb{E}_{(g,a,r)_{0:H+1} \sim \mathcal{B}} \Big[\sum_{h=0}^{H} \lambda^h \Big(\|s_{h+1} - \text{sg}(f_E(g_{h+1}))\|_2^2 + \|\hat{r}_h - r_h\|_2^2 + \|\hat{q}_h - q_h\|_2^2 \Big) \Big], \tag{1}$$

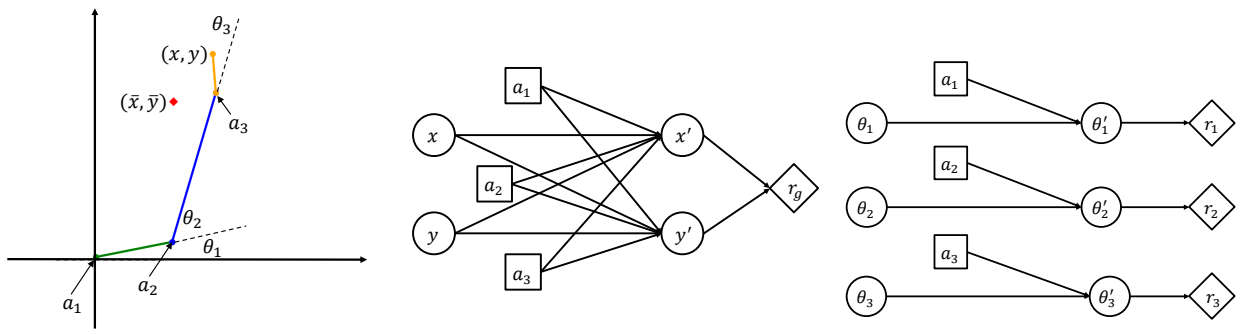
where $s_0 = f_E(g_0), s_{h+1} = f_T(s_h, a_h), \hat{r}_h = f_R(s_h, a_h), \hat{q}_h = f_Q(s_h, a_h)$ are predictions of the world model. sg is the stop-gradient operation, $q_h = r_h + \max_a \gamma f_{\bar{Q}}(f_E(g_{t+1}), a)$ is the TD-target ($f_{\bar{Q}}$ is an exponential moving-average version of f_Q), $\lambda \in [0, 1]$ is a hyperparameter to weigh less on further-step learning. TDMPC utilizes the learned world model by planning on the latent state space with model predictive path integral (MPPI) (Williams et al., 2016), a sampling-based MPC algorithm. Specifically, at each decision step t, the observation g_t is first transformed into the corresponding latent state as $s_t = f_E(g_t)$, and MPPI is leveraged to efficiently find the maximized solution of H-step return $G^H(s_t, a_{t:t+H})$ as follows:

$$\begin{aligned} a^*_{t:t+H} = \underset{a_{t:t+H}}{\text{argmax}} \quad G^H(g_t, a_{t:t+H}) = & \sum_{h=t}^{t+H-1} \gamma^{h-t} f_R(s_h, a_h) + \gamma^H f_Q(s_{t+H}, a_{t+H}), \\ \text{s.t.} \quad & s_t = f_E(g_t), s_{h+1} = f_T(s_h, a_h). \end{aligned} \tag{2}$$

The optimal solution of step t's action a_t^* is acquired and executed in the environment. Optimizing Equation 2 is computationally expensive for tasks with continuous action spaces due to the argmax operation. MPPI therefore implements a sample-based approximation: it iteratively samples and evaluates a batch of candidate action sequences based on the previous proposal and refines the proposal by a weighted average (typically an importance-weighted average) of the higher-return samples. With sufficient samples per iteration and enough iterations, the proposal distribution concentrates on near-optimal trajectories of Equation 2.

4 Fac-TDMPC

4.1 A Motivating Example



(a) Moving a 3-link 2-dimensional manipulator to the goal.

(b) DBN of a centralized MDP with the end-effector position as states.

(c) DBN of an fDec-MDP with the joint angles as states.

Figure 1: Different modelling methods to describe the task of moving a manipulator to the goal.

Before diving into our method, we first start with a simple manipulator task to motivate the use of the factored model. As illustrated in Figure 1a, the task is to move the end effector position (x, y) of a 3-link manipulator to the goal position $(\bar{x}, \bar{y})$ on the 2-dimensional plane, with joint angles denoted as $\theta = (\theta_1, \theta_2, \theta_3)$. Notably, the end effector position and joint angles can be inter-converted with forward and inverse kinematics, written as $(x, y) = f_{FK}(\theta) = f_{FK}(f_{IK}(x, y))$. When the range of (x, y, θ) is restricted, such a transformation is bijective. The action $a = (a_1, a_2, a_3)$ consists of the forces put on the 3 joints. The joint force a_i is first sent into a non-linear actuator model $\theta_i' = f_a(a_i, \theta_i)$ to achieve the next joint angle θ_i' and further influence the end effector's position.

One can simply use the end effector position (x, y) as the state. Then the transition function becomes $(x', y') = f_{FK}(f_a(\theta_1, a_1), f_a(\theta_2, a_2), f_a(\theta_3, a_3))$, where $\theta = f_{IK}(x, y)$ and the reward function is $f_R(x, y, a) = -\|(x, y) - (\bar{x}, \bar{y})\|_2^2$. This centralized MDP model can be visualized as a dynamic Bayesian network (DBN) (Boutilier et al., 1999) in Figure 1b. Alternatively, one can view the joint angles as states, hence the problem can be formulated as an fDec-MDP with 3 agents. For each agent i, the transition function is simply the actuator model $\theta_i' = f_a(\theta_i, a_i)$, so as to be transition independent. The individual reward function $f_{R_i}(\theta_i, a_i) = -(\theta_i - \bar{\theta}_i)^2$, where $\bar{\theta} = f_{IK}(\bar{x}, \bar{y})$. When closed to the optimal point, the global reward function is monotonic on individual reward functions, written as $f_R(x, y, a) = M(f_{R_1}(\theta_1, a_1), f_{R_2}(\theta_2, a_2), f_{R_3}(\theta_3, a_3))$, where M is a monotonic function (See Appendix A.1 for its detailed form).

Although these two modelling methods are equivalent to describe the system, planning using different models can result in huge differences. We apply the model predictive path integral control (MPPI) (Williams et al., 2016) on both models, but for fDec-MDP, each agent can be optimized independently and in parallel. Although both methods can reach the goal position in finite steps, the average planning time on the centralized MDP is 1.20 seconds per step (due to the expensive calculation of inverse and forward kinematics), while the average planning time on the fDec-MDP is 0.04 seconds per step, with around **30**× speedup (See Appendix A.2 for details). This example directly motivates us to incorporate the fDec-MDP modelling into the TDMPC framework for more efficient planning methods as introduced below.

4.2 Factored World Model

From Section 3.2, we observe that all planning in TDMPC is performed within the latent world model. **This motivates us to pose a research question: whether there exists a corresponding factored latent world model that supports optimal control, as the example in Section 4.1.** Such a factored world model is expected to be equivalent to the original in terms of generating optimal actions, but enables more efficient execution of planning algorithms such as MPPI.

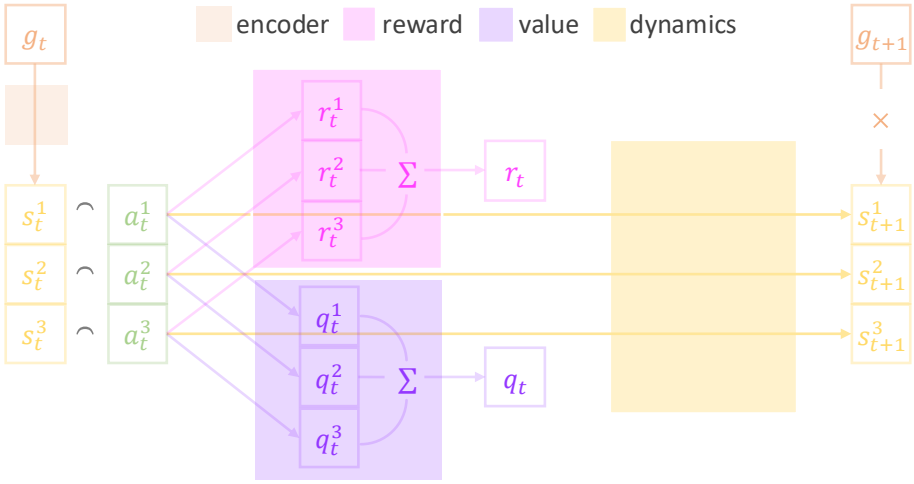


Figure 2: fDec-MDP modelling in the TDMPC framework. The observation g_t is first encoded to the latent state s_t by the encoder function $s_t = f_E(g_t)$. The latent state is factored to $N = \dim(\mathcal{A})$ agents as $s_t = \times_{i=1}^N s_t^i$. On each action dimension i, the independent dynamics function f_{T_i}, reward function f_{R_i}, value function f_{Q_i} are defined locally. The global reward and value can be summed up by the local ones.

To this end, we represent the latent world model as an fDec-MDP model (introduced in Section 3.1). We first view each action dimension of the robot as an independent agent, yielding $N = \dim(\mathcal{A})$ agents. Then, we design the factored latent space as $\mathcal{S} = \times_{i=1}^N \mathcal{S}_i : \mathbb{R}^{mN}$, where m is the dimension for a single agent's hidden state. The encoder module is a centralized neural network as $s_t = f_E(g_t)$. In order to fully decompose agents for efficient planning, we further assume that the fDec-MDP is transition-independent so that for each agent i, it maintains an independent local transition function $s_{t+1}^i = f_{T_i}(s_t^i, a_t^i)$. Meanwhile, each agent maintains an independent reward function and a value function, written as $r_t^i = f_{R_i}(s_t^i, a_t^i)$ and $q_t^i = f_{Q_i}(s_t^i, a_t^i)$. The global reward describing the system-wise performance is the sum of all individual rewards: $r_t = f_R(s_t, a_t) = \sum_{i=1}^N f_{R_i}(s_t^i, a_t^i)$. This condition is stronger than that of the example introduced in Section 4.1. The reward function in the example is strictly concave, so it is sufficient to ensure monotonicity only near the optimal point. However, since a general neural network is used here, we impose monotonicity of the global reward with respect to the individual rewards across all states s. As mentioned in Oliehoek & Amato (2016), transition independence and reward decomposition ensure that the value function is naturally decomposable, written as $q_t = f_Q(s_t, a_t) = \sum_{i=1}^N f_{Q_i}(s_t^i, a_t^i)$.

In summary, the complete factored world model consists of a centralized encoder function f_E, a factored transition function $f_T = (\times_{i=1}^N f_{T_i})$, a factored reward function $f_R = (\times_{i=1}^N f_{R_i})$, and the factored value function $f_Q = (\times_{i=1}^N f_{Q_i})$, as illustrated in Figure 2. During deployment, at each step t, the encoder f_E first maps the global information g_t to each agent's local state s_t^i. Based on this, each agent generates its own optimal action a_t^{i*} using a planning algorithm π_i (e.g., MPPI) and its learned factored world model $(f_{T_i}, f_{R_i}, f_{Q_i})$. The system-wide optimal action is then composed as $a_t^* = \times_{i=1}^N a_t^{i*} = \times_{i=1}^N \pi_i(s_t^i) = \times_{i=1}^N \pi_i(f_E(g_t)^i)$.

The latent state s_t can be viewed as an abstraction of the global information g_t. Let the ground-truth reward and value function given the global state g_t and action a_t be denoted by $R(g_t, a_t)$ and $Q(g_t, a_t)$. Previous work (Li et al., 2006; Ni et al., 2023) categorized abstractions based on the information preserved in the latent state s_t that supports different predictions. In our factored world model, these abstractions are represented as: (1) **Q-irrelevance abstraction** (ϕ_Q) for predicting the return, i.e. $Q(g_t, a_t) = f_Q(f_E(g_t), a_t) = \sum_{i=1}^N f_{Q_i}(s_t^i, a_t^i)$; (2) **reward prediction** (RP) for predicting the reward, i.e. $R(g_t, a_t) = f_R(f_E(g_t), a_t) = \sum_{i=1}^N f_{R_i}(s_t^i, a_t^i)$; (3) **next latent state prediction** (ZP) for predicting the next latent state, i.e. $s_{t+1} = f_E(g_{t+1}) = f_T(f_E(g_t), a_t) = \times_{i=1}^N f_{T_i}(s_t^i, a_t^i)$. Since we aim to learn an efficient world model that can equivalently generate optimal actions, we maintain abstractions in the level of ϕ_Q

and RP, but not ZP which would impose a stricter but unnecessary level of abstraction (expressed as the cross mark between g_{t+1} and s_{t+1} in Figure 2). This relaxation broadens the applicability of the fDec-MDP framework from fully decomposable robotic tasks, as in the example, to more general robotic tasks—a claim further validated in our experiments.

4.3 Decentralized Planning

As one of the main benefits of adopting the factored model, we can further achieve an efficient planning method on the fly. Following the H-step return in TDMPC, we can similarly define an individual H-step return for agent i with its individual dynamics, reward and value functions as $G_i^H(g_t, a_{t:t+H}^i) = \sum_{h=t}^{t+H-1} \gamma^{h-t} f_{R_i}(s_h^i, a_h^i) + \gamma^H f_{Q_i}(s_{t+H}^i, a_{t+H}^i), \text{s.t.} s_t^i = f_E(g_t)^i, s_{h+1}^i = f_{T_i}(s_h^i, a_h^i)$, which is underpinned by the following proposition:

Proposition 1 (Individual Global Max) *For the fDec-MDP system with independent transition, reward and value functions, the H-step return $[G_i^H]$ satisfies the individual global max (Son et al., 2019) under observation g_t, which equivalently says:*

$$a_{t:t+H}^* = \underset{a_{t:t+H}}{argmax} \quad G^H(g_t, a_{t:t+H}) = \times_{i=1}^N \underset{a_{t:t+H}^i}{argmax} \quad G_i^H(g_t, a_{t:t+H}^i). \tag{3}$$

The proposition results from the additivity of the reward and value functions, and the independence of the dynamics function. It implies that each action dimension can be optimized in a decentralized manner for $\boldsymbol{N}$ separate subsystems, without impairing the system-wise performance. For such a factored model, the optimization space of an MPC has been reduced from $\mathcal{O}(|\mathcal{A}|^H) = \mathcal{O}(\times_{i=1}^N |\mathcal{A}^i|^H)$ to $\mathcal{O}(\sum_{i=1}^N |\mathcal{A}^i|^H)$. Moreover, given sufficient computational resources, planning for each action dimension can be executed in parallel, yielding an additional N-fold speedup, with the time complexity $\mathcal{O}(\max_i |\mathcal{A}^i|^H)$.

4.4 Model Distillation

As introduced in Section 3.2, TDMPC employs a centralized world model $(f_{E_e}, f_{T_e}, f_{R_e}, f_{Q_e})$, which achieves strong task performance but is inefficient for planning. Our objective is to develop a factored model (f_E, f_T, f_R, f_Q) that enables more efficient planning without sacrificing performance. To this end, we leverage model distillation (Hinton et al., 2015) to transfer knowledge from the centralized expert model of TDMPC. The distillation procedure is summarized in Algorithm 1.

Algorithm 1 Model Distillation

1: **Input**: expert model $(f_{E_e}, f_{T_e}, f_{R_e}, f_{Q_e})$, dataset $\mathcal{B}$, hyperparameter λ, H
2: **Output**: factored model $(f_E, \times_{i=1}^N f_{T_i}, \times_{i=1}^N f_{R_i}, \times_{i=1}^N f_{Q_i})$
3: **for** iteration $i = 1, 2, ..., I$ **do**
4: Sample a H-step sequential samples $(g, a)_{0:H}$ from the dataset $\mathcal{B}$
5: Update parameters of the factored model based on the loss function as follows:
6: $\mathcal{L}(f_E, f_T, f_R, f_Q) = \mathbb{E}_{(g,a)_{0:H} \sim \mathcal{B}} \Big[\sum_{h=0}^H \lambda^h \Big(D_{\mathrm{KL}}(\mathcal{R}_h, \mathcal{R}_h^e) + D_{\mathrm{KL}}(\mathcal{Q}_h, \mathcal{Q}_h^e) \Big) \Big]$

The loss function in Line 6 aligns the reward and value *distributions* predicted by the factored model with those of the expert model via KL divergence. These objectives are introduced to ensure the ϕ_{Q^*} and RP level of state abstractions as introduced in Section 4.2. Following Rusu et al. (2016), we preserve the ordering of predicted actions rather than directly matching scalar values, since this has been shown to yield more effective knowledge transfer. Concretely, for a trajectory $(g, a)_{0:H}$ we inject Gaussian perturbations $\epsilon_{0:H} \sim \mathcal{N}^{H+1}(0, \Sigma)$ into actions and propagate them through the dynamics $s_{h+1} = f_T(s_h, a_h + \epsilon_h)$, yielding soft action distributions over rewards and values:

$$\mathcal{R}_h(a_h + \epsilon_h) \propto \exp\big(f_R(s_h, a_h + \epsilon_h)/T_s\big), \quad \mathcal{Q}_h(a_h + \epsilon_h) \propto \exp\big(f_Q(s_h, a_h + \epsilon_h)/T_s\big), \tag{4}$$

where T_s is a temperature controlling the softness of the student predictions (Hinton et al., 2015). The expert distributions $\mathcal{R}_h^e$ and $\mathcal{Q}_h^e$ are defined analogously but with a different temperature T_e. We then

minimize the KL divergence between student and expert distributions, which preserves the relative ranking of actions while allowing flexibility in scale. In practice, we approximate the KL terms with a small number of noise samples (10 in our experiments). Unlike the TDMPC objective (Equation 1), we omit explicit consistency losses, since our goal is a factored state abstraction that is optimized for planning rather than strict self-prediction (Li et al., 2006), as explained in Section 4.2.

5 Experiments

5.1 Experimental Setup

(a) Reacher (b) Hopper (c) Fish (d) Cheetah (e) Walker (f) Quadruped (g) Humanoid

Figure 3: Visualization of robots used in the experiments, with increased complexity in action space $\dim(\mathcal{A}) = \{3, 4, 5, 6, 6, 12, 21\}, \dim(\mathcal{O}) = \{8, 15, 24, 17, 24, 78, 67\}$.

Task Setup. We evaluate our method on DMControl (Tassa et al., 2018) benchmarks powered by the MuJoCo (Todorov et al., 2012) simulator. We select 7 robots with increasing state and action complexity as shown in Figure 3 following TDMPC (Hansen et al., 2022), but only robots with more than 3 action dimensions to validate the factored design, and meanwhile, we change the 2-link reacher to 3-link one; this yields 11 tasks in total (see Appendix B).

Expert and Dataset Collection. We train TDMPC with default hyperparameters (Hansen et al., 2022) on each task. The final checkpoint serves as the expert model, and its replay buffer of 10^6 transitions is used as the offline dataset.

Baselines. (1) **X% TDMPC** utilizes the fully centralized models as the expert model but with $X\%$ parameters, and trained with model distillation as introduced in Section 4.4. $X = 100, 10$ are set for the performance of the full and smaller expert models; (2) **TAP** (Zheng et al., 2020) reduces computational efficiency by transforming the original action space into a latent action space with VAE (Kingma & Welling, 2014) and planning in the latent space; (3) **Fac-TDMPC** is our proposed method as described in Section 4. A more detailed introduction on all baselines can be found in Appendix B.

Training hyperparameters. We perform model distillation (Section 4.4) for 10^5 optimization steps and evaluate the factored model every 5×10^3 steps over 10 episodes. TDMPC and TAP's latent dimension is 512 in all experiments; Fac-TDMPC uses a per-agent latent size of $\lfloor 512/N \rfloor$ to keep parameter counts comparable. Other hyperparameters are shared across baselines and listed in Appendix B.3.

5.2 Training Performance

Figure 4 presents the training performance of all baselines. For nearly all tasks, Fac-TDMPC matches or exceeds the asymptotic performance of the expert model (TDMPC), demonstrating that the proposed factored architecture does not sacrifice control quality despite its reduced computational footprint (shown in the next section). In some challenging tasks, such as `Quadruped Walk` and `Humanoid Walk`, Fac-TDMPC even reaches superior performance than the expert model, which indicates the advantage of imitating the world model over simply behaviour cloning. 100% TDMPC attains similar performances as Fac-TDMPC and sometimes converges faster due to the same structure of the expert model. Notably, it fails the most challenging task, `Humanoid Walk`, showing the learning difficulty of such a large action space. In comparison, 10% TDMPC and TAP, with many fewer parameters, achieve reasonable performance in simple robots but cannot accommodate more complex ones. We do a detailed ablation study on key design choices of Fac-TDMPC in Section 5.3.

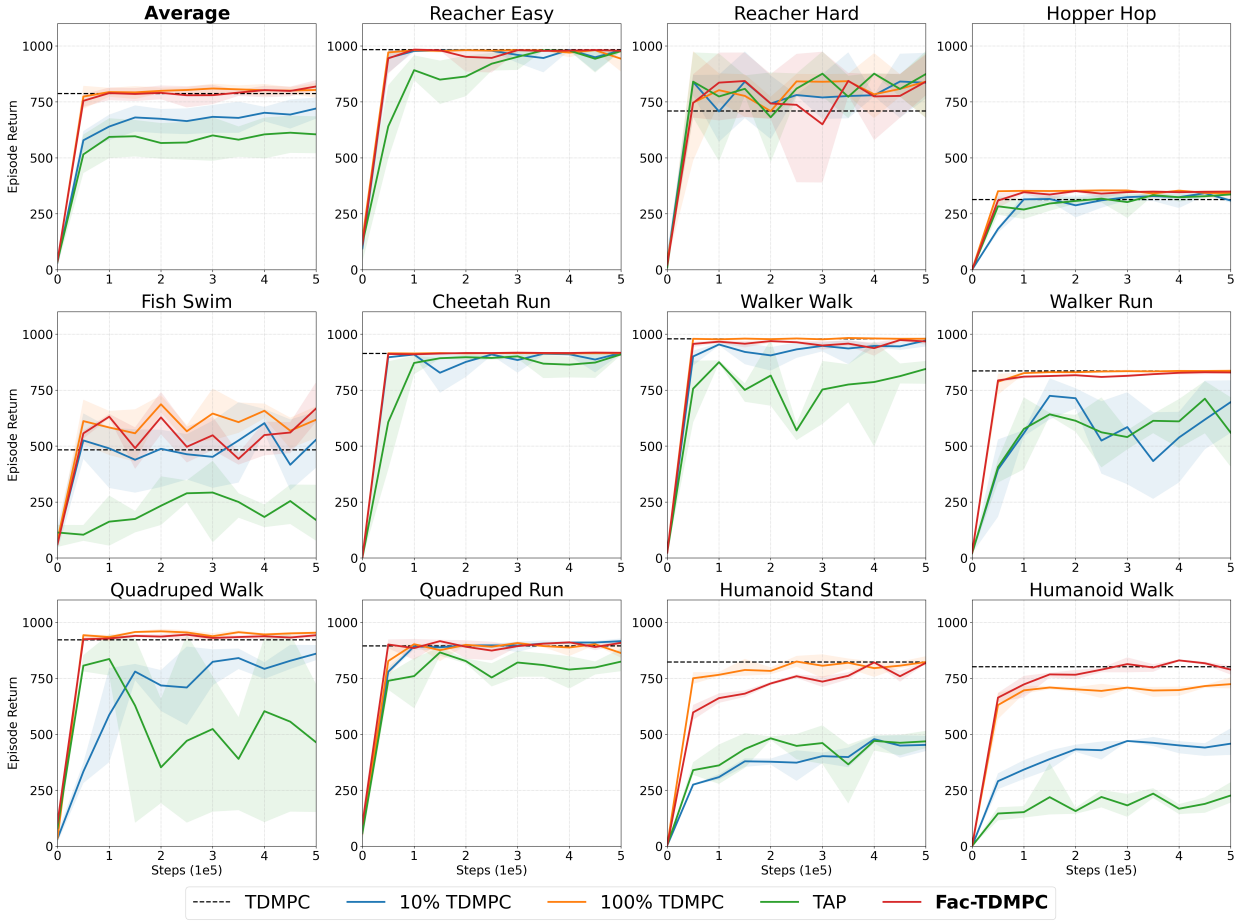
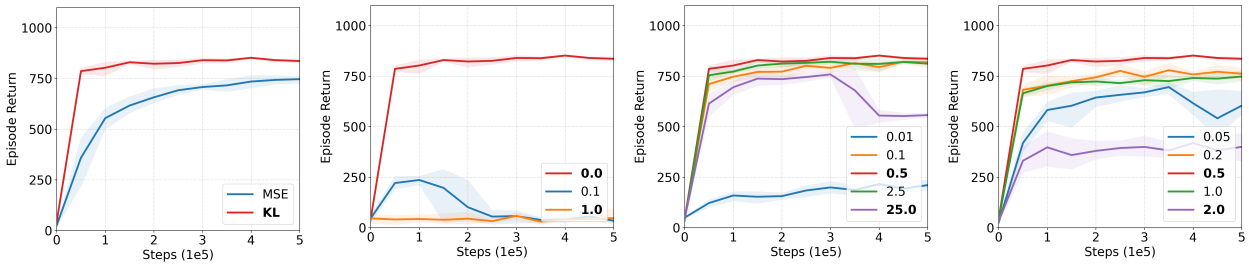


Figure 4: Training curves of all baselines on MuJoCo tasks. Each curve shows the average episode return over three seeds, with shaded regions denoting the standard deviation.

We also evaluate Fac-TDMPC in an online model distillation scheme, where an expert model is trained from scratch following the online reinforcement learning process, and the student must simultaneously track the expert during the learning process. The results in Appendix C.1 illustrate that Fac-TDMPC can successfully track TDMPC in challenging tasks even when TDMPC is constantly updating. This finding is significant in that the efficient student model can be acquired simultaneously with the learning process of the full expert model, thereby saving training time.

5.3 Ablation Study

Key Design Choices. We conduct an ablation study on the three most challenging tasks (`Walker Run`, `Quadruped Run`, and `Humanoid Walk`) to assess the impact of key design choices. As shown in Figure 5, the KL-based loss consistently outperforms MSE, highlighting that ranking the optimal action is more efficient than regressing the exact value in policy distillation (Rusu et al., 2016). Within the KL setup, moderate values of both the noise standard deviation and temperature play crucial roles: a larger temperature smooths the target, while a smaller temperature overly emphasizes one particular action; a larger standard deviation introduces too much noise to the learning, while a smaller standard deviation causes overfitting of the optimal actions in the dataset. Surprisingly, adding a consistency loss as in TDMPC (Hansen et al., 2022) actually harms performance. As mentioned in Section 4.2, Fac-TDMPC actually aims to learn an equivalent world model in the Q-irrelevance (ϕ_Q) and reward prediction (RP) abstraction instead of the next latent state prediction (ZP) abstraction. It indicates that the temporal consistency of the latent states is overly restrictive



(a) Distillation loss type (b) Consistency loss coefficient (c) Expert model temperature (d) Standard deviation of samples

Figure 5: Ablation study on key design choices of Fac-TDMPC on challenging (`Walker Run`, `Quadruped Run`, and `Humanoid Walk`) tasks. Each curve shows the average episode return over three seeds, with shaded regions denoting the standard deviation.

for the factored structures and unnecessary to achieve good planning performance. In addition, we perform an ablation study on the number of agents in our Fac-TDMPC world model, explained in Appendix 5.3.

5.4 Planning Efficiency

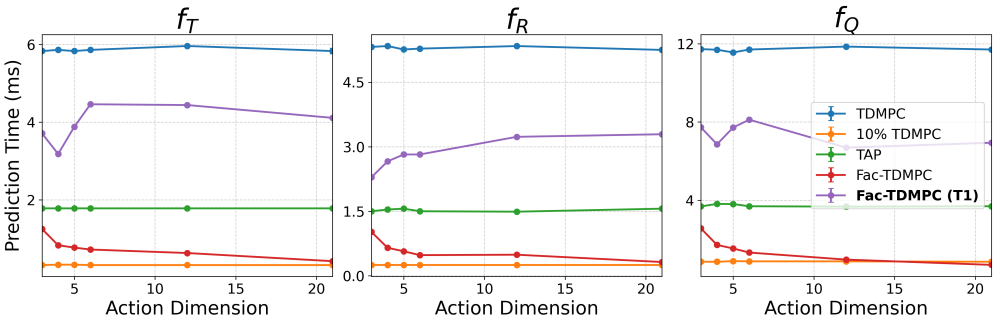


Figure 6: Prediction time of different models on different robots. **All wall times are reported in milliseconds** with average and standard deviation over 100 runs.

In this section, we validate the core motivation behind the proposed Fac-TDMPC: the learned factored structure enables more efficient downstream planning, in particular when applied with the MPPI algorithm (Williams et al., 2016). For MPPI, several key design choices critically affect planning efficiency: (i) the prediction time of the models, which determines the cost of a single run, and (ii) the number of samples and optimization iterations, which determine how many runs are required for precise planning.

Figure 6 compares the prediction time of all baseline models in robots with different action dimensions, since the Fac-TDMPC model structure relates only to the number of action dimensions. We only compare the speed of (f_T, f_R, f_Q) without the encoder f_E since all baselines apply the same size of feedforward network as encoders. All baselines are tested on a 1-thread CPU and Fac-TDMPC on N-thread CPU, depending on the robot's action dimensions N, to enable parallel prediction. Notably, the prediction time for $X\%$ TDMPC and TAP are similar across different robots, since the model structures are unchanged. Across all robots and model components, Fac-TDMPC achieves a substantial reduction in per-step computation time compared to TDMPC, and the efficiency gain continues to improve with the increase in action dimensions. Furthermore, we evaluate the inference time of Fac-TDMPC in a single-threaded setting ("T1"), where it exhibits computational requirements comparable to the TDMPC baseline. This observation confirms that the observed speedup is primarily driven by the parallelized prediction capabilities of the factored architecture. Specifically, for an action space of dimension N, a baseline TDMPC model of size $\mathcal{O}(K)$ is decomposed into N individual sub-models, each with a size of approximately $\mathcal{O}(K/\sqrt{N})$ to maintain a comparable total

parameter count. This factored structure facilitates more efficient computation on a single-threaded CPU and further accelerates the optimization and inference processes when distributed across N independent threads, thereby enabling significant wall-clock efficiency through parallel computation. Its speed even matches 10% TDMPC for humanoid robots, but the control performance is much better as shown in Section 5.2.

Prediction time determines the cost of a single call of the models, while the number of calls also matters. In MPPI, more samples and iterations implies more calls to the models. We test the control performance of the `Walker Run` task with various numbers of samples and iterations, which is reported in Figure 7. The default numbers of samples and iterations are 512 and 6. Reducing them will undermine the control performance of both TDMPC and Fac-TDMPC. However, the Fac-TDMPC suffers a much smaller reduction, and even achieves acceptable performance for 128 samples or 2 iterations, which indicates the possibility of reducing computational cost by $3 \sim 4\times$ without sacrificing planning performance. We attribute this to the individual global max property of the return (Proposition 1), which reduces the optimization space from $\mathcal{O}(\times_{i=1}^{N}|\mathcal{A}^i|^H)$ to $\mathcal{O}(\sum_{i=1}^{N}|\mathcal{A}^i|^H)$. This leads the factored world model to better utilization of limited optimization resources, which is crucial in robotic tasks (Duan et al., 2025).

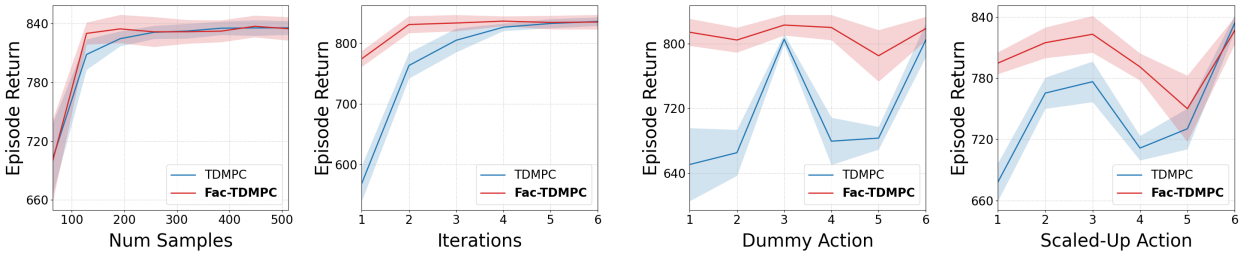


Figure 7: Control performance on `Walker Run` with various optimization parameters.

Figure 8: Control performance on `Walker Run` under various perturbations.

5.5 Robustness to Action Perturbations

The factored structure permits independent optimization of each action dimension that can improve robustness to action perturbations. We evaluate two perturbations on the `Walker Run` task: (1) one action dimension is *dummy* that always follows a Normal distribution; (2) one action dimension is consistently scaled, simulating modified motor strength. Results in Figure 8 show that certain action dimensions are more sensitive to the perturbations, and Fac-TDMPC maintains more stable and higher episode returns across perturbed dimensions.

5.6 Visualization on Factored World Model

In this section, we visualize the learned Fac-TDMPC world model to closely explain the factored behaviours. We select the 3-link reacher robot as an example, and the goal is to reach the red point in the 2-dimensional planar space. The global reward is decided by the distance between the goal position and the end-effector position. For Fac-TDMPC, 3 agents are collaborating to control the joints and finish the task.

We first visualize the latent states of all agents collected from 100 episodes. Each agent's latent state is a 50-dimensional vector; we project these vectors into two dimensions using t-SNE (van der Maaten & Hinton, 2008), as shown in Figure 9. The projection reveals distinct clusters corresponding to different joints. Interestingly, these three clusters exhibit a degree of rotational invariance, which implies that the individual latent states also encode some common knowledge of the entire system for further coordination, cf. the joints' local views of the robot's global states (Yan et al., 2024). We visualize individual and global H-step returns for three representative episodes in Figure 9, and include two rendered snapshots (top) corresponding to the steps marked by vertical dashed lines in the plots. The first two illustrate successful coordination, while the third shows a failure case. In the first success, all joints undergo large angular changes to reach the target position, leading to consistent increases in both individual and global returns. In the second success, the

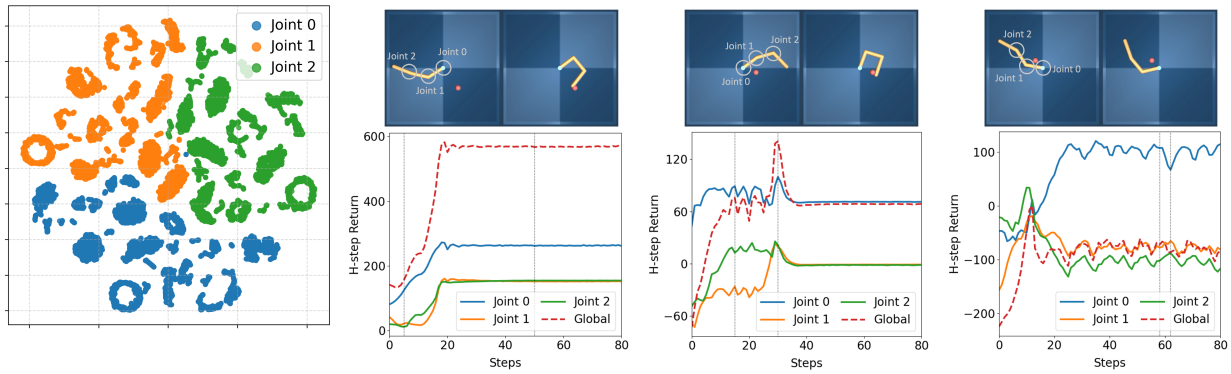


Figure 9: Visualization of latent states and credit assignment. **Left:** t-SNE projection of latent states shows clear clustering by joints. **Middle and right:** individual and global H-step returns of different episodes. The upper two rendered images correspond to the steps with vertical dashed lines in the plots.

phase shift is primarily driven by Joint 1, which accounts for the sharp rise in the orange and red curves. In contrast, the failure case exhibits oscillatory patterns across all curves, where the efforts of Joint 0 conflict with those of Joints 1 and 2. Among these states, the additive reward and value functions are insufficient to capture the global behaviours. Addressing such cases may require more advanced coordination strategies, such as Q-MIX (Rashid et al., 2018). In general, visualizations of latent states and returns provide a powerful tool for analyzing robotic systems and explaining their behaviours.

5.7 Multi-task Learning

As a final contribution, we investigate the performance of Fac-TDMPC in a multi-task learning scenario. We adopt the multi-task benchmark introduced in Hansen et al. (2023), which aggregates 30 tasks from the DeepMind control suite (Tassa et al., 2018). These tasks span 10 robot embodiments with action dimensionalities ranging from 1 to 6. The dataset contains trajectories collected by a 5M-step TDMPC2 expert policy (Hansen et al., 2023), resulting in approximately 3.5×10^8 offline transitions. The expert achieves an average episode return of 283.0 across the tasks.
To apply Fac-TDMPC in this setting, we train a *single shared factorized world model* across all embodiments and tasks. For a robot with N actuated joints, the latent state s_t is decomposed into N joint-specific latent factors,

$$s_t = \{s_t^1, \ldots, s_t^N\}, \tag{5}$$

where each factor corresponds to one action dimension (joint). To capture task-specific variations such as robot morphology or control objectives, we follow Hansen et al. (2023) and introduce a learned task embedding e_k for each task k, which is further concatenated to each factor's latent state to an augmented latent state $\bar{s}_t = \{(s_t^1, e_k), \ldots, (s_t^N, e_k)\}$. Each joint factor is modeled using a shared set of networks that parameterize its latent dynamics, reward and value functions $(f_{T_i}, f_{R_i}, f_{Q_i})$. The dynamics prediction for task k's joint i is represented as $s_{t+1}^{(i)} = f_{T_i}(s_t^i, a_t^i, e_k)$. Since the shape of the augmented factored latent state (s_t^i, e_k) is fixed , the factored world model can be shared *shared across tasks*, allowing the model to reuse joint-level knowledge across different robot embodiments. The world model and policy are trained jointly on the pooled offline dataset following the model distillation training procedure as in Section 5.1.

The leftmost plot of Fig. 10 shows that Fac-TDMPC surpasses the performance of the expert policy while exhibiting significantly improved data efficiency compared to fully centralized models, particularly during the early stages of training. To further analyze the learned representation, we visualize the latent states of the walker and hopper robots. The hopper can be interpreted as a one-legged variant of the walker with an additional waist joint. The middle plot of Fig. 10 shows the projected latent states for three tasks involving the walker robot. The resulting embeddings form distinct clusters corresponding to individual joints, and these clusters remain consistent across tasks with the same embodiment. The second plot from

the right illustrates cross-embodiment knowledge transfer between the walker and hopper robots. We observe weak but consistent correlations between the walker's joints 0–2 and the hopper's joints 1–3 in the projected latent space. These joints correspond to the hip, knee, and ankle of one leg in both embodiments. This alignment indicates that the factorized representation captures shared joint-level dynamics across different robots, which helps explain the improved data efficiency of Fac-TDMPC.

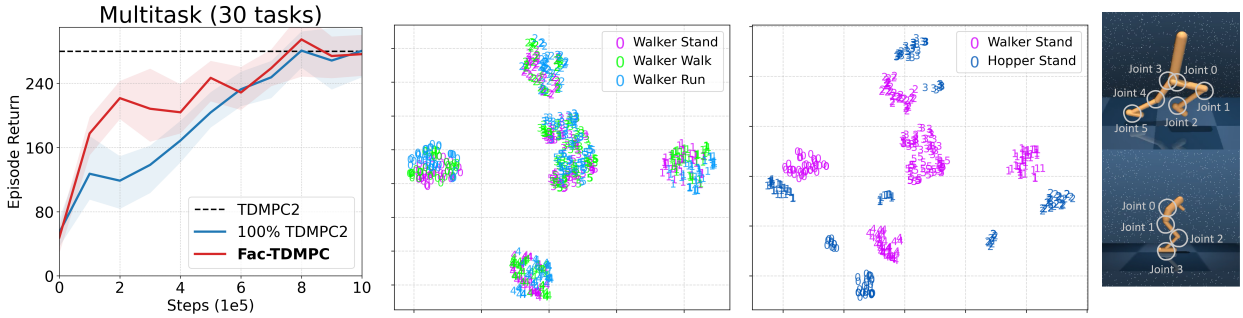


Figure 10: Multi-task learning results. **Left:** Training curves of multi-task learning. The episode return is averaged across 30 tasks. **Middle and right:** t-SNE projection of latent states both within robots (walker) and between robots (walker and hopper). The sign of the points indicates the actual joint index.

6 Conclusion

This paper introduces Fac-TDMPC, a factored world model in the latent space, enabling fast decision-making on deployment and robustness for high-dimensional action scenarios. Fac-TDMPC achieves significant speedups in planning through decentralized planning while maintaining high control performance on various robotic tasks in MuJoCo. Alongside, the learned factored structure enhances robustness to action perturbations.

Notably, in a few highly dynamic tasks that require tight and global coordination, Fac-TDMPC can learn more slowly than a centralized world model. To address these limitations we plan several extensions in future work: (1) incorporate latent histories or recurrent encoders to capture temporal coupling and better model interactions between dimensions (Oliehoek et al., 2021); (2) integrate advanced credit-assignment mechanisms (e.g., QMIX (Rashid et al., 2018) or QTRAN (Son et al., 2019) factorization) to better enforce cooperative behaviors; (3) explore hybrid or learned coupling modules and attention-based factor discovery to automatically and dynamically decompose the action space rather than assuming fixed independence.

References

Ross P. Anderson and Dejan Milutinović. Distributed path integral feedback control based on kalman smoothing for unicycle formations. In *2013 American Control Conference*, pp. 4611–4616, June 2013. doi: 10.1109/ACC.2013.6580550.

Bharathan Balaji, Petros Christodoulou, Xiaoyu Lu, Byungsoo Jeon, and Jordan Bell-Masterson. Factoredrl: Leveraging factored graphs for deep reinforcement learning. In *arXiv*, 2020.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 0095-9057.

Dimitri P. Bertsekas. Multiagent rollout algorithms and reinforcement learning. *CoRR*, abs/1910.00120, 2019.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, July 1999. ISSN 1076-9757. doi: 10.1613/jair.575.

Daniel Claes, Frans A. Oliehoek, Hendrik Baier, and Karl Tuyls. Decentralised online planning for multi-robot warehouse commissioning. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee (eds.), *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pp. 492–500. ACM, 2017.

Aleksander Czechowski and Frans A. Oliehoek. Decentralized mcts via learned teammate models. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 81–88. ijcai.org, 2020. doi: 10.24963/IJCAI.2020/12.

Zhekai Duan, Yuan Zhang, Shikai Geng, Gaowen Liu, Joschka Boedecker, and Chris Xiaoxuan Lu. Fast ecot: Efficient embodied chain-of-thought via thoughts reuse, 2025.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023. doi: 10.48550/ARXIV.2301.04104.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, October 2023.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 8387–8406. PMLR, June 2022.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations*, September 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(65):1789–1828, 2006. ISSN 1533-7928.

Ilya Kostrikov, Laura M. Smith, and Sergey Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.056.

Anson Lei, Bernhard Schölkopf, and Ingmar Posner. Variational causal dynamics: Discovering modular world models from interventions, June 2022.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In *International Symposium on Artificial Intelligence and Mathematics, AI&Math 2006, Fort Lauderdale, Florida, USA, January 4-6, 2006*, 2006.

Ziang Liu, Genggeng Zhou, Jeff He, Tobia Marcucci, Li Fei-Fei, Jiajun Wu, and Yunzhu Li. Model-based control with sparse neural dynamics. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Chenhao Lu, Xuxin Cheng, Jialong Li, Shiqi Yang, Mazeyu Ji, Chengjing Yuan, Ge Yang, Sha Yi, and Xiaolong Wang. Mobile-television: Predictive motion priors for humanoid whole-body control. In *IEEE International Conference on Robotics and Automation, ICRA 2025*, 2025. doi: 10.48550/arXiv.2412.07773. URL `http://arxiv.org/abs/2412.07773`.

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, January 2023. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000086.

Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *The Twelfth International Conference on Learning Representations*, October 2023.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28927-4 978-3-319-28929-8.

Frans A. Oliehoek, Stefan J. Witwicki, and Leslie Pack Kaelbling. A sufficient statistic for influence in structured multiagent environments. *J. Artif. Intell. Res.*, 70:789–870, 2021. doi: 10.1613/JAIR.1.12136.

Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients, May 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4292–4301. PMLR, 2018.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-03051-4.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado (eds.), *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.061.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, D. Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *ArXiv*, May 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar (eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - an Introduction*. Adaptive Computation and Machine Learning. MIT Press, 1998. ISBN 978-0-262-19398-6.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Vilamoura-Algarve, Portugal, October 2012. IEEE. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. doi: 10.1109/IROS.2012.6386109.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric D. Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440, May 2016. doi: 10.1109/ICRA.2016.7487277.

Shengchao Yan, Baohe Zhang, Yuan Zhang, Joschka Boedecker, and Wolfram Burgard. Learning continuous control with geometric regularity from robot intrinsic symmetry. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 49–55. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610949.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25476–25488. Curran Associates, Inc., 2021.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 11458–11468. PMLR, November 2020.

A Additional Method Details

A.1 Monotonic Function for the 3-Link Manipulator

Let $r_g = f_R(x, y, a) = -\|(x, y) - (\bar{x}, \bar{y})\|_2^2$ the global reward and $r_i = f_{R_i}(\theta_i, a_i) = -(\theta_i - \bar{\theta}_i)^2$ the individual rewards. We are going to show $r_g = M(r_1, r_2, r_3)$ when $(x, y) \to (\bar{x}, \bar{y})$, where M is a monotonic function. The specific format of the forward kinematics in our case is $(x, y) = f_{FK}(\theta_1, \theta_2, \theta_3) = (L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3), L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3))$.

We first represent joints θ by individual rewards:

$$r_i = -(\theta_i - \bar{\theta}_i)^2 \tag{6}$$
$$\theta_i = \bar{\theta}_i \pm \sqrt{-r_i} \tag{7}$$

Then replace all joints in the global reward by individual rewards:

$$\begin{aligned} r_e &= -\|(x, y) - (\bar{x}, \bar{y})\|_2^2 && (8)\\ &= -\|f_{FK}(\theta) - (\bar{x}, \bar{y})\|_2^2 && (9)\\ &= -\|f_{FK}(\bar{\theta}_1 \pm \sqrt{-r_1}, \bar{\theta}_2 \pm \sqrt{-r_2}, \bar{\theta}_3 \pm \sqrt{-r_3}) - (\bar{x}, \bar{y})\|_2^2 && (10)\\ &= M(r_1, r_2, r_3) && (11) \end{aligned}$$

We now have the relations between global reward and individual, we are going to show that M is monotonic when $(x, y) \to (\bar{x}, \bar{y})$ or $\theta \to \bar{\theta}$. Let $c_e = -r_e, c_i = \bar{\theta}_i \pm \sqrt{-r_i}$, and the relation becomes as follows:

$$\begin{aligned} c_e &= \|f_{FK}(c_1, c_2, c_3) - (\bar{x}, \bar{y})\|_2^2 && (12)\\ &= (L_1 \cos c_1 + L_2 \cos(c_1 + c_2) + L_3 \cos(c_1 + c_2 + c_3) - \bar{x})^2 && (13)\\ &+ (L_1 \sin c_1 + L_2 \sin(c_1 + c_2) + L_3 \sin(c_1 + c_2 + c_3) - \bar{y})^2 && (14)\\ \frac{\partial c_e}{\partial c_3} &= 2L_3[(y - \bar{y}) \cos(c_1 + c_2 + c_3) - (x - \bar{x}) \sin(c_1 + c_2 + c_3)] && (15)\\ \frac{\partial c_e}{\partial c_2} &= 2[(y - \bar{y})(L_2 \cos(c_1 + c_2) + L_3 \cos(c_1 + c_2 + c_3)) - (x - \bar{x})(L_2 \sin(c1 + c_2) + L_3 \sin(c_1 + c_2 + c_3)] && (16)\\ \frac{\partial c_e}{\partial c_1} &= 2[(y - \bar{y})(L_1 \cos c_1 + L_2 \cos(c_1 + c_2) + L_3 \cos(c_1 + c_2 + c_3)) && (17)\\ &- (x - \bar{x})(L_1 \sin c_1 + L_2 \sin(c_1 + c_2) + L_3 \sin(c_1 + c_2 + c_3)] && (18)\\ &= 2[(y - \bar{y})x - (x - \bar{x})y] = 2(y\bar{x} - x\bar{y}) && (19) \end{aligned}$$

We can focus on the case of $c_i = \bar{\theta}_i + \sqrt{-r_i}$, and then $\bar{x} - x = \sqrt{c_e} \sin(c_1 + c_2 + c_3) > 0, y - \bar{y} = \sqrt{c_e} \cos(c_1 + c_2 + c_3) > 0$. And otother cases followin the same procedure.

$$\begin{aligned} \frac{\partial c_e}{\partial c_3} &= 2L_3[\sqrt{c_e} \cos^2(c_1 + c_2 + c_3) + \sqrt{c_e} \sin^2(c_1 + c_2 + c_3)] = 2L_3\sqrt{c_e} > 0 && (20)\\ \frac{\partial c_e}{\partial c_2} &= 2[L_3\sqrt{c_e} \cos^2(c_1 + c_2 + c_3) + L_2 L_3 \cos(c_1 + c_2) \cos(c_1 + c_2 + c_3)) && (21)\\ &+ L_3\sqrt{c_e} \sin^2(c_1 + c_2 + c_3) + L_2 L_3 \sin(c1 + c_2) \sin(c_1 + c_2 + c_3)] && (22)\\ &= 2[L_3\sqrt{c_e} + L_2 L_3 \cos c_3] > 0 && (23)\\ \frac{\partial c_e}{\partial c_1} &= 2(y\bar{x} - x\bar{y}) > 0 && (24) \end{aligned}$$

In addition, we know that $\frac{\partial c_e}{\partial r_e} < 0$ and $\frac{\partial c_i}{\partial r_i} < 0$. By the transitivity of monotonicity, $\frac{\partial r_e}{\partial r_i} > 0$, so that M is a monotonic function.

A.2 Experiments for the 3-Link Manipulator

Hyperparameters of MPPI. We apply the model predictive path integral control (MPPI) (Williams et al., 2016) on both centralized MDP and fDec-MDP. In specific, the shared hyperparameters of MPPI are shown in Table 1 for fair comparison. Notably, the discount is set to 0.0 for simplicity, so that only transition and reward functions matter in our cases. For fDec-MDP, since each agent (joint) has individual transition and reward functions, the optimization can be executed independently and in parallel.

Table 1: Hyperparameters of MPPI for the 3-Link Manipulator.

Planning (MPPI / MPC)		
`iterations`	6	Number of MPPI optimization iterations per control step.
`num_samples`	50	Number of trajectory samples drawn per iteration.
`num_elites`	5	Number of top samples (elites) used to update the sampling distribution.
`horizon`	4	Planning horizon (timesteps) for MPPI (comment shows prior value 5).
`min_std`	0.05	Minimum standard deviation for the sampling noise (stability floor).
`max_std`	2.0	Maximum standard deviation allowed for sampling noise.
`temperature`	0.5	Softmax/temperature parameter used when computing weights for samples.
`discount`	0.0	Discount factor.
`freq`	10	Planning frequency, the step interval is then 0.1 seconds.
`steps`	100	Planning steps.

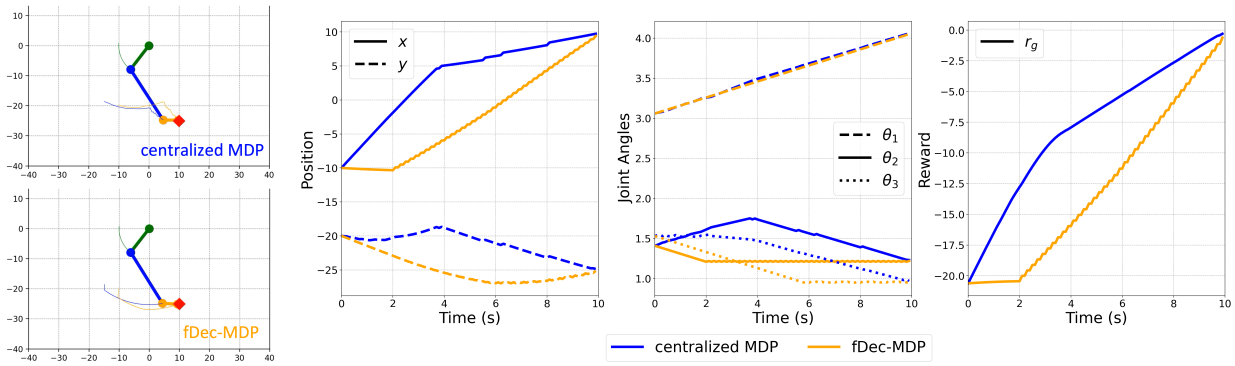


Figure 11: Planning results on both centralized MDP and fDec-MDP. **Left:** The trajectory and the final state of both methods. The trajectory of all joints are visualized in solid lines. **Middle and right:** The x-axis is the elapsed time, and the y-axis represents the end effector position, joint angles and total reward.

Results. We visualize the two planning results in Figure 11. Both methods reach the goal position in the end as shown by the reward, and planning on the fDec-MDP leads to a smoother trajectory on the end effector's positions. Besides, distributed planning on fDec-MDP drives θ_2 and θ_3 more actively for faster convergence, instead of being predominantly occupied by one dimension (θ_1). Both planning processes are executed on the same hardware and hyperparameters. The average planning time on the centralized MDP is 1.20seconds per step, while the average planning time on the fDec-MDP is 0.04 seconds per step, with around $\times$**30** speedup.

A.3 Violation on Individual Global Max

The success of decentralized planning method depends on the existence of Proposition 1 as introduced in Section 4.3, i.e. $a^*_{t:t+H} = \underset{a_{t:t+H}}{\operatorname{argmax}} \; G^H(g_t, a_{t:t+H}) = \times_{i=1}^{N} \underset{a^i_{t:t+H}}{\operatorname{argmax}} \; G^H_i(g_t, a^i_{t:t+H})$. In other words, if finding the decomposition of the global returns into the sum of the individual returns as $G^H(g_t, a_{t:t+H}) = \sum_i G^H_i(g_t, a^i_{t:t+H})$ either theoretically (as the 3-link manipulator example in Section 5) or empirically (as the MuJoCo experiments in Section 5), one can say in confidence that the decentralized planning provdes

the exact same results as the global planning method. Here, we provide a failure case that does not satisfy the above-mentioned decomposition. Assume a two-agent case with horizon $H = 1$ and constant states $g_t = g$ for simplicity, we can abbreviate the condition as $G(a) = G_1(a^1) + G_2(a^2)$. For the global return with structure: $G(a) = a^1 a^2$, it is not hard to prove that we can never find G_1, G_2 to exactly satisfy the condition $G(a) = G_1(a^1) + G_2(a^2)$. When applied to robotics, a perfect example is a robot using two arms to lift a heavy object. Only when both arms lift with equal force can the task succeed. It is difficult to find a decomposition to execute decentralized planning in this scenario even with the global state g for both agents.

A.4 Deviations from full factorization and optimality

This problem can be answered from the perspective of how the terminal value learning in our algorithm is updated in function approximation version. Basically, the update of each decentralized terminal value function $f_{Q_i}^{\theta_i}(s_t^i, a_t^i)$ with stochastic gradient descent (SGD) as the optimizer follows the equation:

$$\theta_i \leftarrow \theta_i + \alpha \left(r_t + \max_a \gamma f_{\bar{Q}}(s_{t+1}, a) - \sum_{j=1}^{N} f_{Q_j}^{\theta_j}(s_t^j, a_t^j) \right) \nabla_{\theta_i} f_{Q_i}^{\theta_i}(s_t^i, a_t^i), \tag{25}$$

where each $f_{Q_i}^{\theta_i}(s_t^i, a_t^i)$ is presumed to be parameterized by θ_i, $f_{\bar{Q}}$ functions the global terminate value, and α is the learning rate. It is obvious that the approximation error induced by the full factorization would influence the credit denoted as $r_t + \max_a \gamma f_{\bar{Q}}(s_{t+1}, a) - \sum_{j=1}^{N} f_{Q_j}^{\theta_j}(s_t^j, a_t^j)$, during the optimization process of updating θ_i. Assume this error is ϵ_t at some step t, the credit for updating parameters would be deviated by $\alpha\epsilon_t$. As a result, it will lead to total error of the resulting parameter, accumulated along the optimization trajectory, upper bounded by $m\alpha\epsilon_{\max} \max_{s_t^i, a_t^i} \nabla_{\theta_i} f_{Q_i}^{\theta_i}(s_t^i, a_t^i)$, where m is the number of updating parameters and $\epsilon_{\max} = \max_t \epsilon_t$.

A.5 Discussion on modular world models

There are also works learning structured world models in the latent space. One choice is to learn a factored world model in the latent space (Balaji et al., 2020; Lei et al., 2022), which usually aims to capture the causality of the system with attention-based architectures. However, directly using it as the transition function in the MPPI method does not incur efficient planning. Because the factored structure is still a connected graph, and the optimization space is still $\mathcal{O}(|\mathcal{A}|^H)$, in comparison with Fac-TDMPC $\mathcal{O}(\sum_{i=1}^{N} |\mathcal{A}^i|^H)$. Among the literature, Liu et al. (2023) is motivated to learn a structured transition function for further efficient planning. They first learned a sparsified neural network and then adopted mixed-integer programming (MIP) to plan on it. However, as is well known, MIP planner is usually slower than MPPI planner, and their methods are restricted to the ReLU activations in the transition functions. Our method allows arbitrary forms of transition functions.

B Additional Experimental Setups

B.1 Task Description

- `Reacher Easy`: A simple reaching task where a 3-link arm must move its end-effector to a target position with relatively slow dynamics.
- `Reacher Hard`: A more challenging version of the 3-link reaching task with faster dynamics or smaller target tolerance, requiring more precise control.
- `Hopper Hop`: A single-legged hopper must maintain balance and hop forward, testing stability and rhythmic control.
- `Fish Swim`: A simulated fish must propel itself forward and maneuver in water, involving complex body coordination.

- `Cheetah Run`: A two-legged cheetah to run while keeping stability under high-speed conditions.
- `Walker Walk`: A bipedal walker must move forward at a moderate speed, emphasizing stable gait generation.
- `Walker Run`: The walker must achieve faster locomotion while maintaining balance, increasing control difficulty.
- `Quadruped Walk`: A four-legged robot must walk steadily forward, testing coordination across multiple legs.
- `Quadruped Run`: A faster variant requiring the quadruped to run while keeping stability under high-speed conditions.
- `Humanoid Stand`: A two-legged humanoid robot must stand steadily, testing coordination across multiple joints.
- `Humanoid Walk`: A faster variant requiring the humanoid to walk while keeping stability.

B.2 Baselines

For TDMPC, we directly adopt the official source code[1] as the implementation. For 10% TDMPC, we directly set the hidden units from 512 to 66, ending up with around 10% parameters of the original model. For TAP, the original method learns a latent action space, compressing both large action space and long time horizons. To fairly compare with other methods in the following MPPI planning methods, we restrict the compression only in the action space. As a result, the method essentially becomes to map the original action space into a latent action space, empirically set as 2 dimensions. Both transition, reward and value functions are grounded on the latent state and action space. The same planning algorithm MPPI is performed on the latent action space, and mapped to the original action space with a trained decoder. The training is accomplished with an additional VAE reconstruction loss on actions.

B.3 Hyperparameters

To compare all algorithms fairly, we set the model structures and hyperparameters equally. All algorithms are trained with Adam optimizer (Kingma & Ba, 2015). The full hyperparameters are shown in Table 2. All experiments are carried out on NVIDIA GeForce RTX 2080 Ti and Pytorch 1.10.1.

C Additional Experimental Results

C.1 Online Model Distillation

We further evaluate Fac-TDMPC in an online model distillation setting, where the lightweight student model is required to continuously track the expert policy as it improves during training. In this setting, the expert (TDMPC) is not fixed but rather being optimized from scratch and in parallel, posing a more challenging scenario than standard offline distillation.

Figure 12 summarizes the results across 3 most challenging tasks (`Walker Run`, `Quadruped Run`, and `Humanoid Walk`) as well as the averaged performance. Despite the non-stationarity introduced by the evolving expert, Fac-TDMPC is able to reliably follow the learning curve of TDMPC across all tasks. This finding is particularly meaningful: it shows that an efficient distilled model can be acquired online, simultaneously with the training of the full expert model. Such a property eliminates the need for a separate post-hoc distillation phase, thereby reducing overall training time and enabling practical deployment of efficient policies in real-time learning scenarios.

The result for `Humanoid Walk` is interesting, in that Fac-TDMPC even surpasses TDMPC at the beginning, since learning a centralized model is difficult for such a large action space. But in the end, the minor performance gaps that appear might result from a slow convergence of the expert model.

[1] https://github.com/nicklashansen/tdmpc

Table 2: Hyperparameters used in experiments.

Training	**Values**	**Description**
`steps`	`10_000_00`	Total number of training steps (gradient updates).
`batch_size`	`512`	Mini-batch size used for each optimization step.
`reward_coef`	`0.5`	Weight for reward prediction / reward-loss term in the training objective.
`value_coef`	`0.1`	Weight for value / critic loss term.
`consistency_coef`	`0.0`	Weight for any model-consistency loss (if used).
`rho`	`0.5`	Mixing / interpolation coefficient used in targets or blending (implementation-specific).
`lr`	`1e-4`	Base learning rate for optimizer.
`enc_lr_scale`	`1.0`	Multiplier applied to the encoder learning rate (encoder lr = `lr` × this).
`grad_clip_norm`	`10`	Maximum gradient norm for clipping.
`tau`	`0.01`	Soft-update coefficient for target networks (EMA factor).
`discount`	`0.99`	Discount factor.
`buffer_size`	`1_000_000`	Replay buffer capacity (number of transitions).
Planning (MPPI / MPC)		
`mpc`	`true`	Whether to enable model-predictive control (MPPI) at inference time.
`iterations`	`6`	Number of MPPI optimization iterations per control step.
`num_samples`	`512`	Number of trajectory samples drawn per iteration.
`num_elites`	`64`	Number of top samples (elites) used to update the sampling distribution.
`num_pi_trajs`	`24`	Number of policy (pi) trajectories evaluated (if using policy proposals).
`horizon`	`3`	Planning horizon (timesteps) for MPPI (comment shows prior value 5).
`min_std`	`0.05`	Minimum standard deviation for the sampling noise (stability floor).
`max_std`	`2.0`	Maximum standard deviation allowed for sampling noise.
`temperature`	`0.5`	Softmax/temperature parameter used when computing weights for samples.
Architecture		
`num_enc_layers`	`2`	Number of encoder (feedforward/conv) layers.
`enc_dim`	`256`	Width (hidden units) of encoder layers.
`num_channels`	`32`	Number of channels if using convolutional layers (per layer).
`mlp_dim`	`512`	Width of MLP heads / fully-connected layers.
`latent_dim`	`50`	Dimension of learned latent state per agent (or total, per design).
`num_q`	`2`	Number of Q-networks (ensemble) used for critic.
`dropout`	`0.01`	Dropout probability used in networks.

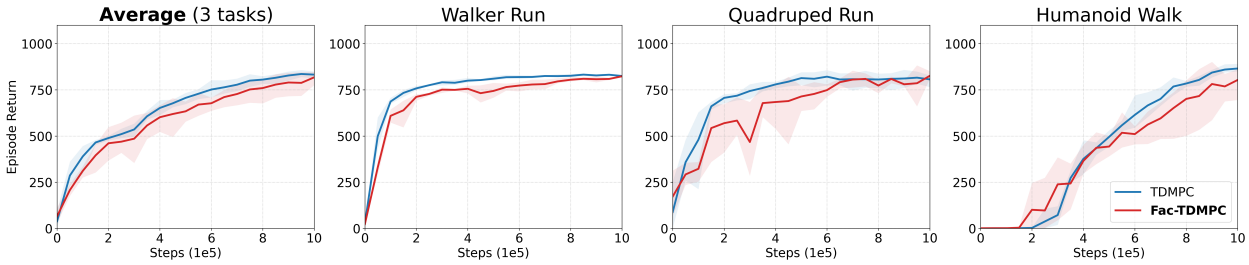


Figure 12: Online model distillation on challenging (`Walker Run`, `Quadruped Run`, and `Humanoid Walk`) tasks. Each curve shows the average episode return over three seeds, with shaded regions denoting the standard deviation.

C.2 Online Learning

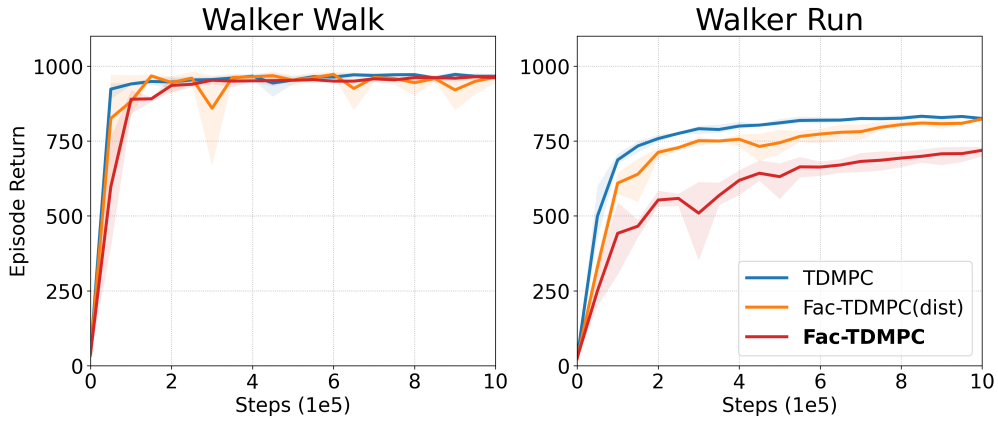


Figure 13: Online learning on two tas ks (`Walker Walk` and `Walker Run`). Each curve shows the average episode return over three seeds, with shaded regions denoting the standard deviation.

In this section, we evaluate Fac-TDMPC in an online learning setting, alongside its distillation-enhanced variant, Fac-TDMPC(dist) (see Section C.1), across two tasks of varying complexity. While the factored model is computationally more efficient, results demonstrate that for simpler tasks such as `Walker Walk`, all methods achieve comparable peak returns. However, on the more challenging `Walker Run` task, the purely online-trained Fac-TDMPC struggles to reach the optimal policy. Crucially, employing online model distillation enables the model to recover this performance gap, nearly matching the baseline's return. These results imply that factored models gain computational efficiency from a sreduced function space, yet may suffer from limited exploration in online settings—a constraint that can be effectively mitigated by distilling knowledge from a full model.

C.3 Detailed Prediction Time

Table 3: Prediction time of different models on different robots. All wall times are reported in milliseconds with average and standard deviation over 100 runs. T1 represents testing on a single-thread machine.

Methods		Reacher	Hopper	Fish	Cheetah	Walker	Quadruped	Humanoid
TDMPC	f_T	5.83 ± 0.01	5.86 ± 0.01	5.83 ± 0.01	5.63 ± 0.01	5.86 ± 0.01	5.96 ± 0.01	5.83 ± 0.07
	f_R	5.32 ± 0.01	5.34 ± 0.01	5.26 ± 0.01	5.06 ± 0.01	5.28 ± 0.01	5.34 ± 0.01	5.25 ± 0.01
	f_Q	11.73 ± 0.02	11.70 ± 0.04	11.56 ± 0.04	11.35 ± 0.05	11.71 ± 0.04	11.86 ± 0.05	11.71 ± 0.06
10% TDMPC	f_T	0.32 ± 0.01	0.33 ± 0.01	0.33 ± 0.01	0.32 ± 0.00	0.32 ± 0.01	0.32 ± 0.01	0.32 ± 0.01
	f_R	0.25 ± 0.00	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.00	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01
	f_Q	0.85 ± 0.01	0.85 ± 0.02	0.88 ± 0.03	0.88 ± 0.03	0.87 ± 0.02	0.87 ± 0.01	0.85 ± 0.03
TAP	f_T	1.78 ± 0.00	1.78 ± 0.00	1.78 ± 0.00	1.78 ± 0.00	1.78 ± 0.00	1.78 ± 0.00	1.78 ± 0.01
	f_R	1.50 ± 0.01	1.54 ± 0.01	1.56 ± 0.01	1.53 ± 0.01	1.50 ± 0.01	1.49 ± 0.00	1.56 ± 0.00
	f_Q	3.68 ± 0.02	3.81 ± 0.03	3.80 ± 0.02	3.69 ± 0.03	3.69 ± 0.02	3.67 ± 0.02	3.69 ± 0.03
Fac-TDMPC	f_T	1.25 ± 0.01	0.83 ± 0.02	0.77 ± 0.01	0.75 ± 0.03	0.72 ± 0.01	0.63 ± 0.01	0.42 ± 0.01
	f_R	1.02 ± 0.00	0.65 ± 0.01	0.57 ± 0.01	0.49 ± 0.03	0.48 ± 0.01	0.49 ± 0.00	0.32 ± 0.01
	f_Q	2.56 ± 0.03	1.71 ± 0.03	1.53 ± 0.03	1.34 ± 0.05	1.32 ± 0.03	0.96 ± 0.04	0.69 ± 0.03
Fac-TDMPC (T1)	f_T	3.71 ± 0.01	3.18 ± 0.01	3.88 ± 0.02	4.50 ± 0.02	4.46 ± 0.01	4.44 ± 0.01	4.11 ± 0.02
	f_R	2.29 ± 0.02	2.66 ± 0.02	2.82 ± 0.03	2.93 ± 0.03	2.82 ± 0.03	3.23 ± 0.02	3.29 ± 0.01
	f_Q	7.73 ± 0.03	6.86 ± 0.02	7.72 ± 0.02	7.92 ± 0.04	8.12 ± 0.03	6.69 ± 0.04	6.94 ± 0.05

This is the detailed prediction time of all baselines as in Table 3.

C.4 End-to-end Planning Time

We present the end-to-end planning time per step in Table 4, including the forward passes of the encoder, reward, value and transition functions within the MPPI iterations. For a fair comparison, the numbers of samples and iterations are set to 512 and 6 across all experiments. The testing hardware is set the same as in Section 5.4. Similarly to Table 3, Fac-TDMPC achieves a substantial reduction in per-step planning time compared to TDMPC, and the efficiency gain continues to improve with the increase in action dimensions. Its speed even matches 10% TDMPC for humanoid robots, but the control performance is much better as shown in Table 4.

Table 4: End-to-end prediction time per planning step of different models on different robots. All wall times are reported in milliseconds with average and standard deviation over 100 runs.

Methods	Reacher	Hopper	Fish	Cheetah	Walker	Quadruped	Humanoid
TDMPC	203.66 ± 0.44	203.71 ± 0.59	203.84 ± 0.32	204.13 ± 0.30	204.01 ± 0.27	204.02 ± 0.27	204.41 ± 0.83
10% TDMPC	14.59 ± 0.07	14.58 ± 0.08	14.62 ± 0.07	14.65 ± 0.06	14.65 ± 0.05	14.71 ± 0.12	14.84 ± 0.07
TAP	64.94 ± 0.05	64.97 ± 0.07	64.94 ± 0.0	64.98 ± 0.0	65.02 ± 0.14	65.06 ± 0.07	65.10 ± 0.14
Fac-TDMPC	43.86 ± 0.09	29.30 ± 0.10	26.02 ± 0.08	22.92 ± 0.09	21.90 ± 0.06	21.34 ± 0.05	14.02 ± 0.07
Fac-TDMPC (T1)	132.33 ± 0.30	114.48 ± 0.35	131.78 ± 0.42	137.57 ± 0.55	129.70 ± 0.48	149.02 ± 0.32	142.93 ± 0.77