ACCELERATING MODEL-BASED REINFORCEMENT LEARNING WITH STATE-SPACE WORLD MODELS

Maria Krinner,* Elie Aljalbout,* Angel Romero & Davide Scaramuzza
Robotics and Perception Group, University of Zurich, Switzerland

ABSTRACT

Model-based RL (MBRL) simultaneously learns a policy and a world model that captures the environment's dynamics and rewards. The world model can either be used for planning, for data collection, or to provide first-order policy gradients for training. Leveraging a world model significantly improves sample efficiency compared to model-free RL. However, training a world model alongside the policy increases the computational complexity, leading to longer training times that are often intractable for complex real-world scenarios. In this work, we propose a new method for accelerating model-based RL using state-space world models. Our approach leverages state-space models (SSMs) to parallelize the training of the dynamics model, which is typically the main computational bottleneck. Additionally, we propose an architecture that provides privileged information to the world model during training, which is particularly relevant for partially observable environments. We evaluate our method in several real-world agile quadrotor flight tasks, involving complex dynamics, for both fully and partially observable environments. We demonstrate a significant speedup, reducing the world model training time by up to 10 times, and the overall MBRL training time by up to 4 times. This benefit comes without compromising performance, as our method achieves similar sample efficiency and task rewards to state-of-the-art MBRL methods.

1 INTRODUCTION

Robot learning has proven to be a very successful paradigm for acquiring complex robotic skills. Recent work has demonstrated the applicability of reinforcement learning (RL) approaches to manipulation Andrychowicz et al. (2020); Handa et al. (2023); Aljalbout et al. (2024a); O'Neill et al. (2024), locomotion Tan et al. (2018); Hwangbo et al. (2019), and aerial robotics Kaufmann et al. (2023); Song et al. (2023). For such methods to succeed, they either require encoding the task into a reward function or access to a dataset of expert demonstrations. Learning control policies reduces the barrier to skill acquisition in robotics, requiring less human interventions and efforts to equip a robot with a new skill. In addition, RL methods impose minimal constraints in the design of reward functions. As a result, they allow for greater flexibility in the design of robotic systems, which can lead to outperforming classical control approaches Song et al. (2023); Lee et al. (2020a). Despite their success, one major challenge in robot learning is the need for large amounts of physical interaction data, which can be very expensive and challenging to obtain. In addition, RL suffers from high-variance gradients, often leading to unstable training and raising the need for even more data samples. This problem is further exacerbated when learning vision-based control policies due to the high-dimensionality

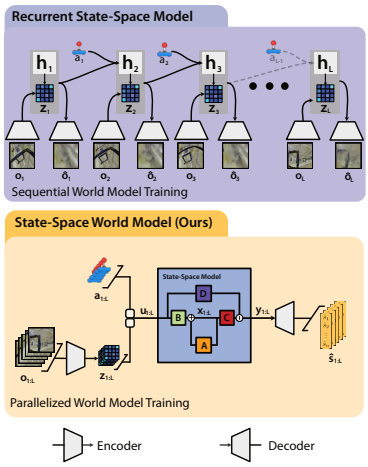

Figure 1: Most model-based RL methods employ recurrent state-space models (RSSMs) as the world model backbone, which are slow in training due to the sequential nature of RNNs. We leverage state-space models (SSMs) to parallelize the sequence dimension of the world model.

*Equal Contribution.

of image-based observations. To alleviate this problem, multiple solutions have been proposed to leverage prior knowledge in RL. These solutions range from pretraining the policy or parts of it using either imitation learning Xing et al. (2024) or self-supervised learning objectives Yarats et al. (2021); Aljalbout et al. (2021); Lee et al. (2020b), to embedding inductive biases into RL policy architectures and training pipelines to reduce the policy complexity Lutter et al. (2021); Funk et al. (2022). Model-based RL has emerged as a promising alternative to improve sample efficiency in comparison to model-free RL Deisenroth et al. (2011); Heess et al. (2015); Hafner et al. (2020); Aljalbout et al. (2024b).

However, model-based RL methods are typically slower in training than their model-free counterparts because they additionally train the world model (WM), which can be slow due to the sequential nature of the dynamics. This aspect limits the applicability of MBRL, particularly for cases where fast training is desirable.

In this work, we propose a method for accelerating model-based reinforcement learning (MBRL). We leverage state-space models (SSMs) to parallelize the sequence dimension of the world model, thereby reducing the computational complexity of MBRL. We build on the family of Dreamer-based MBRL methods Hafner et al. (2020; 2023). In our approach, we replace the recurrent state-space model (RSSM) with a modern parallelizable SSM as the dynamics model.

To evaluate our approach, we perform experiments in a drone racing environment, involving complex dynamics, for both fully and partially observable environments. We then compare our method to state-of-the-art model-free and model-based RL methods. Our method achieves a significant speedup, reducing the overall training time by up to 4 times. This benefit comes without compromising performance, as our method attains similar sample efficiency and task rewards as state-of-the-art methods.

2 Related Work

While model-free RL methods, such as proximal policy optimization (PPO), have been prevalent in robot learning Hwangbo et al. (2019); Andrychowicz et al. (2020); Alles & Aljalbout (2022); Song et al. (2023); Handa et al. (2023); Aljalbout et al. (2024a), model-based RL is gaining popularity due to its sample efficiency. The main difference between the two paradigms is that MBRL learns a model of the environment's dynamics, in addition to the policy Sutton & Barto (2018). This model, known as world model, can either be used for i) planning and control Nagabandi et al. (2018); Hansen et al. (2022), ii) sampling environment interactions for policy training Janner et al. (2019); Yu et al. (2020); Kidambi et al. (2020), or iii) obtaining first-order policy gradients through imagination Heess et al. (2015); Hafner et al. (2020; 2023). World models play a crucial role in MBRL, with recurrent neural networks (RNNs) being the most widely adopted architecture. One popular RNN-based world model architecture is the Recurrent State-Space Model (RSSM), introduced in the Dreamer framework Hafner et al. (2019). However, RSSMs struggle to scale efficiently to long sequences, as their computational complexity increases significantly with the sequence length.

Recent works have explored alternative architectures, such as Transformers and SSMs. Transformers, in particular, have been widely adopted as world model backbones in MBRL, showing advantages both for sample efficiency and computational complexity Robine et al. (2023); Zhang et al. (2023); Micheli et al. (2023); Chen et al. (2022). Similarly, in Deng et al. (2024) an S4-based world model is introduced. However, this work does not explore the usage of such world models for MBRL and focuses on mastering predictions in long-range memory tasks.

One of the earliest applications of MBRL to robotics was shown on a low-cost manipulator and using a Gaussian process-driven policy representation Deisenroth et al. (2011). Later work adopted a combination of deep dynamics models with model predictive control to scale this concept to vision-based robotics tasks Wahlström et al. (2015). Nagabandi et al. (2018) proposed using MBRL to gather expert data for training an initial policy using model-free methods. By doing so, they managed to alleviate the inferior task performance of MBRL methods. Chua et al. (2018) proposed using ensembles of dynamics model to enable uncertainty propagation during trajectory sampling. These approaches have also been extended to jointly learn a value function together with the dynamics sequence model Hansen et al. (2022; 2024). Most of these methods perform receding horizon control with a sampling scheme.

Another line of work leverages first-order policy gradients that are backpropagated through a learned latent dynamics model Heess et al. (2015); Hafner et al. (2020; 2023), which has been successfully applied to multiple robotic tasks in manipulation, locomotion, and drone flight Wu et al. (2023); Becker-Ehmck et al. (2020); Brunnbauer et al. (2022); Richard et al. (2022); Aljalbout et al. (2024b); Bi & D'Andrea (2024); Yamada et al. (2024).

3 PRELIMINARIES

State space models (SSMs) provide an efficient framework for sequence modeling tasks and have shown great success in capturing long-term temporal and spatial dependencies Gu et al. (2022); Smith et al. (2023); Patro & Agneeswaran (2024). At time t, a first-order linear system maps the input $u(t) \in \mathbb{R}^H$ to the output $y(t) \in \mathbb{R}^P$ via the following system dynamics,

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

where A, B, C, D are learnable parameters which capture the specific dependencies in the data, and $x(t)$ is the hidden state. To model sequences with a fixed time step Δ, the system is discretized as follows,

$$\begin{aligned} x_t &= \bar{A}x_{t-1} + \bar{B}u_t \\ y_t &= \bar{C}x_t + \bar{D}u_t, \end{aligned} \tag{2}$$

where $\bar{A}, \bar{B}, \bar{C}, \bar{D}$ are the discretized matrices, computed as functions of the continuous-time matrices A, B, C, D.

Given a sequence of inputs $e_{1:L}$, the parallel scan operation efficiently applies an associative binary operator, $\bullet$, to its inputs. Specifically, for a sequence of length L, the parallel scan computes:

$$[e_1, e_1 \bullet e_2, e_1 \bullet e_2 \bullet e_3, \ldots, e_1 \bullet e_2 \bullet \cdots \bullet e_L]. \tag{3}$$

Equation 3 can be computed in $\mathcal{O}(\log(L))$ using L parallel processors. S5 Smith et al. (2023) is an SSM that leverages the parallel scan to compute the sequence of hidden states $x_{1:L}$. In the context of S5, the binary operator is defined as

$$a_i \bullet a_j = (a_{j,a} \odot a_{i,a}, a_{j,a} \otimes a_{i,b} + a_{j,b}), \tag{4}$$

where $\odot$ represents matrix-matrix multiplication and $\otimes$ represents matrix-vector multiplication. The operator is applied to the initial elements e_k, which are given by the pairs of matrices

$$e_k = (e_{k,a}, e_{k,b}) := (\bar{A}, \bar{B}u_k). \tag{5}$$

Using this definition, the sequence of hidden states $x_{1:L}$ is computed iteratively as

$$\begin{aligned} e_1 &= (\bar{A}, \bar{B}u_1) = (\bar{A}, x_1) \\ e_1 \bullet e_2 &= (\bar{A}^2, \bar{A}x_1 + \bar{B}u_2) = (\bar{A}^2, x_2) \\ e_1 \bullet e_2 \bullet e_3 &= (\bar{A}^3, \bar{A}x_2 + \bar{B}u_3) = (\bar{A}^3, x_3). \end{aligned} \tag{6}$$

In general, matrix-matrix multiplications with a matrix $\bar{A}$ would incur a time complexity of $\mathcal{O}(P^3)$. However, by diagonalizing the system's matrices, the complexity is reduced to $\mathcal{O}(PL)$, where P is the size of the matrix and L is the length of the sequence. As a result, the parallel scan operation is computed in $\mathcal{O}(\log(PL))$, making it scalable to long sequences.

4 METHODOLOGY

In this work, we propose a model-based RL approach that leverages state-space world models. At each training step, we sample data from the replay buffer to train the world model as explained in section 4.1 and the actor-critic as explained in section 4.2.

4.1 State-Space World Model

The proposed world model is based on DreamerV3 Hafner et al. (2023). However, we replace the recurrent state-space model (RSSM) Hafner et al. (2019) with an SSM as the sequence model approximating the transition dynamics. Our model includes an encoder for latent variable inference, which maps the observations o_t, to posterior stochastic representations z_t. For a prediction horizon H, the sequence model predicts the deterministic representations $y_{t+1:t+H}$ given the corresponding actions $a_{t:t+H-1}$, and previous posterior $z_{t:t+H-1}$. We then infer the prior stochastic representation $\hat{z}_t$ from y_t. In addition, the model includes reward and episode continuation predictors as well as an observation decoder,

$$\begin{aligned}
&\text{Encoder:} && z_t \sim q_\phi(\cdot \mid o_t) \\
&\text{Sequence model:} && x_t, y_t = \text{SSM}(z_{t-1}, a_{t-1}, c_{t-1}, x_{t-1}) \\
&\text{Dynamics:} && \hat{z}_t \sim p_\phi(\cdot \mid y_t) \\
&\text{Decoder:} && \hat{o}_t \sim p_\phi(\cdot \mid y_t, z_t) \\
&\text{Reward prediction:} && \hat{r}_t \sim p_\phi(\cdot \mid y_t, z_t) \\
&\text{Continue prediction:} && \hat{c}_t \sim p_\phi(\cdot \mid y_t, z_t).
\end{aligned} \tag{7}$$

We fuse z_t and a_t using a multi-layer perceptron (MLP) which outputs a single vector u_t. We then feed u_t as input to the SSM Block (see Appendix F).

Unlike RSSMs, which process the sequence one step at a time, SSMs parallelize over the sequence dimension, making them particularly efficient for training on long sequences. Among the different SSM architectures, we specifically choose S5 Smith et al. (2023), which employs the parallel scan operation. One key advantage of S5 is its ability to reset the hidden state, which is not possible in other SSMs, such as S4 Gu et al. (2022). This is crucial for preventing the model from carrying over irrelevant information from previous episodes, which is particularly relevant in scenarios where sequence boundaries or discontinuities must be handled. We leverage the associative operator introduced in Lu et al. (2024), which enables efficient resetting of the hidden state. We extend the initial elements e_k from equation 5 to include the continuity predictions $c_k \in \{0, 1\}$,

$$e_k = (e_{k,a}, e_{k,b}, e_{k,c}) := (\bar{A}, \bar{B}u_k, 1 - c_k), \tag{8}$$

where $e_{k,c}$ represents whether the episode is done. We then adapt the binary operator from 4 to incorporate the continuity predictors,

$$a_i \bullet a_j = \begin{cases} (a_{j,a} \odot a_{i,a}, a_{j,a} \otimes a_{i,b} + a_{j,b}, a_{i,c}) & \text{if } a_{j,c} = 0, \\ (a_{j,a}, a_{j,b}, a_{j,c}) & \text{if } a_{j,c} = 1. \end{cases} \tag{9}$$

Using these definitions, we compute the sequence of hidden states $x_{1:L}$ as in equation 6. Since our world model uses an S5 backbone for sequence modeling, we refer to our method as S5WM.

For optimizing the world model, we use the same loss function as in Hafner et al. (2023), which is derived based on the evidence lower bound of the marginal data likelihood. The loss combines a prediction loss $\mathcal{L}_{pred}$, a dynamic loss $\mathcal{L}_{dyn}$, and a representation loss $\mathcal{L}_{rep}$,

$$\mathcal{L}(\phi) \doteq \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T} \beta_{pred}\mathcal{L}_{pred}(\phi) + \beta_{dyn}\mathcal{L}_{dyn}(\phi) + \beta_{rep}\mathcal{L}_{rep}(\phi)\right], \tag{10}$$

where β_{pred}, β_{dyn}, and β_{rep} are hyperparameters that modulate the effects of the different loss components,

$$\begin{aligned}
\mathcal{L}_{\text{pred}}(\phi) &\doteq -\ln p_\phi(o_t \mid y_t, z_t) - \ln p_\phi(r_t \mid y_t, z_t) \\
&\quad - \ln p_\phi(c_t \mid y_t, z_t) \\
\mathcal{L}_{\text{dyn}}(\phi) &\doteq \max\big(1, D_{\text{KL}}\left[\text{sg}(q_\phi(z_t \mid o_t)) \,\|\, p_\phi(z_t \mid x_t)\ \right]\big) \\
\mathcal{L}_{\text{rep}}(\phi) &\doteq \max\big(1, D_{\text{KL}}\left[q_\phi(z_t \mid o_t)) \,\|\, \text{sg}(p_\phi(z_t \mid x_t))\right]\big),
\end{aligned} \tag{11}$$

where D_{KL} refers to the Kullback-Leibler divergence, and sg to the stop gradient operation.

Each component of $\mathcal{L}(\phi)$ contributes to training a different component of the state-space world model. The representation loss ensures that the encoder learns an informative posterior stochastic latent $q_\phi(z_t \mid o_t)$. The dynamic loss helps the world model learn the transition dynamics by encouraging the prior stochastic latent $p_\phi(z_t \mid x_t)$, to align with the posterior. The prediction loss trains the decoder, as well as the reward and continue prediction heads, guiding the model to accurately predict next states.

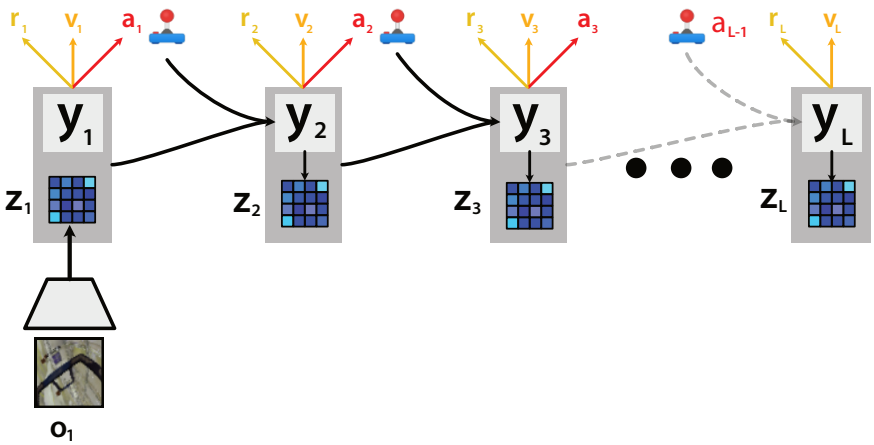

Figure 2: To train the actor and critic, we leverage imaginations in the latent state and using our state-space world model. We obtain these imaginations by encoding the initial observation and rolling out the sequence model in the latent space. Despite the possibility of training the world model in a parallel fashion, the imagination step (needed to train the actor-critic) cannot be parallelized due to the dependence on the policy to generate actions needed for rolling out the trajectories.

4.2 Actor-Critic Training

In practice, the world model serves as an approximate differentiable simulator providing first-order gradients for policy updates Xu et al. (2022). This helps reduce variance estimation, especially for systems with smooth dynamics Suh et al. (2022). Unlike model-free RL methods, which rely solely on actual environment interactions, in model-based RL, the actor-critic is trained using imagined trajectories generated by the world model. For a given horizon length H, we generate imagined trajectories by stepping through the world model, starting from the initial states $\{\hat{z}_0, x_0\}$. For $t \in \{1, H\}$,

$$\begin{aligned} x_t,\ y_t &= \text{SSM}(\hat{z}_{t-1}, a_{t-1}, \hat{c}_{t-1}, x_{t-1}) \\ \hat{z}_t &\sim\ p_\phi\left(\cdot \,|\, y_t\right) \\ \hat{r}_t &\sim\ p_\phi\left(\cdot \,|\, y_t, \hat{z}_t\right) \\ \hat{c}_t &\sim\ p_\phi\left(\cdot \,|\, y_t, \hat{z}_t\right) \\ a_t &\sim \pi_\theta\left(\cdot \,|\, y_t, \hat{z}_t\right). \end{aligned} \tag{12}$$

Unlike during world model training (see equation (7)), where we have access to the posterior stochastic latent z_t, during imagination, we rely solely on the prior stochastic latent $\hat{z}_t$. We sample the initial states $\{\hat{z}_0, x_0\}$, from the data used during world model training while ensuring sufficient context to generate meaningful predictions, and enough diversity among the initial states to learn a robust value function.

The actor π_θ is conditioned on both the deterministic representations y_t, and the prior stochastic latent $\hat{z}_t$, as shown in equation (12). The policy follows a Gaussian distribution, with actions sampled as $a_t \sim \pi_\theta\left(\cdot \,|\, y_t, \hat{z}_t\right)$. In practice, the policy is represented with an MLP outputting the mean and variance of a Gaussian distribution of actions.

The critic, similarly conditioned on y_t and $\hat{z}_t$, as well as $\hat{r}_t$ and $\hat{c}_t$, is trained to predict the bootstrapped λ-returns Sutton & Barto (2018), as proposed in Hafner et al. (2020). The critic's loss function is given by

$$\begin{aligned} \mathcal{L}(\psi) &\doteq -\textstyle\sum_{t=1}^{T} \ln p_\psi\left(V_t^\lambda \,\middle|\, x_t\right) \\ V_t^\lambda &\doteq \hat{r}_t + \gamma \hat{c}_t \Big((1-\lambda) v_t + \lambda V_{t+1}^\lambda \Big), \end{aligned} \tag{13}$$

where γ is the discount factor, $v_t \doteq \mathbb{E}[v_\psi\left(. \,|\, y_t, \hat{z}_t\right)]$ is the predicted value from the critic, and $V_T^\lambda \doteq v_T$.

The actor is trained to maximize the expected returns, which are estimated using a combination of short-horizon imagined trajectories and the value function for long-term value prediction (see Fig. 2).

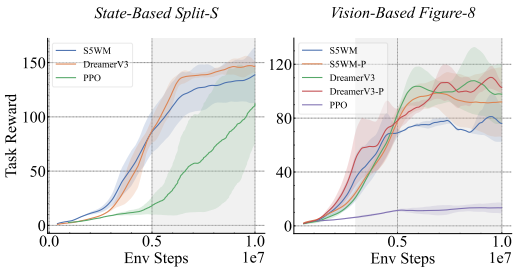

Figure 3: Task reward over the number of environment interactions for S5WM, DreamerV3, and PPO.

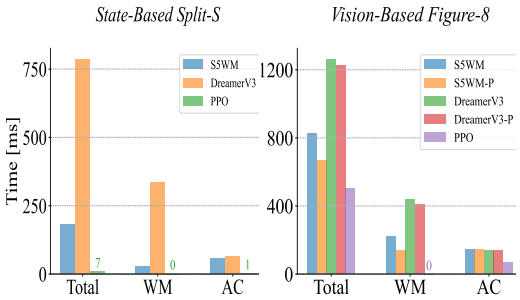

Figure 4: Each training step is divided into: i) training the world model (WM), ii) optimizing the policy (AC), and iii) collecting new data. We show the times for each stage, as well as the overall duration per step, averaged over 5×10^5 steps.

To encourage exploration, an entropy regularization term $-\eta\mathbb{H}[a_t|\hat{z}_t]$ is introduced in the actor's loss function to penalize deterministic policies and to promote more diverse actions for improved exploration,

$$\mathcal{L}(\theta) \doteq \mathbb{E}_{p_\phi, p_\theta}\Big[\sum_{t=1}^{H-1}\big(-V_t^\lambda - \eta\mathbb{H}[a_t|\hat{z}_t]\big)\Big], \tag{14}$$

Unlike DreamerV3 Hafner et al. (2023), we use the first-order gradients backpropagated through the dynamics model.

4.3 Privileged World Models

In environments with high-dimensional and partially observable inputs, such as the ones encountered in vision-based RL, learning informative latent representations is challenging. Additional supervision signals are typically required to help the model learn meaningful representations, which are essential for efficient policy optimization Lillicrap (2015); Kalashnikov et al. (2018).

When training policies in simulation, it is possible to take advantage of additional observations that are rarely available during real-world deployments, such information is typically referred to as privileged information and can be used in various ways to boost the training performance in simulation Pinto et al. (2018); Chen et al. (2020); Yamada et al. (2024). We explore the use of privileged information for training the world model. In this setup, the world model has access to privileged information during training, which is not available during real-world deployment.

In this work, we specifically explore the use of privileged state observations in a vision-based setting. During training, the image observation, o_t, is encoded into a posterior stochastic state, z_t, to be processed by the sequence model (see section 4.1). We then replace the decoder from equation (7) with a privileged decoder that reconstructs the privileged state observation, s_t, rather than the image observation,

$$\text{Privileged decoder:} \quad \hat{s}_t \sim p(\cdot|y_t, z_t), \tag{15}$$

Reconstructing the state involves predicting $s \in \mathbb{R}^{24}$, which is computationally more tractable than inferring a high dimensional image using an expensive convolutional neural network (CNN) decoder. Furthermore, learning to reconstruct state observations, rather than images, provides a stronger signal for training the encoder. This encourages the latent representation to capture information that is more relevant for policy optimization, rather than optimizing for accurate image reconstruction.

5 Experiments

5.1 Setup

Our setup is consistent across all experiments and utilizes the same quadrotor configuration. We first perform training in a simulation environment. We then validate our approach in the real

Table 1: Lap Time (LT) [s] and Success Rate (SR) [%] for S5WM (Ours), DreamerV3 Hafner et al. (2023) and PPO Schulman et al. (2017) in both simulation, as well as for the real-world experiments. For PPO, the *state-based Split-S* was not available for real-world deployment, while for *vision-based Figure-8*, the asymmetric PPO crashes after the first lap and does not learn to consistently fly through the full track.

Task	**Env**	**S5WM**		**DreamerV3**		**PPO**	
		LT	**SR**	**LT**	**SR**	**LT**	**SR**
State-based Split-S	Sim	4.82	100	4.79	100	4.76	100
	Real	4.87	100	4.92	100	-	-
Vision-based Figure-8	Sim	3.77	100	3.87	100	4.62	44
	Real	3.81	100	3.74	100	-	-

world. For the environment setup, we use a combination of the Flightmare Song et al. (2020) and Agilicious Foehn et al. (2022) software stacks. All experiments were conducted on the same hardware under uniform conditions.

5.1.1 Tasks

We investigate two tasks, both involving a quadrotor flying in a drone racing environment.

For the first task, termed *state-based Split-S*, the quadrotor flies through the Split-S track and has access to the full-state observations. This task presents highly complex dynamics, requiring the policy to push the drone's agility to the physical limits of the platform. Due to the complexity of the track, the task demands both long-term memory and high precision, as the drone must plan ahead to navigate the complex maneuvers at high speeds.

The second task, termed *vision-based Figure-8*, involves flying through the Figure-8 track and involves simpler dynamics, with the drone flying at a slower pace. However, the challenge in this task lies in learning the dynamics model from high-dimensional image observations, which can be challenging due to the partial observability of the resulting environment.

5.2 Baselines

We compare our approach against both model-based and model-free baselines. For the model-based baseline, we use DreamerV3 Hafner et al. (2023); Romero et al. (2025), while for the model-free baseline, we choose PPO Schulman et al. (2017). For the vision-based task, we introduce an additional baseline that modifies DreamerV3 to decode privileged state information instead of observation decoding. We refer to this baseline as "DreamerV3-P". We also compare against a variant of our method which omits decoding the privileged information and instead decodes raw observations. Our approach uses the same model architecture and hyperparameters as DreamerV3, with the exception of the world model configuration, where we replace the RSSM with S5WM, as introduced in section 4. We design S5WM to have a comparable number of parameters to RSSM. We tune the hyperparameters of the baselines to ensure a fair comparison. For the *state-based Split-S* task, we use standard PPO, and for the *vision-based Figure-8* task, we use PPO with the asymmetric actor-critic architecture Pinto et al. (2018), which provides privileged information to the critic, and employs the same CNN encoder architecture as used in our method and DreamerV3. We train the CNN encoder jointly with the policy. For each task, all models share the same observation, action, and reward configurations, as described in Section 5.1.

5.3 Simulation Results

We train the policies in a high-fidelity simulator on an A100 GPU. For all experiments, we simulate 50 environments in parallel and limit the number of interactions with the environment to 10^7. We use a fixed-size replay buffer containing 10^6 samples, from which we uniformly sample at each

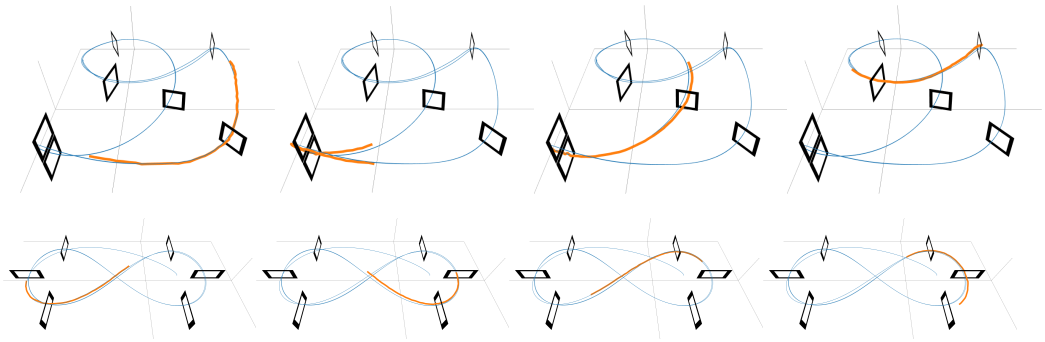

Figure 6: Imagined trajectories for S5WM on the *state-based Split-S* (top) and *vision-based Figure-8* (bottom) tasks over $H = 50$ imagination steps, initialized with a context length $C = 16$.

training step. During evaluation, we rollout the policy over 1000 steps, which corresponds to 20s of flight given a 50Hz control rate. We report the average task reward obtained per episode. We evaluate our approach based on performance, accuracy and training efficiency.

5.3.1 Performance

In Fig. 3, we evaluate the average task reward over the number of environment interactions for each task. Overall, we observe a clear advantage of MBRL approaches in terms of sample efficiency. This result corroborates previous findings from the MBRL literature Hafner et al. (2020); Janner et al. (2019); Yu et al. (2020). For the *state-based Split-S* task, we find that both S5WM and DreamerV3 achieve similar sample efficiency and converge to the maximum reward, while PPO struggles to yield competitive policies within the limited data budget.

For the *vision-based Figure-8* task, we compare both S5WM and DreamerV3 with and without the privileged world model, as well as the asymmetric PPO. We find that leveraging privileged information results in higher sample efficiency for both S5WM and DreamerV3. These benefits are more pronounced for S5WM, where the posterior stochastic latent z_t is inferred directly from the observation. In contrast, RSSMs also leverage the deterministic state y_t to infer the posterior z_t. Overall, our method performs on par with the DreamerV3 variants while being substantially faster to train as we discuss in the next section. This is evident from the final lap time and success rate achieved by our method compared to the baselines, as shown in Table 1.

For both tasks, we find that the most difficult part is completing the track after learning how to fly through the gates. Additionally, for the *vision-based Figure-8* task, the drone struggles to transition from Gate 3 to Gate 4, as it tends to overshoot and is unable to *see* the next gate.

5.3.2 Training Efficiency

We profile S5WM, along with the baselines, to assess the differences in training times. Each training step is divided into three stages: i) training the world model (observation), ii) optimizing the policy (imagination), and iii) collecting new data by interacting with the environment. Fig. 4 shows the average times for each stage, as well as the overall duration per step. We find that S5WM outperforms DreamerV3 in terms of overall training speed in both tasks, as the observations in S5WM are parallelized over the sequence length. In the *state-based Split-S* task, the observation step is up to 10 times faster, leading to an overall speedup of up to 4 times. In the *vision-based Figure-8* task, the observation step is up to 4 times faster, leading to an overall speedup of up to 2 times. These benefits are most pronounced when the main computational bottleneck lies in modeling the dynamics, such as in the *state-based Split-S* task. In tasks that require learning a latent representation from complex visual inputs, such as the *vision-based Figure-8* task, parallelizing the sequence model contributes less to the overall speedup.

5.3.3 PREDICTION ACCURACY

We further evaluate the prediction accuracy of the world model. We provide an initial context length $C = 16$ to build a history of hidden states $x_{1:C}$, followed by imagination over the horizon H. During the context period, the world model has access to the true observations from the environment, while during the imagination period, the world model predicts the trajectory without access to the observations. Fig. 10 shows the trajectories imagined by S5WM for $H = 50$ for different parts of the track, along with the ground truth, which is obtained by interacting with the environment.

5.4 REAL WORLD DEPLOYMENT

We test our approach in the real world using a high-performance racing drone Foehn et al. (2022). We select the best policy obtained during training and deploy it in the real world. A video of the real-world deployment of both tasks can be found in the supplementary material. In addition, Fig. 8 shows the trajectories from the real-world experiments. For the *state-based Split-S* task, the Split-S maneuver at $x = -4.3\mathrm{m}$ and $y = -5.1\mathrm{m}$, stands out as a critical test of each approach's characteristics. This complex maneuver requires the drone to fly through a higher gate and then immediately descend through a second gate located directly below the first one, with both gates sharing the same x, y coordinates. This is the most challenging maneuver of the Split-S track, significantly influencing the overall lap time.

For the *vision-based Figure-8* task, we use a hardware-in-the-loop (HIL) setup, where images are rendered in real-time from the simulator based on state estimation from a motion-caption system estimating the position of the real robot. For both tasks, our method enables high-speed flight on very smooth trajectories.

6 CONCLUSION

Despite their notable sample efficiency, model-based reinforcement learning methods can be extremely slow to train, which limits their usability in robotic systems. This paper introduces S5WM, a state-space world model designed to accelerate training in model-based reinforcement learning. We adapt the S5 SSM architecture to handle resetability at the episode boundaries, and use it as a sequence model of the dynamics in our world model. Additionally, we introduce asymmetry in the world model by providing privileged state information during training in simulation.

We compare our approach against model-based and model-free baselines, focusing on training time, performance, and sample efficiency. Our method achieves faster training times without compromising performance and sample efficiency. Additionally, we test our approach on a drone racing environment and conduct real-world experiments for two distinct tasks involving both state and image observations. Our results demonstrate that S5WM performs well in challenging real-world scenarios. Furthermore, we conduct a series of ablations to highlight the key components of our approach.

While the presented approach shows promising results, we identify several directions for future work. For instance, one valuable area of research would be to speed up the actor-critic training in MBRL. Additionally, our proposed approach is not as robust to hyperparameter changes as more mature MBRL methods such as Dreamer. Future work could investigate modifications to our method to improve its robustness to hyperparameter choice. Furthermore, it would be interesting to apply and evaluate our method on different robotic domains such as locomotion and manipulation.

In summary, our method offers a promising framework for accelerating model-based reinforcement learning, combining high sample efficiency with faster training times, making it more feasible for real-world robotic deployment.

REFERENCES

Elie Aljalbout, Ji Chen, Konstantin Ritt, Maximilian Ulmer, and Sami Haddadin. Learning vision-based reactive policies for obstacle avoidance. In *Conference on Robot Learning*, pp. 2040–2054. PMLR, 2021.

Elie Aljalbout, Felix Frank, Maximilian Karl, and Patrick van der Smagt. On the role of the action space in robot manipulation learning and sim-to-real transfer. *IEEE Robotics and Automation Letters*, 2024a.

Elie Aljalbout, Nikolaos Sotirakis, Patrick van der Smagt, Maximilian Karl, and Nutan Chen. Limt: Language-informed multi-task visual world models. *arXiv preprint arXiv:2407.13466*, 2024b.

Marvin Alles and Elie Aljalbout. Learning to centralize dual-arm assembly. *Frontiers in Robotics and AI*, 9:830007, 2022.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Philip Becker-Ehmck, Maximilian Karl, Jan Peters, and Patrick van der Smagt. Learning to fly via deep model-based reinforcement learning. *arXiv preprint arXiv:2003.08876*, 2020.

Thomas Bi and Raffaello D'Andrea. Sample-efficient learning to solve a real-world labyrinth game using data-augmented model-based reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7455–7460. IEEE, 2024.

Axel Brunnbauer, Luigi Berducci, Andreas Brandstátter, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. Latent imagination facilitates zero-shot transfer in autonomous racing. In *2022 international conference on robotics and automation (ICRA)*, pp. 7513–7520. IEEE, 2022.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pp. 66–75. PMLR, 2020.

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

Marc Deisenroth, Carl Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, 2011.

Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: Rnns, transformers, and s4. *Advances in Neural Information Processing Systems*, 36, 2024.

Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67), 2022. doi: 10.1126/scirobotics.abl6259.

Niklas Funk, Georgia Chalvatzaki, Boris Belousov, and Jan Peters. Learn2assemble with structured representations and search for robotic architectural construction. In *Conference on Robot Learning*, pp. 1401–1411. PMLR, 2022.

Ismail Geles, Leonard Bauersfeld, Angel Romero, Jiaxu Xing, and Davide Scaramuzza. Demonstrating agile flight from pixels without state estimation. *arXiv preprint arXiv:2406.12505*, 2024.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=uYLFoz1vlAC`.

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

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation*, pp. 5977–5984. IEEE, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

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

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

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

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pp. 651–673. PMLR, 2018.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, Aug 2023. ISSN 1476-4687.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020a.

Michelle A Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *IEEE Transactions on Robotics*, 36(3):582–596, 2020b.

TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4163–4170. IEEE, 2021.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

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

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Badri Narayana Patro and Vijay Srinivas Agneeswaran. Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges. *arXiv preprint arXiv:2404.16112*, 2024.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *Robotics: Science and Systems XIV*, 2018.

Antoine Richard, Stéphanie Aravecchia, Matthieu Geist, and Cédric Pradalier. Learning behaviors through physics-driven latent imagination. In *Conference on Robot Learning*, pp. 1190–1199. PMLR, 2022.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

Angel Romero, Ashwin Shenai, Ismail Geles, Elie Aljalbout, and Davide Scaramuzza. Dream to fly: Model-based reinforcement learning for vision-based drone flight. *arXiv preprint arXiv:2501.14377*, 2025. URL https://arxiv.org/abs/2501.14377.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. 2019. URL https://arxiv.org/abs/1904.01201.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017. URL https://arxiv.org/abs/1707.06347.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Conference on Robot Learning*, 2020.

Yunlong Song, Angel Romero, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82):eadg1462, 2023.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pp. 20668–20696. PMLR, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *Robotics: Science and Systems XIV*, 2018.

Niklas Wahlström, Thomas B. Schön, and Marc P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. In *Deep Learning Workshop at ICML 2015*, 2015. URL https://arxiv.org/abs/1502.02251.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pp. 2226–2240. PMLR, 2023.

Jiaxu Xing, Angel Romero, Leonard Bauersfeld, and Davide Scaramuzza. Bootstrapping reinforcement learning with imitation for vision-based agile flight. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=bt0PX0e4rE.

Jie Xu, Miles Macklin, Viktor Makoviychuk, Yashraj Narang, Animesh Garg, Fabio Ramos, and Wojciech Matusik. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZSKRQMvttc.

Jun Yamada, Marc Rigter, Jack Collins, and Ingmar Posner. Twist: Teacher-student world model distillation for efficient sim-to-real transfer. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9190–9196. IEEE, 2024.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pp. 10674–10681, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

A HYPERPARAMETERS

The RSSM is configured as in DreamerV3 Hafner et al. (2023), whereas the parameters related to the world model have been adjusted for the S5WM.

	S5WM	RSSM
General		
Steps	10^7	10^7
Optimizer	Adam	Adam
Activation	SiLU	SiLU
MLP units	512	512
MLP layers	2	2
Batch Size	32	16
Batch Length	128	128
Dataset Size	10^6	10^6
Num. Environments	50	50
Discount Factor γ	0.997	0.997
Dynamic Factor β_{dyn}	0.5	0.5
Representation Factor β_{rep}	0.1	0.1
World Model		
Learning Rate	5 x 10^{-4}	1 x 10^{-4}
Deterministic State y	512	512
Stochastic State z	32x32	32x32
Imagination Horizon H	10	10
Imagination Gradient	Dynamics	Dynamics
Context Length	32	32
State Space Model		
Hidden State x	256	-
Activation	Half GLU	-
Num. Layers	1	-
Degree	1	-
Learning Rate	5 x 10^{-4}	-
Actor		
Activation	Tanh	Tanh
Distribution	Gaussian	Gaussian
Entropy Coefficient η	2 x 10^{-3}	2 x 10^{-3}
Learning Rate	3 x 10^{-5}	3 x 10^{-5}
Critic		
Activation	Tanh	Tanh
Distribution	symlog_disc	symlog_disc
Learning Rate	3 x 10^{-5}	3 x 10^{-5}

Table 2: Hyperparameters for the *state-based Split-S* task.

	S5WM (x10^6)	RSSM (x10^6)
Encoder	0.28	0.28
Dynamics	3.55	3.94
Reward	1.44	1.44
Continuity	1.05	1.05
Decoder	1.06	1.06
Actor	1.05	1.05
Critic	1.18	1.18
Total	8.62	8.94

Table 3: Number of parameters by network component for the *state-based Split-S* task.

B Imagination Prediction

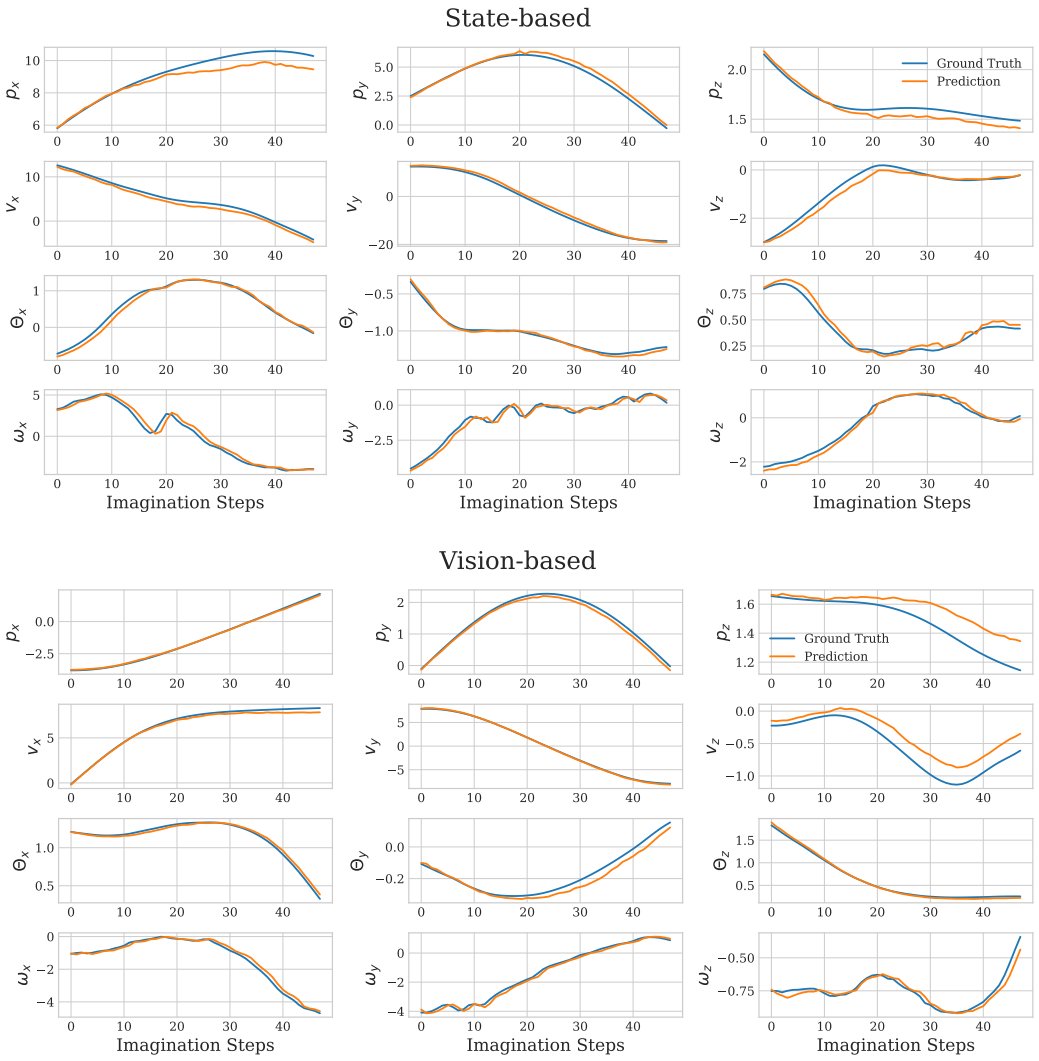

Figure 7: Imagined state for S5WM on *state-based Split-S* and *vision-based Figure-8* tasks over $H = 50$ steps.

	S5WM	RSSM
General		
Steps	10^7	10^7
Optimizer	Adam	Adam
Activation	SiLU	SiLU
MLP units	768	768
MLP layers	4	4
Batch Size	32	16
Batch Length	128	128
Dataset Size	10^6	10^6
Num. Environments	50	50
Discount Factor γ	0.997	0.997
Dynamic Factor β_{dyn}	0.5	0.5
Representation Factor β_{rep}	0.1	0.1
World Model		
Learning Rate	10^{-3}	10^{-4}
Deterministic State y	1024	1024
Stochastic State z	32x32	32x32
Imagination Horizon H	10	10
Imagination Gradient	Dynamics	Dynamics
Context Length	32	32
State Space Model		
Hidden State x	512	-
Activation	Half GLU	-
Num. Layers	2	-
Degree	1	-
Learning Rate	5 x 10^{-4}	-
Actor		
Activation	Tanh	Tanh
Distribution	Gaussian	Gaussian
Entropy Coefficient η	2 x 10^{-3}	2 x 10^{-3}
Learning Rate	3 x 10^{-5}	3 x 10^{-5}
Critic		
Activation	Tanh	Tanh
Distribution	symlog_disc	symlog_disc
Learning Rate	3 x 10^{-5}	3 x 10^{-5}

Table 4: Hyperparameters for the *vision-based Figure-8* task.

C Observation Space

For the *state-based Split-S* task, we define the observation $s = [p, \tilde{R}, v, \omega, i, d, a_{prev}] \in \mathbb{R}^{24}$, where $p \in \mathbb{R}^3$ is the position of the drone, $\tilde{R} \in \mathbb{R}^6$ contains the first two columns of the rotation matrix, $v \in \mathbb{R}^3$ is the linear velocity, and $\omega \in \mathbb{R}^3$ is the angular velocity. The vector $i \in \mathbb{R}^2$ encodes the gate index using sine-cosine encoding to address the periodicity of the track. The continuous gate index i_c is defined as

$$i_c = i + \frac{2}{1 + \exp(k \cdot d)}, \tag{16}$$

where $d \in \mathbb{R}$ is the distance to the next gate, and $a_{prev} \in \mathbb{R}^4$ represents the previous action.

For the *vision-based Figure-8* task, the observation space is given by RGB images of size $64 \times 64 \times 3$, rendered from the simulator Savva et al. (2019). The raw pixel-level input is then processed through a CNN encoder to extract relevant features. Similarly to Geles et al. (2024), we include the previous

	S5WM (x10^6)	RSSM (x10^6)
Encoder	0.69	0.69
Dynamics	13.05	12.60
Reward	3.94	3.94
Continuity	3.35	3.35
Decoder	3.37	3.37
Actor	3.35	3.35
Critic	3.54	3.54
Total	31.29	30.84

Table 5: Number of parameters by network component for the *vision-based Figure-8* task.

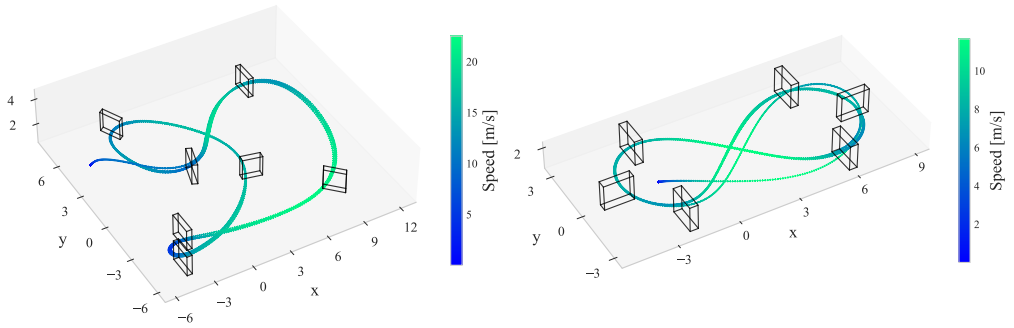

Figure 8: Real world flight trajectories with S5WM: *State-based Split-S* task (left) and *Vision-based Figure-8* task (right).

3 actions in the observation to provide historical context.

D Action Space

For both tasks, we define $a = [c, \omega_{des}] \in \mathbb{R}^4$, where c is the mass-normalized collective thrust and $\omega_{des} \in \mathbb{R}^3$ represents the desired body rates. These commands are then processed by a low-level controller, which outputs the desired motor speeds.

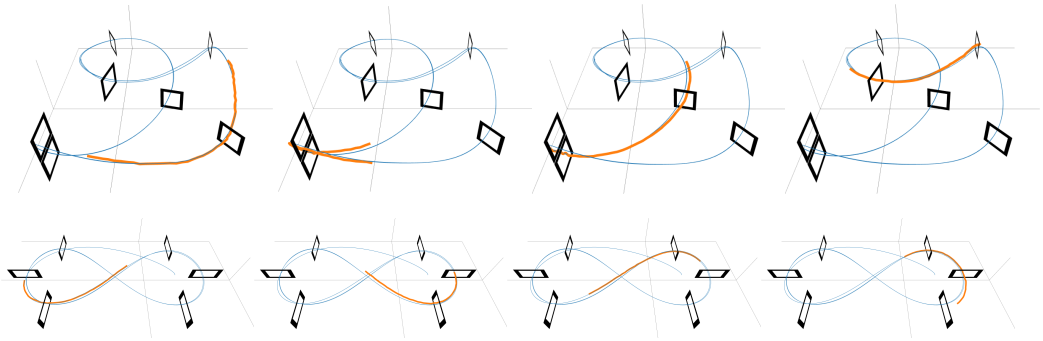

Figure 10: Imagined trajectories for S5WM on the *state-based Split-S* (top) and *vision-based Figure-8* (bottom) tasks over $H = 50$ imagination steps, initialized with a context length $C = 16$.

E REWARD

Similar to prior works on drone racing Kaufmann et al. (2023); Song et al. (2023), we encode the task using a dense reward function r_t,

$$r_t = \begin{cases} r_{\text{crash}} & \text{crashed} \\ r_{\text{pass}} & \text{gate passed} \\ r_{\text{prog}} + r_{\text{omega}} + r_{\text{cmd}} + r_{\Delta\text{cmd}} & \text{otherwise.} \end{cases} \tag{17}$$

where r_{crash} is a terminal penalty for collisions, r_{pass} encourages passing a gate, r_{prog} encourages progress along the track, r_{omega} penalizes excessive body rates, r_{cmd} penalizes aggressive actions, and $r_{\Delta\text{cmd}}$ encourages smooth actions.

E.1 REWARD COMPONENTS

i. *Progress Reward*: Encourages the drone to progress along the track as fast as possible,

$$r_{\text{prog}} = \lambda_1(d_{t-1} - d_t), \tag{18}$$

where d_t is the distance from the position at time t to the next gate, and $\lambda_1 = 1$.

ii. *Omega Penalty*: Penalizes large angular velocities,

$$r_{\text{omega}} = \lambda_2 \left(\frac{\|\omega\|}{\|\omega_{\text{max}}\|} \right)^2, \tag{19}$$

where ω is the angular velocity, ω_{max} is the maximum angular velocity, and $\lambda_2 = -0.05$.

iii. *Command Omega Penalty*: Penalizes large desired angular velocity values (actions),

$$r_{\text{cmd}} = \lambda_3 \left(\frac{\|\omega_{\text{des}}\|}{\|\omega_{\text{max}}\|} \right)^2, \tag{20}$$

where ω_{des} is the desired angular velocity command, ω_{max} is the maximum angular velocity, and $\lambda_3 = -0.025$.

iv. *Command Difference Penalty*: Penalizes abrupt changes between consecutive actions to encourage smooth signals,

$$r_{\Delta\text{cmd}} = \lambda_4 \left(\frac{\|\Delta a_t\|}{\|a_{\text{max}}\|} \right)^2, \tag{21}$$

where $\Delta a_t = a_t - a_{t-1}$, $a_t \in \mathbb{R}^4$ is the control action at time t, a_{max} is the maximum control action, and $\lambda_4 = -0.05$.

v. *Crash Penalty*: Terminal reward that penalizes collisions with the gates,

$$r_{\text{crash}} = -4. \tag{22}$$

vi. *Passing Reward*: Encourages the drone to pass through the gate successfully,

$$r_{\text{pass}} = 10. \tag{23}$$

F S5 SEQUENCE LAYER

For the Gated Linear Unit (GLU), we use a specific gating function, defined as *half GLU*, which maps its input u to the output y as follows,

$$x_1 = GELU(u) \tag{24}$$

$$y = u \cdot \sigma(W^T x_1) \tag{25}$$

where W represents the weights of a linear layer, $GELU$ is a standard Gaussian Error Linear Unit, and σ is the sigmoid function.

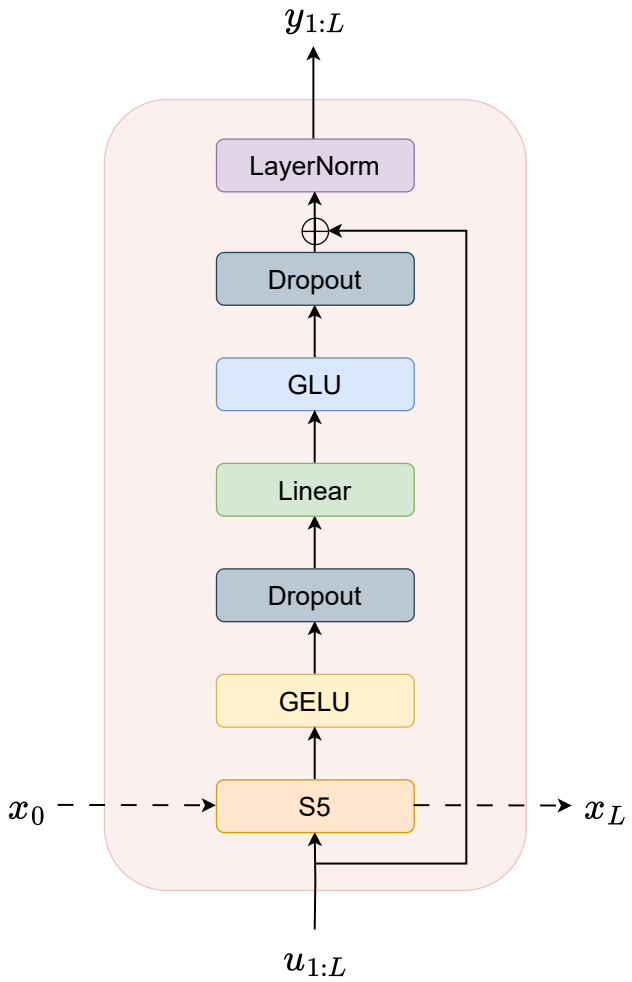

Figure 11: Schematic representation of the S5 sequence layer, composed of the resettable S5 SSM described in 4.1, nonlinearities, and gating operations.

G TRAJECTORIES

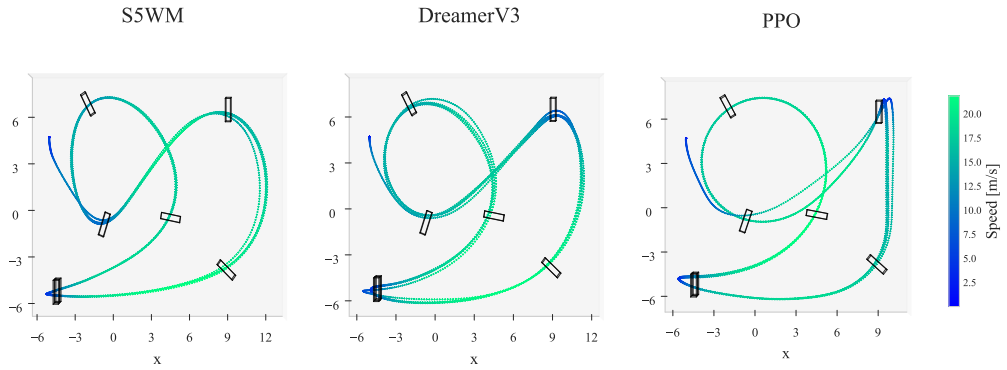

Figure 12: *State-based Split-S task:* Comparison for S5WM, with baselines: DreamerV3 and PPO.

H Ablations

We discuss several key components that we find to be essential to the success of our approach.

H.1 Horizon Length

In model-based RL, the actor-critic uses the world model to plan over a prediction horizon, which can substantially improve sample efficiency compared to model-free RL. While a longer horizon is typically preferred, prediction errors tend to compound over time and therefore limit the effectiveness of imagination. Additionally, imagination steps are computationally expensive and cannot be parallelized, making longer horizons computationally more expensive. To investigate the influence of the horizon length on our method, we train the *state-based Split-S* task using different horizon lengths. Fig. 14 shows the task reward over the number of environment interactions for $H \in \{5, 10, 15\}$, as well as the time it takes till convergence. We find that $H = 10$ offers a good balance between sample efficiency and computational complexity, and we therefore use it for all of our experiments.

H.2 Reward Smoothing

Reward smoothing is essential for enhancing training. In model-free RL, it is common to adapt the reward function during training to increase the task difficulty. While this helps the agent adapt to progressively harder settings, we find that directly altering the reward function in a model-based RL setting can be counterproductive, because the world model needs to *unlearn* the old reward signal and relearn the new one.

To address this issue, we propose a smoothing strategy that decouples the total reward into two complementary components,

$$r = r_{\text{nom}} + r_{\text{aug}}, \tag{26}$$

where r_{nom} (the nominal reward) is active from the start of training, and r_{aug} (the smoothing reward) is learned as well from the start but only applied once the agent becomes sufficiently proficient. Initially, the value function only receives r_{nom}, and later r_{aug} is also added. We use this strategy to encourage smoother actions, which is particularly important for real-world deployment. We define

$$\begin{aligned} r_{\text{nom}} &= r_{\text{prog}} + r_{\text{omega}} + r_{\text{cmd}} \\ r_{\text{aug}} &= r_{\Delta\text{cmd}}, \end{aligned} \tag{27}$$

where we follow the definitions from section E.1. We apply r_{aug} halfway through training, after 5×10^6 steps. If r_{aug} is instead applied from the beginning, the agent struggles to learn the task. By delaying the smoothing reward, the agent first explores to discover a feasible solution, and later refines it to produce smoother actions. Fig. 15 shows the actions learned with and without reward smoothing. By introducing this smoothing curriculum, we significantly improve the smoothness of

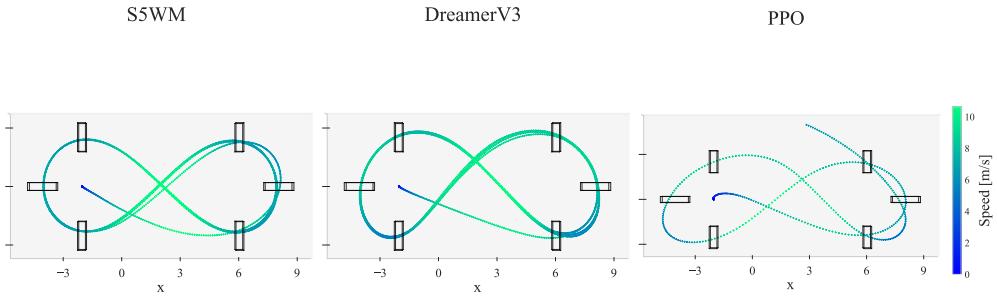

Figure 13: *Vision-based Figure-8 task:* Comparison for S5WM (privileged), with baselines: DreamerV3 (privileged) and PPO (asymmetric). The asymmetric PPO crashes after the first lap and does not learn to consistently fly through the track.

the control commands, preventing motor damage and reducing the sim-to-real gap by ensuring the policy does not learn infeasible command sequences in simulation.

H.3 Recurrent vs Parallel

One main difference between our S5WM architecture (see section 4.1) and RSSMs lies in how the posterior stochastic latent z_t is computed. In RSSMs, each step computes a prior $\hat{z}_t \sim p_\phi(\cdot \mid y_t)$ and a posterior $z_t \sim q_\phi(\cdot \mid y_t, o_t)$), both of which include the deterministic representation y_t. Therefore, the posterior receives the same information as the prior plus access to the latest observation o_t. By contrast, since S5WM processes the entire sequence in parallel, the posterior is determined only from the observation, $z_t \sim q_\phi(\cdot \mid o_t)$, and does not have access to historical context encoded in y_t. To compensate for this, we append past actions to o_t, as explained in section D, providing S5WM with additional temporal context.

We investigate this difference by implementing a *recurrent S5WM* variant where the posterior also depends on y_t, i.e., $z_t \sim q_\phi(\cdot \mid y_t, o_t)$. We compare the *recurrent S5WM* to the standard parallel S5WM on the *vision-based Figure-8* task. Note that the *recurrent S5WM* reintroduces the sequential dependencies of RSSMs, and therefore offers no speed-up in training. This comparison is included mainly to highlight the importance of historical context in partially observable environments.

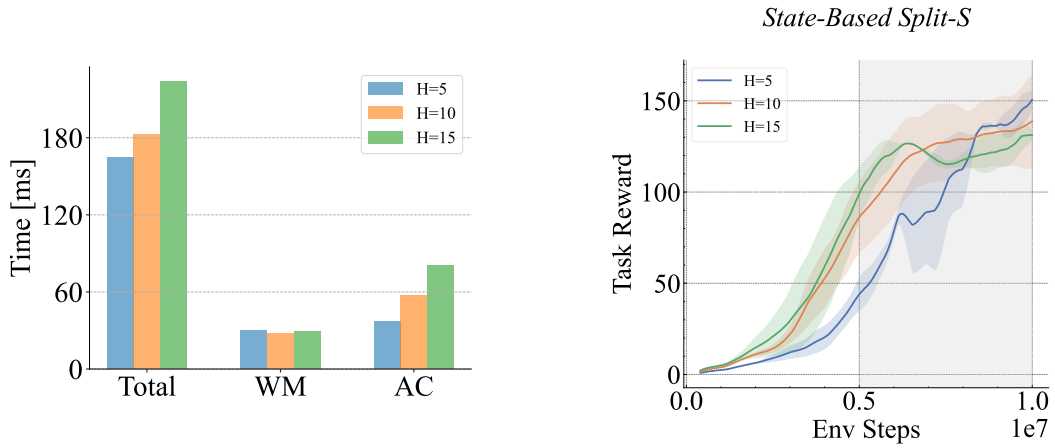

Figure 14: Task reward over the number of environment interactions on S5WM for different imagination horizon lengths $H \in \{5, 10, 15\}$.

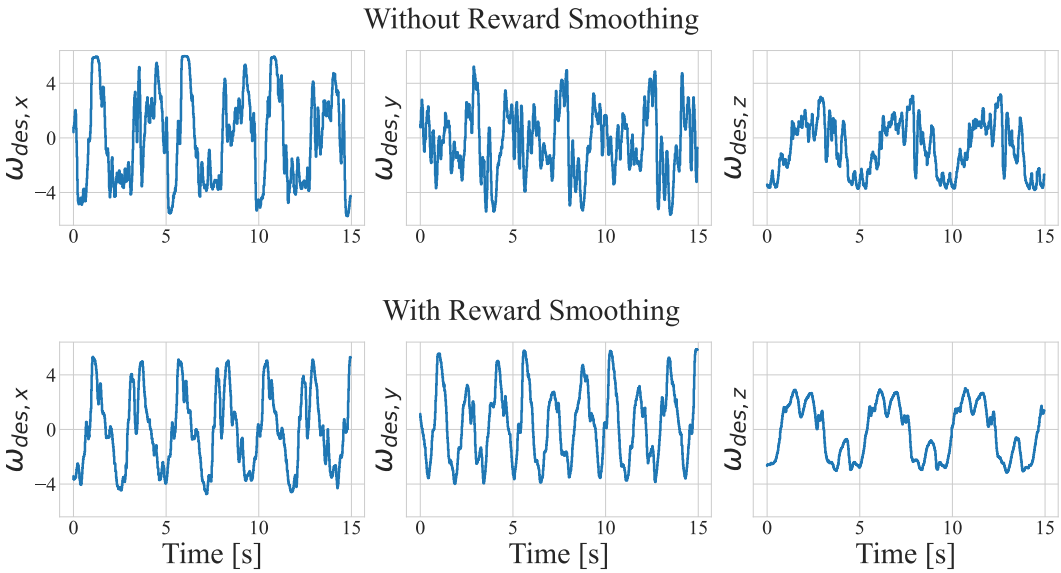

Figure 15: Desired body rates (actions) learned with and without reward smoothing on the *state-based Split-S* task.

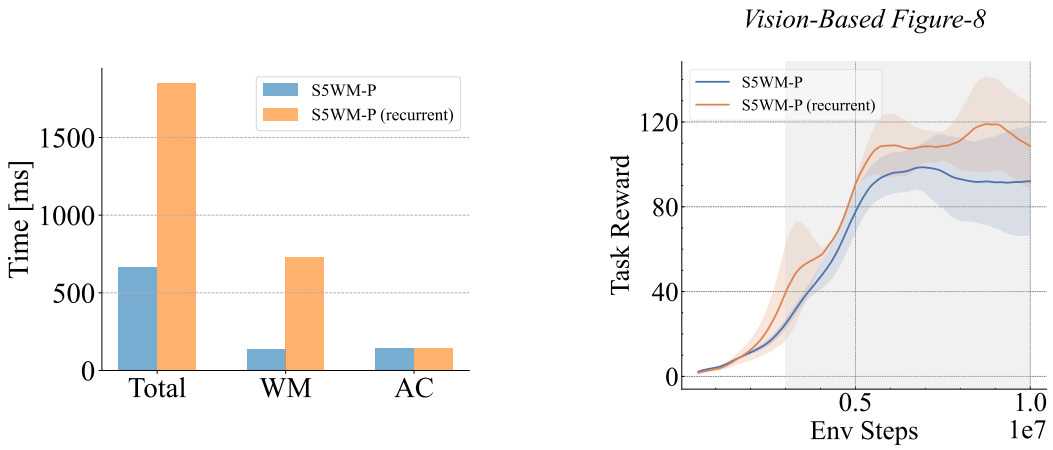

Figure 16: Task reward over the number of environment interactions for parallel and recurrent variants of S5WM.