Modeling Dynamics under Random Delays in Reinforcement Learning

Bokai Ji
Xidian University
Xi'an, China
jibokai@stu.xidian.edu.cn

Guangxia Li*
Xidian University
Xi'an, China
gxli@xidian.edu.cn

Yulong Shen
Xidian University
Xi'an, China
ylshen@mail.xidian.edu.cn

ABSTRACT

Reinforcement learning in real-world systems often encounters delays in sensing and actuation, violating the standard Markov decision process (MDP) assumptions of immediate and fully observed states. While world models offer a promising potential to solve such random-delayed MDPs by imagining undelayed environment dynamics, random actuation delays introduce uncertainty that hinders their direct application. Specifically, world models require the executed actions, rather than the issued ones, to make accurate imaginations of the current state. We present a novel analysis that distinguishes the effects of observation and action delays on world models, revealing an asymmetry that can be exploited to improve the learning process. To address the uncertainty caused by stochastic action execution delays, we propose representing imagined latent states as expected latent states-probability-weighted averages over all possible action-execution trajectories. Compared to sampling the latent state via a single possible action execution trajectory, the expected latent reduces variance in training targets and captures multiple plausible futures at inference. We instantiate our approach using DreamerV3 and validate it on the DeepMind Control Suite with visual inputs. Experimental results show that our method achieves significantly higher returns, more accurate dynamics predictions, and improved training stability across a wide range of delay settings compared to strong baselines.

KEYWORDS

Random Delay; Model-Based Reinforcement Learning

ACM Reference Format:
Bokai Ji, Guangxia Li, and Yulong Shen. 2026. Modeling Dynamics under Random Delays in Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/EAJI2382

1 INTRODUCTION

Reinforcement Learning (RL) has achieved remarkable success in solving Markov Decision Processes (MDPs). In standard MDPs, it is assumed that observations provide complete and immediate information about the current environment state, and that the environment dynamics evolve solely based on the current state and the agent's action. However, many real-world systems violate these assumptions. Delays in observation and action execution often arise due to physical or technological constraints in sensing, communication, and actuation [7, 8]. For instance, autonomous vehicles may experience sensor latency due to computational bottlenecks in real-time perception pipelines [11], while industrial manipulators often introduce deliberate actuation delays to comply with safety certification protocols [3].

RL algorithms that assume action execution and feedback struggle in such delayed environments, often leading to unstable learning and poor performance. Methods have been proposed to address this. Memoryless approaches [17] take prior knowledge of delay durations into consideration and restrict policies to only the most recent observation. Another line of work [6, 20] reformulates the delayed environment as an undelayed MDP by augmenting the observation space with recent action commands issued by the policy. Further, [4] rearranges delayed replay buffers to construct temporally consistent transitions for training. More recently, world-model-based approaches [12] have emerged that imagine the current state in a latent space by leveraging the last observed frame and the sequence of action commands issued since then.

While world-model-based approaches [2, 10, 16] achieve state-of-the-art performance under constant-delay settings, their applicability to randomly delayed environments, which are more realistic in the real world, remains limited. In the constant-delay setting, the agent knows exactly how many timesteps ago an observation was generated and when an issued action will be executed. This allows deterministic construction of imagined rollouts using world models. However, under random delays, the actual sequence of executed actions becomes uncertain, undermining both the training and inference stages of the world model. Furthermore, existing works using world models treat delays as a unified total delay (i.e., the sum of observation and action execution delays), without distinguishing between the two. While treating the delays in this way promotes a certain branch of works [1, 14, 21], we argue that this is not the best choice when working with world models—observation and action delays impact the world model in fundamentally different ways. As we will discuss in this work, observation delays only affect training, while action execution delays affect both training and inference. Exploiting this asymmetry enables better use of the world model's predictive capacity.

In this work, we begin with a detailed analysis of the challenges posed by stochastic delays for world-model-based reinforcement learning. We take DreamerV3 [10] as our foundation due to its strong performance and widespread use. We first show that observation delays can be mitigated entirely during training by relabeling transitions. In contrast, stochastic action delays lead to ambiguity

*Corresponding Author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/EAJI2382

about which actions have been executed, resulting in a mismatch between imagined and actual outcomes. This not only degrades the quality of the learned dynamics but also leads to an unreliable policy during inference.

To address this, we propose to use the expectation of the predicted latent states over all possible action execution trajectories, rather than relying on a single sampled rollout. Expected latent states reduce variance in training targets and provide a richer, more comprehensive representation of possible outcomes during inference. They allow agents to consider multiple plausible futures rather than committing to a single guess. Mathematically, the expected latent state is a weighted average over all latent trajectories, where the weight is determined by the probability of each action sequence being executed. These probabilities can be computed given the statistics of the delay duration.

Since traversing all possible trajectories is computationally intensive, we propose an efficient approximation. We merge the branches of the imagined trajectory tree every K steps to reduce the exponential complexity from $O(\dot{d}_a^T)$ to $O(\frac{T}{K}\dot{d}_a^K)$, where T is the imagination horizon and $\dot{d}_a$ is the maximum action delay. Our experiments on DeepMind Control Suite (DMC) [19] with visual inputs show that our method, despite its simplicity, significantly outperforms existing baselines in terms of returns. We further demonstrate that using expected latent states leads to lower dynamics prediction error and improved training stability compared to the world-model-based baselines. Our contributions are summarized as follows:

(1) We provide novel insights showing that observation and action execution delays affect world models differently and should be treated separately.
(2) We propose the use of expected latent states with theoretically guaranteed variance reduction in the policy gradient to handle action execution delays, and introduce a computationally efficient approximation for their computation.
(3) We validate our approach with extensive experiments on Vision DMC, demonstrating improved model accuracy, stabilized training, and higher returns than strong baselines.

2 RELATED WORKS

Delayed Markov Decision Processes (MDPs) were first introduced in [13], which studied asynchronous decision-making where the agent receives rewards with a delay relative to the time of action execution. This early formulation primarily focused on reward delays, analyzing how temporal misalignment between actions and feedback affects the optimal policy structure. The key insight was that although delayed rewards break the immediate Markov property, optimal control can still be achieved by reformulating the problem over an augmented state that accounts for delayed signals. However, this augmentation quickly becomes computationally intractable as the delay horizon grows.

Subsequent research extended this idea to constant-delay MDPs, in which both observation and action execution are delayed by a fixed number of timesteps [20]. Under this setting, the agent perceives the environment state and executes actions that are out of phase with the underlying true state dynamics. To handle this, [20] proposed to convert the constant-delay MDP into a Partially Observable MDP (POMDP) by augmenting the delayed observation with the sequence of actions taken since the last true observation. This augmentation essentially reconstructs the hidden true state from observable information, leveraging the fact that the environment dynamics are known and deterministic. Similar formulations were later refined by [6], who further analyzed the theoretical equivalence between constant-delay MDPs and finite-history POMDPs. They provided algorithms that explicitly construct the augmented belief space and use model-based planning or value iteration over this extended space. While such approaches are theoretically sound, they face severe scalability challenges: the dimensionality of the augmented state grows exponentially with the delay length, rendering exact inference and policy optimization intractable for non-trivial delays or continuous control domains.

More recent approaches attempt to circumvent this intractability by learning predictive models of the environment. Instead of expanding the state space explicitly, Karamzade et al. [12] proposed to leverage world models, latent dynamics models trained to simulate future observations, to "imagine" the undelayed state at the current timestep. Concretely, their method recursively applies the learned transition model to roll forward from the last observed frame, conditioned on the sequence of executed actions during the delay window. The resulting predicted observation serves as a proxy for the true undelayed state, allowing standard reinforcement learning algorithms to operate as if no delay existed. This idea bridges model-based RL with delayed MDPs and demonstrates substantial performance gains in constant-delay environments.

However, the effectiveness of such world-model-based methods relies critically on the assumption of deterministic delay. When delays vary stochastically across time or episodes, the agent no longer knows the exact number or sequence of actions that have taken effect since the last observation. This uncertainty in action execution timing introduces ambiguity into the model's imagined trajectories, leading to compounding prediction errors and unstable policy learning. In random-delay settings, the transition from observed to current state becomes a distribution over possible latent trajectories rather than a single deterministic rollout. Existing approaches, which depend on known delay horizons, thus fail to capture this uncertainty structure.

3 PRELIMINARIES

3.1 Reinforcement Learning with Random Observation and Action Delays

We consider a reinforcement learning setting with stochastic observation and action delays. Formally, we define a delayed partially observable Markov decision process (DPOMDP) as a tuple $\langle S, A, T, \Omega, O, D_o, D_a, \gamma \rangle$, where S is the state space, A is the action space, $T(s', r \mid s, a)$ is the transition distribution, Ω is the observation space, $O(o \mid s)$ is the observation distribution, D_o and D_a are distributions over non-negative integers representing random observation and action delays respectively, and $\gamma \in [0, 1)$ is the discount factor. Observation delay and action execution delay at each time step are denoted by random variables d_o and d_a, respectively.

Following [4], our work focuses on a certain scenario where all delays in the environment are communication delays. In other words, the delayed environment is composed of an agent $\pi(a_t|o_{t-d_o})$,

random observation delays $d_o \sim D_o \in [0, \dot{d}_o]$, random action delays $d_a \sim D_a \in [0, \dot{d}_a]$, an executor, and an undelayed underlying environment. At each time step t, the agent takes the most recently received observation o_{t-d_o} delayed by $d_o \sim D_o$, and computes an action $a_t \sim \pi(a_t \mid o_{t-d_o})$ based on this delayed input. Meanwhile, the executor executes the most recently received action a_{t-d_a}, delayed by $d_a \sim D_a$. After executing a_{t-d_a} and transitioning to the next state, the environment emits a new observation, which will be received by the agent after d_o timesteps.

3.2 Reinforcement Learning with World Models

In this section, we provide an overview of DreamerV3 [10] and the Recurrent State-Space Model (RSSM) [9], which is the core of DreamerV3. We focus on how RSSM enables imagination-based learning by modeling the latent dynamics of the environment.

3.2.1 DreamerV3 Overview. DreamerV3 is a model-based reinforcement learning algorithm that learns a world model from real environment interactions and utilizes it to perform policy learning through imagination. Instead of learning directly from the environment, DreamerV3 trains an agent inside its latent space by simulating trajectories using the learned world model. This significantly improves sample efficiency in complex environments with high-dimensional observations, such as pixel-based inputs.

Formally, DreamerV3 aims to maximize the expected return $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t]$ by optimizing a policy $\pi(a_t \mid s_t)$, where γ is the discount factor and r_t is the reward at time t. Rather than using raw observations o_t for learning the dynamics, DreamerV3 constructs and trains a latent dynamics model to approximate the transition dynamics and reward function:

$$p(o_{0:T}, a_{0:T-1}, r_{0:T-1}) \approx \prod_{t=0}^{T} p(o_t \mid s_t) p(r_t \mid s_t, a_t) p(s_{t+1} \mid s_t, a_t)$$

where s_t denotes the latent state at time t.

3.2.2 Recurrent State-Space Model (RSSM). The RSSM is a latent variable model designed to capture both stochastic and deterministic components of the environment dynamics. It consists of two types of latent variables:

- h_t: a deterministic hidden state updated via a recurrent neural network (e.g., GRU or LSTM).
- s_t: a stochastic latent state sampled from a learned distribution conditioned on h_t and s_{t-1}.

The RSSM defines the generative process as follows:

$$\begin{aligned} h_t &= f(h_{t-1}, s_{t-1}, a_{t-1}) \quad \text{(deterministic transition)} \\ s_t &\sim p(s_t \mid h_t) \quad \text{(stochastic transition)} \\ o_t &\sim p(o_t \mid h_t, s_t) \quad \text{(observation model)} \\ r_t &\sim p(r_t \mid h_t, s_t) \quad \text{(reward model)} \end{aligned}$$

During training, an encoder infers the posterior $q(s_t \mid h_t, o_t)$ using observed transitions. The learning objective maximizes the evidence lower bound (ELBO) of the sequence likelihood:

$$\begin{aligned} \mathcal{L}_{\text{ELBO}} = \sum_{t=0}^{T} \mathbb{E}_{q(s_t \mid h_t, o_t)} [\log p(o_t \mid h_t, s_t) + \log p(r_t \mid h_t, s_t)] \\ - \text{KL}[q(s_t \mid h_t, o_t) \parallel p(s_t \mid h_t)] \end{aligned} \tag{1}$$

where $p(s_t \mid h_t)$ is the prior which we aim to align with the posterior through maximizing the ELBO.

3.2.3 Imagination and Policy Learning. Once trained, the RSSM serves as a world model to perform imagination-based rollouts in the latent space. The agent samples sequences of latent states (s_t, h_t) by applying its policy $\pi(a_t \mid h_t, s_t)$:

$$\begin{aligned} a_t &\sim \pi(a_t \mid h_t, s_t) \\ h_{t+1} &= f(h_t, s_t, a_t) \\ s_{t+1} &\sim p(s_{t+1} \mid h_{t+1}) \end{aligned}$$

The imagined rollouts are used to optimize the policy via backpropagation through the differentiable world model. Typically, actor-critic methods are employed, where the critic estimates the value function over imagined trajectories, and the actor is updated to maximize expected returns. DreamerV3 adopts various regularization and normalization strategies to ensure stability and scalability across diverse control domains.

3.3 Notations

Throughout the rest of this paper, we adopt the following notational conventions: d_o and d_a are stochastic delays in observation and action execution respectively; o_t is the observation emitted by the environment at time t; a_t is the action issued by the agent at time t (not necessarily executed immediately); $\tilde{a}_t$ is the action actually executed by the environment at time t; B_a is a fixed-length buffer storing recently issued actions for tracking delay effects; (h_t, s_t) is the RSSM latent belief state at time t.

4 METHOD

In this section, we present our approach to modeling and mitigating the effects of temporal delays in world models. Our goal is to enable agents to *maintain coherent predictions and decision-making under randomly delayed observation and action executions.*

We begin by analyzing how observation and action delays affect the learning and inference processes of world models, and formalize the induced temporal inconsistency (Section 4.1). Next, we introduce the concept of state expectation that explicitly captures the uncertainty due to the random action execution delay in Section 4.2. We then describe how to obtain such state expectation in Section 4.3. Finally, we provide a theoretical analysis of why the exploitation of such state expectation can improve the learning process by variance reduction in the posterior and policy gradient.

4.1 Impact of Delays on World Models

In environments with delayed observations and action executions, world models must be adapted to account for temporal inconsistencies between what the agent perceives and what actually occurs in the environment. Observation delays introduce a gap between the actual state of the environment and the observation available to

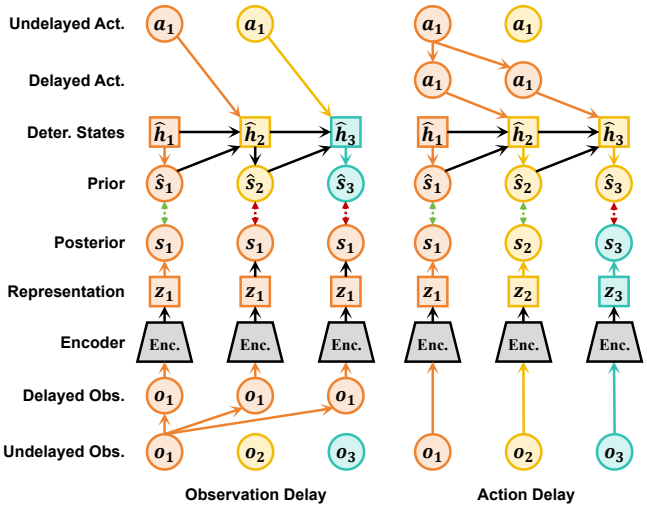


Figure 1: Illustration of the effects of observation delay and action delay at the training stage. Circles represent stochastic variables and squares deterministic variables. Different colors distinguish different timesteps. Red arrows represent mismatches between the prior and posterior latent states.

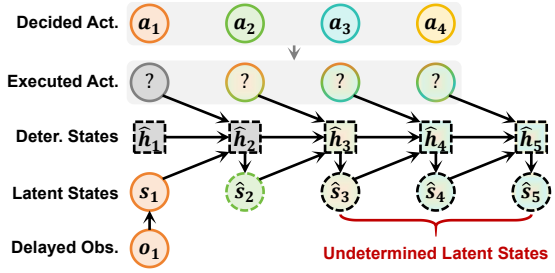


Figure 2: Effect of randomness in action execution delays. Circles represent stochastic variables and squares deterministic variables. Different colors distinguish different timesteps. Mixed colors represent uncertainty in action execution. Specifically, there is uncertainty about when an issued action will be executed.

the agent, while action delays create uncertainty about when an action will be executed after it is issued. This consequently produces uncertainty about the imagined dynamics.

Observation delays primarily affect the training process of the RSSM, as shown on the left of Figure 1. At time step t, a world model trained to predict the current observation $\hat{o}_t = M(o_{t-d_o}, [\tilde{a}_t, \tilde{a}_{t-1}, \ldots, \tilde{a}_{t-d_o}])$ based on a past observation o_{t-d_o} and a sequence of planned actions may be misaligned with the ground truth, which corresponds to o_{t-d_o} rather than o_t. This mismatch can hinder the optimization of L_{ELBO} by falsely attributing prediction errors to model inaccuracies rather than to delay-induced misalignment.

In contrast, action delays impact both training and inference. During training, the executed action sequence following a past observation o_{t-d_o} is unknown due to the stochastic nature of action delays. This results in ambiguity over which actions produced the resulting observation o_t. When the assumed action sequence deviates from what was actually executed, the resulting latent rollouts become semantically inconsistent with the ground truth, introducing variance and hindering model convergence.

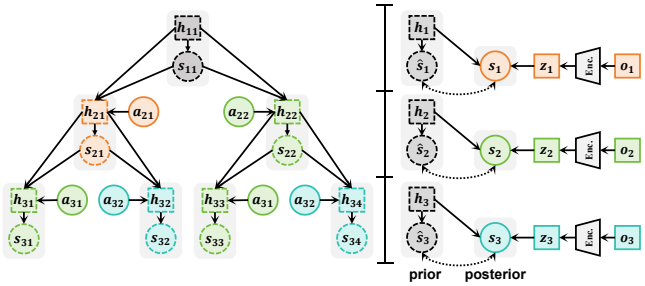


Figure 3: Underlying rollout process and dynamics loss computation. The left side shows the exhaustive rollout of the possible action execution trajectories. Circles represent stochastic variables and squares deterministic variables. In this example, we assume $d_a = 2$, which means there are 2 possible actions to be executed at each time step. The right side illustrates the dynamics loss computation. Gray squares and circles represent expected latent states merged from all possibilities in the same layer of the rollout tree.

At inference time, world models guide decision making by simulating future trajectories. Uncertainty about which actions have been executed can cause large divergence between imagined and actual states, leading to unreliable predictions and suboptimal policy behavior. This makes handling action delays critical for maintaining robust decision making in model-based RL.

4.2 State Expectation

Building on the insight that observation and action delays have different effects, we propose a method tailored to their unique impacts. We remove observation delays from training by relabeling the replay buffer to align observations with their correct timesteps. However, because action execution cannot be rearranged during inference, it is crucial for the agent to directly model its stochasticity during both training and rollout.

Instead of sampling a single executed action sequence, we define the *state expectation*—the expected latent state over all possible action execution trajectories. Let τ denote a trajectory and $p(\tau)$ its probability. We define the expected latent features and state as:

$$\bar{h} = \sum_{\tau} p(\tau) h_\tau \tag{2}$$

$$\bar{s} = p_\theta(\bar{s}|\bar{h}) \tag{3}$$

Here, h_τ and s_τ are the latent features and stochastic states along trajectory τ, and p_θ is the stochastic transition model in RSSM.

For an action assumed to be executed at time k, denoted $\tilde{a}_k$, the trajectory probability is:

$$p(\tau) = \prod_{k=1}^{T} p_{\text{exe}}(\tilde{a}_k), \tag{4}$$

which is the product of the action execution probability $p_{\text{exe}}(\tilde{a}_k)$ at each imagination step. Latent states along τ can be obtained

following the original imagination process of DreamerV3, where at each step k

$$h_{k+1} = f_\psi(h_k, s_k, \tilde{a}_k) \tag{5}$$

$$s_{k+1} \sim p_\theta(s_{k+1}|h_{k+1}) \tag{6}$$

where $f_\psi(\cdot)$ represents the recurrent network in the RSSM.

Employing state expectation to handle stochastic action delays yields several notable benefits. First, since state expectation integrates over all possible action execution trajectories weighted by their respective probabilities, it provides a smoother and more stable training signal. This contrasts with using a single sampled trajectory, which introduces high variance due to the stochasticity of delays. Moreover, the expected state $\bar{s}$ can be interpreted as a weighted summary of all plausible future states under the delay distribution. This forms a more informative latent representation compared to individual samples. Such a representation benefits downstream components of the agent, including value estimation and policy learning, as it captures the full uncertainty arising from the action execution process. Finally, in model-based reinforcement learning, decision making heavily relies on the accuracy of imagined rollouts. State expectation ensures that these imagined rollouts account for delay-induced uncertainty, which leads to more reliable value targets and gradient signals for the actor and critic. Consequently, the agent is less likely to make poor decisions due to misleading or overly optimistic rollouts.

In summary, state expectation serves as a principled mechanism to aggregate over delay-induced uncertainty in both training and inference. This improves the statistical efficiency of learning and enhances the robustness of the agent's behavior in stochastic environments with action execution delays.

4.3 Estimation of Execution Probability

Trajectory probability $p(\tau)$ depends on the action execution probability at each imagination step, according to Equation (4). In this section, we discuss the estimation of the action execution probability in detail. At each imagination step, only actions in the action buffer B_a have the chance to be executed. Thus, rather than defining $p_{\text{exe}}(a)$ as the probability of executing an action, we defining $p_{\text{exe}}(i)$ as the probability of executing the i-th action in an action buffer B_a of size $d_a^{\max}$. We denote the i-th action in B_a as $B_a(i)$.

Also, we define the probability that the executor has already received $B_a(i)$ as $p_{\text{rec}}(i)$. Because the executor always executes the newest action received, we can draw the relationship that at any time step as

$$p_{\text{exe}}(i) = p_{\text{rec}}(i) \cdot \prod_{i'>i} (1 - p_{\text{rec}}(i')) \tag{7}$$

Intuitively, an action can only be executed when it is received by the executor and all newer actions have not yet been received.

According to Equation (7), we can concretely get the estimation of $p_{\text{exe}}(i)$ once we have the estimation of $p_{\text{rec}}(i)$. $p_{\text{rec}}(i)$ is closely related to the action delay distribution D_a, where

$$p_{\text{rec}}(i) = p(d_a = i) \tag{8}$$

For example, if the probability of $d_a = 3$ is 0.5, it means the probability of the action being received after three timesteps is 0.5, while the action should be the third action in B_a after three timesteps.

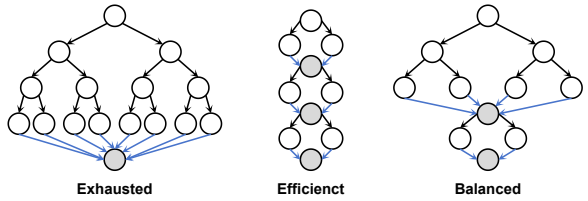


Figure 4: Merging strategies during the underlying rollout process. Blue lines indicate merging all possible latent states into an expected latent state and setting it as a new starting point of the underlying rollout process. Gray nodes represent the expected latent states.

With all the above analysis in mind, we find that we only have to estimate a statistic of d_a to get a certain estimation of $p(\tau)$. DreamerV3 starts by pre-filling the replay buffer with a random policy. The statistic of d_a can then be easily obtained from the replay buffer by

$$p(d_a = k) = \frac{N(k)}{N} \tag{9}$$

where $N(k)$ is the count of occurrences where $d_a = k$ and N is the total number of samples. This statistic can be updated in an incremental manner during both training and inference stages.

4.4 Dynamic Learning with State Expectation

We here describe the dynamic learning process with our proposed state expectation. As shown on the left side in Figure 3, we roll out all possible action execution trajectories, where k steps of imagination result in $d_a^{\max k}$ possible latent states. At each step of imagination, we compute the state expectation at each step of the rollout latent states and regard the state expectation as the prior in the KL divergence loss in RSSM. The posterior given the undelayed observation o_t is obtained in the same manner as in the original RSSM. Finally, we optimize the RSSM with Equation (1).

It is clear that with the growth of the imagination length, the number of possible trajectories grows exponentially. A simple approach alleviating such computation and memory inefficiency is to take the state expectation as a re-initial state of the imagination at each step, reducing the complexity to $O(T)$ with T steps of imagination. However, this excessively loses multi-modal information of the dynamics and may result in poor performance in the downstream tasks. We propose a balanced method where we take the state expectation every $K < T$ steps, as shown in the "Balanced" approach in Figure 4. This allows a performance-efficiency balance.

4.5 Theoretical Analysis

In this section, we provide theoretical justification for using expected latent states to handle stochastic action delays in world-model-based reinforcement learning.

Variance Reduction during Training. Let $\mathcal{L}_{\text{dyn}}(\tau)$ denote the dynamics reconstruction loss under a specific action execution trajectory τ. World models optimize the expectation of this loss using Monte Carlo (MC) sampling

$$\mathbb{E}_{\tau\sim p(\tau)}[\mathcal{L}_{\text{dyn}}(\tau)] \approx \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{\text{dyn}}(\tau_i)$$

where N sampled trajectories τ_i are used for training. However, because $\mathcal{L}_{\text{dyn}}(\tau)$ is highly nonlinear with respect to τ, the variance of this estimator is proportional to $\text{Var}[\mathcal{L}_{\text{dyn}}(\tau)]$, which increases rapidly when delay-induced stochasticity grows.

By using the expected latent state $\bar{s}$, we instead train the world model with

$$\mathcal{L}_{\text{exp}} = \mathcal{L}_{\text{dyn}}(\bar{s})$$

According to the law of total variance, this effectively replaces the stochastic latent variable s_τ by its expectation $\bar{s} = \mathbb{E}_\tau[s_\tau]$, removing the within-trajectory variance term:

$$\text{Var}(s_\tau) = \mathbb{E}_\tau[\text{Var}(s_\tau|\tau)] + \text{Var}_\tau[\mathbb{E}(s_\tau|\tau)]$$

Thus, $\bar{s}$ removes the first term, yielding a lower-variance estimator for the same expected loss. This stabilizes the training of the transition model and allows faster convergence, especially under high-delay uncertainty.

Connection to Bayesian and Ensemble Inference. From a probabilistic viewpoint, the expected latent state

$$\bar{s}_t \;=\; \mathbb{E}_{\tau\sim p(\tau|o_{t-d_o},[\tilde{a}_t,\tilde{a}_{t-1},\ldots,\tilde{a}_{t-d_o}])}\big[\, s_t^{(\tau)}\big]$$

is simply the posterior mean of the latent-state distribution marginalized over action-execution uncertainty. The posterior mean has two immediately useful properties:

1) *Minimum mean-square error.* For the quadratic loss the Bayes-optimal point estimator is the posterior mean [5]. Concretely, for any estimator $\hat{s}$ measurable,

$$\hat{s}^* \;=\; \arg\min_{\hat{s}}\; \mathbb{E}\big[\|s_t^{(\tau)} - \hat{s}\|^2\big] \quad\Longrightarrow\quad \hat{s}^* = \mathbb{E}[s_t^{(\tau)}] = \bar{s}_t. \tag{10}$$

2)*Variance reduction by conditioning (Rao–Blackwell type argument).* Let $s = s_t^{(\tau)}$ be the single-trajectory latent (the "raw" MC estimator), and let $\mathcal{G}$ be the σ-algebra corresponding to the information we condition on (the most recent observation and the action sequence since then). The law of total variance gives

$$\text{Var}(s) = \mathbb{E}\big[\text{Var}(s \mid \mathcal{G})\big] + \text{Var}\big(\mathbb{E}[s \mid \mathcal{G}]\big). \tag{11}$$

Thus the Rao–Blackwellized estimator $\mathbb{E}[s \mid \mathcal{G}] = \bar{s}_t$ satisfies

$$\text{Var}(\bar{s}_t) \;=\; \text{Var}\big(\mathbb{E}[s \mid \mathcal{G}]\big) \;\le\; \text{Var}(s), \tag{12}$$

indicating that conditioning (marginalizing over τ) cannot increase variance and typically reduces it. This formalizes why $\bar{s}_t$ gives a more stable training target than a single-sampled $s_t^{(\tau)}$ and draws the same conclusion as we analyzed from an optimization perspective.

If the downstream predictor/decoder $g(\cdot)$ (for example, the observation decoder, reward model, or critic) is *linear* then

$$g(\bar{s}_t) = g\big(\mathbb{E}[s]\big) = \mathbb{E}[g(s)], \tag{13}$$

so $\bar{s}_t$ reproduces the ensemble average exactly — there is no bias introduced by replacing the ensemble by the mean. For nonlinear $g(\cdot)$, Jensen's inequality implies a gap, but this gap can be quantified. Using a second-order Taylor expansion of g around $\bar{s}_t$,

$$\mathbb{E}[g(s)] = g(\bar{s}_t) + \tfrac{1}{2}\,\mathbb{E}\big[(s-\bar{s}_t)^\top H_g(\xi)\,(s-\bar{s}_t)\big], \tag{14}$$

for some ξ on the segment between s and $\bar{s}_t$, where H_g is the Hessian of g. If $\|H_g(\cdot)\|_2 \le M$ on the domain of interest, then

$$\|\,\mathbb{E}[g(s)] - g(\bar{s}_t)\,\| \;\le\; \tfrac{M}{2}\,\mathbb{E}\big[\|s-\bar{s}_t\|^2\big]. \tag{15}$$

Therefore the bias introduced by using $g(\bar{s}_t)$ instead of $\mathbb{E}[g(s)]$ scales with the latent variance and the curvature of $g(\cdot)$. Practically, when 1) the decoder/critic is locally nearly linear, or 2) $\text{Var}(s)$ is small, the approximation error is negligible. Thus $\bar{s}_t$ is a principled, low-variance surrogate for the ensemble mean; any remaining bias is controllable and vanishes as the latent variance or decoder curvature decreases.

Improved Policy-Gradient Estimation. Finally, we make explicit how reducing latent variance via $\bar{s}_t$ propagates to lower-variance policy-gradient updates and thus more stable training and faster convergence. For clarity, we show the argument on a single-step policy-gradient estimator; the multi-step case follows by the same reasoning with extra indices.

Let the per-timestep stochastic gradient estimator be

$$g(\theta) = \nabla_\theta \log \pi_\theta(a \mid s)\cdot A, \quad A = G - V(s), \tag{16}$$

where G is a MC estimate of the return and $V(s)$ the baseline/critic. For a scalar random variable A with $\mathbb{E}[A] = 0$ and a vector $X = \nabla_\theta \log \pi_\theta(a \mid s)$, by the Cauchy–Schwarz inequality,

$$\text{Var}\big(XA\big) \;\le\; \mathbb{E}\big[\|X\|^2\big]\;\cdot \text{Var}(A). \tag{17}$$

Thus, reducing the variance of advantage estimation $\text{Var}(A)$ directly reduces an upper bound on the variance of gradient estimation $\text{Var}(g(\theta))$ [18]. The advantage A satisfies

$$\text{Var}(A) \;=\; \text{Var}(G - V(s)) \;\le\; 2\,\text{Var}(G) + 2\,\text{Var}\big(V(s)\big). \tag{18}$$

If the critic $V(\cdot)$ is L_V-Lipschitz in the latent s (i.e. $|V(s_1) - V(s_2)| \le L_V\|s_1 - s_2\|$), then

$$\text{Var}\big(V(s)\big) \le L_V^2\;\text{Var}(s). \tag{19}$$

Combining the inequalities we obtain the upper bound

$$\text{Var}\big(g(\theta)\big) \;\le\; \mathbb{E}\big[\|\nabla_\theta \log \pi\|^2\big]\;\big(2\text{Var}(G) + 2L_V^2\text{Var}(s)\big). \tag{20}$$

Hence any reduction in $\text{Var}(s)$ yields a proportional reduction in this upper bound on the gradient variance.

Because $\bar{s}_t$ is the Rao-Blackwellized latent as we discussed previously, it satisfies $\text{Var}(\bar{s}_t) \le \text{Var}(s_t^{(\tau)})$. Substituting $\bar{s}_t$ into the critic reduces the second term in the bound above:

$$\text{Var}\big(V(\bar{s}_t)\big) \le L_V^2\;\text{Var}(\bar{s}_t) \le L_V^2\;\text{Var}(s_t^{(\tau)}). \tag{21}$$

Consequently, the upper bound on the gradient variance drops by

$$\Delta \le 2\,\mathbb{E}\big[\|\nabla_\theta \log \pi\|^2\big]\;L_V^2\;\big(\text{Var}(s_t^{(\tau)}) - \text{Var}(\bar{s}_t)\big), \tag{22}$$

which is nonnegative. Thus $\bar{s}_t$ provably reduces the upper bound on gradient variance under the stated regularity conditions.

5 EXPERIMENT

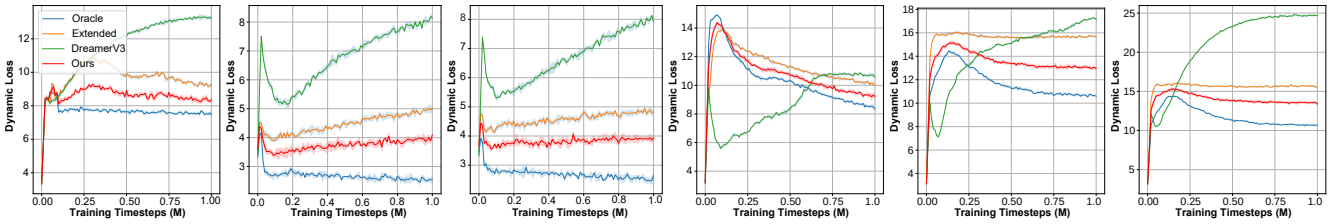
In this section, we compare DreamerV3 augmented by our proposed state expectation with a wide range of baselines in terms of 1) the overall performance; 2) the dynamic loss which directly reflects how the state expectation improves the dynamic learning; and 3) the gradient variance during the learning process, demonstrating our method successfully stabilizes the training.

5.1 Evaluation Protocol

We evaluate our method on six tasks from the Vision DMC [19], using the stochastic delay wrapper from [4] to inject random observation and action delays. For each episode, observation and action delays are independently sampled from predefined distributions, and the environment delivers the most recent available signal at

Table 1: Episodic return across the DMC environments and varies action delays. D-TRPO is trained for 5×10^6 timesteps and other methods are trained for 1×10^6 timesteps.

Method	Finger-Spin			Finger-Turn(Easy)			Finger-Turn(Hard)		
	2	4	8	2	4	8	2	4	8
DreamerV3	121 ± 14	82 ± 16	55 ± 14	678 ± 22	483 ± 26	239 ± 43	637 ± 24	489 ± 25	244 ± 32
Extended	133 ± 13	99 ± 17	67 ± 13	746 ± 23	581 ± 28	287 ± 40	701 ± 26	587 ± 27	293 ± 30
Latent	146 ± 15	95 ± 14	64 ± 15	813 ± 24	556 ± 24	275 ± 46	765 ± 22	563 ± 23	281 ± 35
DCAC	128 ± 13	87 ± 18	58 ± 16	712 ± 20	509 ± 29	252 ± 41	670 ± 21	514 ± 27	257 ± 34
Memoryless	126 ± 12	85 ± 15	59 ± 13	714 ± 25	504 ± 27	250 ± 45	672 ± 26	516 ± 22	256 ± 29
D-TRPO	129 ± 16	88 ± 14	57 ± 15	709 ± 21	511 ± 24	249 ± 39	668 ± 23	519 ± 28	258 ± 31
Ours ($K = 1$)	134 ± 15	99 ± 15	67 ± 16	747 ± 25	580 ± 24	288 ± 45	702 ± 26	588 ± 23	295 ± 35
Ours ($K = 2$)	140 ± 13	104 ± 17	70 ± 13	782 ± 20	606 ± 29	300 ± 41	735 ± 22	613 ± 27	307 ± 30
Ours ($K = 4$)	152 ± 16	108 ± 15	73 ± 15	850 ± 23	630 ± 28	313 ± 46	798 ± 25	638 ± 22	319 ± 35
Ours ($K = 6$)	154 ± 14	109 ± 16	74 ± 14	860 ± 24	637 ± 25	316 ± 40	807 ± 23	645 ± 26	323 ± 31
Method	**Walker-Run**			**Walker-Stand**			**Walker-Walk**		
	2	4	8	2	4	8	2	4	8
DreamerV3	112 ± 44	89 ± 34	57 ± 37	651 ± 32	402 ± 27	298 ± 25	418 ± 32	259 ± 12	120 ± 45
Extended	135 ± 40	107 ± 36	69 ± 41	782 ± 29	483 ± 30	359 ± 28	502 ± 35	311 ± 11	145 ± 49
Latent	129 ± 48	103 ± 31	66 ± 34	750 ± 35	463 ± 25	343 ± 23	481 ± 30	299 ± 13	139 ± 41
DCAC	118 ± 41	94 ± 37	61 ± 33	683 ± 30	423 ± 29	314 ± 27	440 ± 29	273 ± 11	127 ± 44
Memoryless	115 ± 47	92 ± 32	59 ± 39	689 ± 34	417 ± 25	309 ± 22	437 ± 34	270 ± 13	125 ± 47
D-TRPO	120 ± 43	95 ± 36	60 ± 40	678 ± 29	428 ± 24	316 ± 26	434 ± 33	269 ± 14	126 ± 43
Ours ($K = 1$)	135 ± 46	108 ± 37	69 ± 40	782 ± 35	483 ± 25	358 ± 27	504 ± 28	312 ± 13	145 ± 42
Ours ($K = 2$)	141 ± 41	112 ± 30	72 ± 34	815 ± 31	503 ± 29	374 ± 24	524 ± 35	325 ± 11	151 ± 44
Ours ($K = 4$)	146 ± 48	116 ± 36	75 ± 41	847 ± 29	523 ± 24	388 ± 26	544 ± 30	338 ± 13	157 ± 49
Ours ($K = 6$)	148 ± 43	117 ± 38	76 ± 35	856 ± 34	529 ± 26	392 ± 23	550 ± 33	342 ± 14	159 ± 47



(a) FingerSpin (b) FingerTurn(Easy) (c) FingerTurn(Hard) (d) WalkerRun (e) WalkerStand (f) WalkerWalk

Figure 5: Comparison of the dynamic loss across different tasks during training. Oracle denotes an idealized setting with no delays, which can be seen as the lower bound of the dynamic loss. Across all tasks, our approach consistently achieves lower dynamic loss than other baselines.

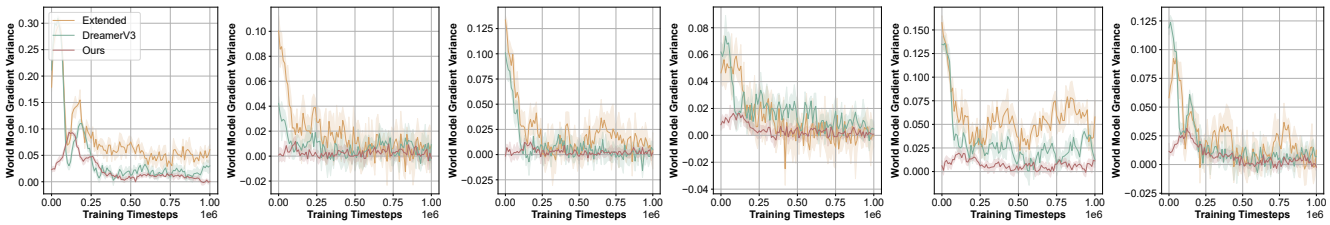
each timestep. We use a uniform distribution $U[0, \dot{d}_a]$ for all evaluations. To isolate the impact of action delays, we fix the maximum observation delay to $\dot{d}_o = 8$ and vary the maximum action delay $\dot{d}_a \in \{2, 4, 8\}$. Our method incorporates expected latent states with varying merge frequencies $K \in \{1, 2, 4, 6\}$.

Baselines include both model-based and model-free approaches. Model-based baselines are: (1) *Extended*, which augments the observation with $\dot{d}_o + \dot{d}_a$ past actions but does not use sequential imagination; and (2) *Latent*, which maintains the original observation space but uses imagined rollouts for inference. Model-free baselines include *DCAC* [4], *D-TPRO* [15], and *Memoryless* [17], where the former two methods are trained using action-augmented observations and the *Memoryless* uses unaugmented observations.

For our proposed method, we integrate it with DreamerV3 and also leverage the augmented observation space. The merging frequency K can also impact the performance of our method. We evaluated the performance with $K \in \{1, 2, 4, 6\}$ for a comprehensive analysis. Each configuration is run with five random seeds.

5.2 Main Results

Table 1 demonstrates the performance comparison in terms of returns, in where DreamerV3 is the direct appliance to the random



(a) FingerSpin (b) FingerTurn(Easy) (c) FingerTurn(Hard) (d) WalkerRun (e) WalkerStand (f) WalkerWalk

Figure 6: Comparison of the gradient variance across different tasks during training. Our method consistently achieves lower and more stable gradient variance throughout training, indicating improved optimization stability.

delayed scenario (i.e., elimination of observation delays, no extension of observation space, and no extra imagination at inference stage) and Oracle stands for the performance of DreamerV3 in undelayed settings. We first observed a significant advantage of the model-based methods compared to the model-free methods. This is because model-free methods are trained end-to-end and are unaware of the dynamics and uncertainty of the environment. Our approach outperforms other model-based methods across all six tasks when $K \geq 2$. Performance improves as K increases from 1 to 4, with only marginal gains beyond $K = 4$, indicating diminishing incremental returns relative to the additional computation. This opens avenues for more sophisticated adaptive merging strategies that allocate more resources to harder cases and less to easier cases. The development of such adaptive strategies, which allocate resources commensurate with task difficulty, is left for future investigation.

5.3 Dynamic Loss

Since our approach mainly focuses on enhancing the ability of handling random delays of the world model, we compare the dynamic loss learning curve against other model-based methods. As shown in Figure 5, dynamic loss of the vanilla DreamerV3 increases as training progresses in all of the six environments. This is because as the training progresses, the agent's policy improves and explores a larger portion of the observation space. This increases the difficulty of the dynamics learning. As discussed in the Section 4.1, stochastically delayed environments confuse the vanilla DreamerV3 due to the mismatch between the prior and posterior.

Extended approach reduced the dynamic loss by providing historical actions within the total delay window. This requires the model to learn the action execution probability from the noisy data. Moreover, since the distribution of the delay is independent of the observation and issued actions, the extended observation does not carry information about the execution distributions, making the learning procedure even harder.

The dynamic loss curve for our method is shown for the case where $K = 4$. Although there is still a gap comparing with the Oracle, our method achieves the lowest error compared to the baselines. Our method continually optimizes the dynamics error along with the training process, while the baselines exhibit an increase in the dynamic loss in most environments. This implies that the expected latent state successfully fuses the possible future and thus serves as a more informative representation.

5.4 Variance Reduction

To further assess the effectiveness of our approach, we evaluate the variance of the gradients during the learning of the dynamics model. Gradient variance serves as a direct indicator of training stability, reflecting how consistently the model updates its parameters over time. In delayed environments, uncertainty in action execution propagates through the temporal dependencies of the model, amplifying stochasticity in gradient estimates and often leading to unstable convergence. Oracle results are excluded from this analysis since, by definition, the Oracle model operates under fully deterministic execution and thus exhibits negligible variance.

As illustrated in Figure 6, our method achieves a remarkably lower gradient variance throughout training compared with all baselines. This empirical result substantiates our theoretical claim in Section 4.2. By replacing random single-trajectory sampling with the expected latent state, the dynamics model effectively performs a MC marginalization over all possible action execution paths. This process yields an unbiased estimator of the latent dynamics while significantly reducing variance in the gradient updates. Consequently, the learning process becomes smoother and more robust to stochastic delays, leading to faster convergence.

6 CONCLUSION

We studied reinforcement learning under random observation and action delays, revealing that these two types of delays affect world models in fundamentally different ways. Observation delays can be fully mitigated by transition relabeling, while stochastic action delays introduce uncertainty that persists during inference. To address this, we proposed the expected latent state, which marginalizes over all possible action execution trajectories to provide a variance-reduced, unbiased representation of the latent dynamics. Theoretically, the expected latent state corresponds to the posterior mean estimator that minimizes uncertainty under quadratic loss. Empirical evaluations on the Vision DMC benchmark demonstrated that incorporating expected latent states consistently improves both prediction accuracy and policy performance across diverse stochastic delay settings. Our method stabilizes training, achieves lower dynamics prediction errors, and approaches the performance of the undelayed oracle even under severe random delays. These results highlight the importance of treating delays asymmetrically and pave the way for more robust world model applications in realistic delayed environments.

ACKNOWLEDGMENTS

This work was supported by the Major Research Plan of the National Natural Science Foundation of China (Grant No. 92267204).

REFERENCES

[1] Eitan Altman and Philippe Nain. 1992. Closed-Loop Control with Delayed Information. In *Proceedings of the 1992 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, Newport, Rhode Island, USA, June 1-5, 1992*, Michael K. Molloy and Blaine D. Gaither (Eds.).

[2] Sriharshitha Ayyalasomayajula, Banafsheh Rekabdar, and Christos Mousas. 2023. Deep Hierarchical Variational Autoencoders for World Models in Reinforcement Learning. In *2023 Fifth International Conference on Transdisciplinary AI (TransAI), Laguna Hills, CA, USA, September 25-27, 2023.*

[3] Jaemin Baek, Soonwan Cho, and Soohee Han. 2018. Practical Time-Delay Control With Adaptive Gains for Trajectory Tracking of Robot Manipulators. *IEEE Trans. Ind. Electron.* (2018), 5682–5692.

[4] Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher J. Pal, and Jonathan Binas. 2021. Reinforcement Learning with Random Delays. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.*

[5] George Casella and Roger L Berger. 2002. *Statistical inference.* Vol. 2. Duxbury Pacific Grove, CA.

[6] Esther Derman, Gal Dalal, and Shie Mannor. 2021. Acting in Delayed Environments with Non-Stationary Markov Policies. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.*

[7] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Mach. Learn.* (2021).

[8] Yuan Ge, Qigong Chen, Ming Jiang, and Yiqing Huang. 2014. SCHMM-based modeling and prediction of random delays in networked control systems. *J. Frankl. Inst.* (2014), 2430–2453.

[9] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA.*

[10] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. 2023. Mastering Diverse Domains through World Models. *CoRR* (2023).

[11] Xiangkun He, Baichuan Lou, Haohan Yang, and Chen Lv. 2023. Robust Decision Making for Autonomous Vehicles at Highway On-Ramps: A Constrained Adversarial Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* (2023), 4103–4113.

[12] Armin Karamzade, Kyungmin Kim, Montek Kalsi, and Roy Fox. 2024. Reinforcement Learning from Delayed Observations via World Models. *RLJ* (2024).

[13] Konstantinos V. Katsikopoulos and Sascha E. Engelbrecht. 2003. Markov decision processes with delays and asynchronous cost collection. *IEEE Trans. Autom. Control.* (2003).

[14] Pierre Liotet, Davide Maran, Lorenzo Bisi, and Marcello Restelli. 2022. Delayed Reinforcement Learning by Imitation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA.*

[15] Pierre Liotet, Erick Venneri, and Marcello Restelli. 2021. Learning a Belief Representation for Delayed Reinforcement Learning. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021.*

[16] Pietro Novelli, Marco Pratticò, Massimiliano Pontil, and Carlo Ciliberto. 2024. Operator World Models for Reinforcement Learning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.*

[17] Erik Schuitema, Lucian Busoniu, Robert Babuska, and Pieter Jonker. 2010. Control delay in Reinforcement Learning for real-time dynamic systems: A memoryless approach. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan.*

[18] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning - an introduction, 2nd Edition.* MIT Press. http://www.incompleteideas.net/book/the-book-2nd.html

[19] Yuval Tassa, Yotam Doron, Alistair Muldal, et al. 2018. DeepMind Control Suite. *CoRR* (2018).

[20] Thomas J. Walsh, Ali Nouri, Lihong Li, and Michael L. Littman. 2009. Learning and planning in environments with delayed feedback. *Auton. Agents Multi Agent Syst.* (2009).

[21] William Wei Wang, Dongqi Han, Xufang Luo, and Dongsheng Li. 2024. Addressing Signal Delay in Deep Reinforcement Learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.*