State–Readout Decoupling for Latent World Models

Xianxin Lai,[1] **Ziyi Ding,**[2] **Weiyu Chen,**[1]
Xiao-Ping Zhang,[2] **Jiayu Chen**[1,3†]
[1]The University of Hong Kong
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University
[3]INFIFORCE Intelligent Technology
[†]Corresponding author.
xianxin.lai@connect.hku.hk, dingzy25@mails.tsinghua.edu.cn, weiyuc@connect.hku.hk,
xpzhang@ieee.org, jiayuc@hku.hk

Abstract

Reconstruction-free latent world models commonly construct multi-step trajectories by recursively feeding each predicted latent into a one-step transition model. Because the same latent serves as both a predictor readout and the input to the next transition, prediction errors are directly returned to later dynamics. We propose State–Readout Decoupling (SRD), which reformulates rollout around a planning-horizon sequence of recurrent hidden states and reads out future latents from this sequence without reusing them as inputs to subsequent transitions. SRD therefore changes the native prediction object from a recursively constructed chain of latent predictions to a horizon-level hidden state trajectory. We realize this formulation with a matched hidden state rollout predictor: a GRU advances the hidden state sequence over the full action horizon, while a shared readout maps every hidden state to a planner-facing latent prediction under horizon-aligned supervision. Across LeWM and PLDM on four visual goal-reaching tasks, SRD achieves higher mean success in seven of eight backbone–task settings while using 53–58% fewer predictor parameters and 44% less evaluation time on average. In the long-horizon stress test, SRD's success advantage over autoregressive rollout widens from 8 to 32 percentage points; at the longest horizon, it reduces open-loop error by 17.2% and evaluation time by 86.8%.

Introduction

Reconstruction-free latent world models commonly construct a multi-step rollout by recursively feeding each predicted latent into the next transition. The same latent therefore serves two distinct roles: it is exposed to the planner for supervision and goal comparison, and it is reused as the state from which the following prediction is generated. An inaccurate intermediate readout thus changes the input to every subsequent transition. We refer to this shared interface as *state–readout coupling*.

This rollout pattern appears in latent world models that support planning by predicting how compact visual representations evolve under candidate actions. Such models have enabled recurrent simulation, imagined control, search, and model-predictive control (Ha and Schmidhuber 2018; Hafner et al. 2019, 2020; Schrittwieser et al. 2020; Hafner et al. 2025; Hansen, Su, and Wang 2024; Finn and Levine 2017). Recent reconstruction-free models, including PLDM and LeWM, further support offline, reward-free, goal-conditioned planning (Sobal et al. 2025; Maes et al. 2026).

Existing approaches mitigate multi-step degradation through long-horizon latent supervision and auxiliary future-prediction objectives (Hafner et al. 2019; Schwarzer et al. 2021; Li et al. 2026). Figure 1(a) shows that extending supervision can leave the autoregressive rollout interface unchanged: each planner-facing latent prediction is still reused as the input to the next transition. The rollout therefore remains a recursively constructed chain of predicted latents with a direct latent readout feedback path.

This observation motivates a different question: *what should be the native rollout object of a latent world model over a planning horizon?* We propose **State–Readout Decoupling (SRD)**, which organizes rollout around a planning-horizon trajectory of recurrent hidden states. Figure 1(b) shows the corresponding SRD rollout. Starting from the current latent representation z_t, SRD initializes a recurrent hidden state and advances it through the complete candidate action sequence. A shared readout maps each post-action hidden state to a planner-facing latent prediction. These predictions remain available for supervision and goal-conditioned scoring but are never fed back into later transitions. The hidden state therefore carries the rollout dynamics, whereas the latent predictions serve only as planner-facing readouts.

SRD changes the native prediction object from a one-step latent transition recursively composed at evaluation to a horizon-level hidden state trajectory. We realize this formulation with a matched hidden state rollout predictor: a GRU (Cho et al. 2014) advances the hidden state sequence over the full action horizon, while a shared readout maps each post-action state to the latent space of the original backbone. Because this predictor natively produces one hidden state and one latent readout for every action prefix, we supervise the complete readout trajectory with a horizon-aligned objective. The hidden state trajectory is therefore the common object advanced by the recurrent predictor, mapped to planner-facing predictions, and trained across the complete horizon. The encoder architecture, target latent construction, goal representation, and downstream planner remain unchanged.

We evaluate SRD with LeWM and PLDM on four visual goal-reaching tasks. SRD improves success in seven of eight backbone–task settings while reducing predictor parameters

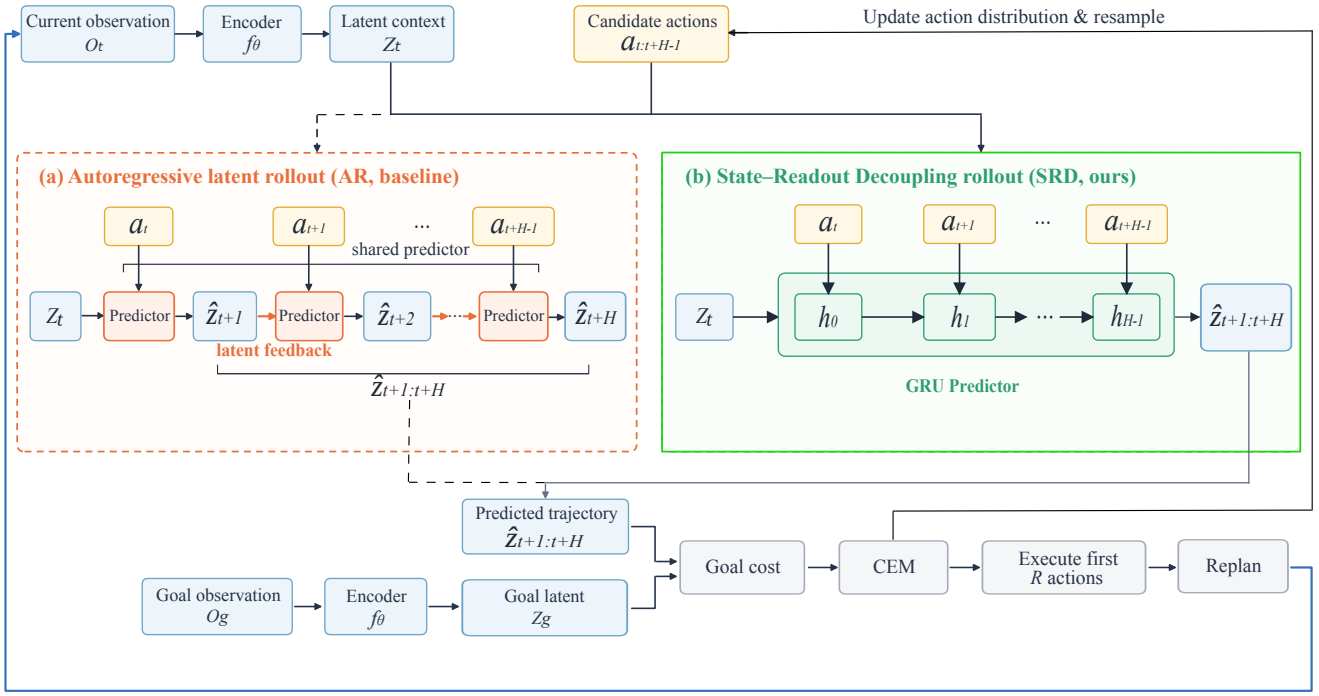

Figure 1: Comparison of the original autoregressive rollout and our SRD rollout. SRD replaces AR as the sole rollout module, while all other components of the goal-conditioned planning pipeline remain unchanged.

by 53–58% and evaluation time by 44% on average. Under jointly increasing planning horizon and goal distance, its success advantage over autoregressive rollout grows from 8 percentage points at $H = 5$ to 32 points at $H = 20$. Ablations further separate the effects of multi-step supervision, recurrent architecture, and the representation used as the rollout carrier. Our contributions are:

1. We identify **state–readout coupling** as an explicit source of error feedback in autoregressive rollout for reconstruction-free latent world models.
2. We propose **SRD**, which separates the recurrent hidden state from planner-facing latent readouts and realizes this interface through an efficient horizon-level state rollout.
3. We characterize the resulting removal of latent readout feedback and demonstrate improved long-horizon control, open-loop prediction, and rollout efficiency across two latent world model backbones.

Related Work

Latent world models for control. World models support planning and behavior learning by predicting future task-relevant states under candidate actions (Silver et al. 2017; Oh, Singh, and Lee 2017; Gelada et al. 2019; Hansen, Su, and Wang 2022). Early recurrent world models maintain hidden states for simulating future experience (Ha and Schmidhuber 2018). PlaNet and Dreamer combine recurrent latent dynamics with planning or imagined behavior learning (Hafner et al. 2019, 2020, 2021, 2025; Hafner, Yan, and Lillicrap 2025), while MuZero and TD-MPC2 learn latent transition models for search and model-predictive control (Schrittwieser et al. 2020; Hansen, Su, and Wang 2024). Recent work has also studied policy-driven model adaptation for robust offline model-based reinforcement learning (Chen et al. 2025) and causal latent structures for intervention-aware counterfactual dynamics (Ding et al. 2026). Reconstruction-free models such as DINO-WM, PLDM, and LeWM further use predicted visual representations directly for goal-conditioned planning (Zhou et al. 2025; Sobal et al. 2025; Maes et al. 2026). These approaches vary in their representation learning objectives, dynamics structures, and uses of predicted states. Our focus is more specific: the representation recursively propagated when a one-step latent transition is composed over a planning horizon.

Predictive representation learning. Joint-Embedding Predictive Architectures learn representations by predicting target embeddings instead of reconstructing observations. I-JEPA and V-JEPA establish this approach for images and videos (Assran et al. 2023; Bardes et al. 2024), while Self-Predictive Representations learn action-conditioned transitions toward future latent targets (Schwarzer et al. 2021). DINO-WM predicts features from a frozen visual encoder, whereas PLDM and LeWM jointly learn the visual representation and latent dynamics (Zhou et al. 2025; Sobal et al. 2025; Maes et al. 2026). This line of work primarily determines the latent space and the targets against which future representations are trained. SRD retains these backbone-specific choices, including the encoder architecture, target

construction, goal representation, and planner-side scoring procedure.

Multi-step prediction and rollout objectives. Autoregressive sequence models can suffer from a mismatch between training on observed inputs and inference on their own predictions (Bengio et al. 2015). In learned dynamics, composing imperfect one-step transitions can propagate local errors, motivating self-correcting, short, or uncertainty-aware rollouts (Talvitie 2017; Janner et al. 2019). One-step prediction accuracy may also be misaligned with downstream control performance (Lambert et al. 2020). Multi-step objectives such as latent overshooting and self-predictive learning therefore supervise representations farther into the future (Hafner et al. 2019; Schwarzer et al. 2021). More recent reconstruction-free methods attach multi-horizon supervision to an autoregressive predictor (Li et al. 2026; Rao et al. 2026). These methods primarily extend the supervision horizon while retaining planner-facing latent predictions as the recursively propagated inputs of the rollout. SRD changes this rollout carrier: it advances an hidden state through the action sequence and obtains planner-facing latent predictions through a separate readout. The multi-step objective then supervises the readout associated with every state in the resulting hidden state trajectory.

Preliminaries

Goal-Conditioned Latent Planning

At time t, the planner receives a current observation o_t, a goal observation o_g, and a candidate action sequence of planning horizon H,

$$a_{t:t+H-1} = (a_t, \dots, a_{t+H-1}), \tag{1}$$

where a_{t+k} denotes the action applied at rollout step k.

The visual encoder f_θ maps the current and goal observations to latent representations,

$$z_t = f_\theta(o_t), \qquad z_g = f_\theta(o_g). \tag{2}$$

We use z_t to denote the latent representation available to the rollout predictor at time t.

For the future observations associated with the candidate action sequence, the original backbone constructs a target latent trajectory

$$z^{\mathrm{tgt}}_{t+1:t+H} = \left(z^{\mathrm{tgt}}_{t+1}, \dots, z^{\mathrm{tgt}}_{t+H}\right), \tag{3}$$

using either stop-gradient representations or a target encoder. Given z_t and $a_{t:t+H-1}$, the rollout model predicts

$$\hat{z}_{t+1:t+H} = \left(\hat{z}_{t+1}, \dots, \hat{z}_{t+H}\right). \tag{4}$$

The unchanged downstream planner scores the predicted latent trajectory with respect to the goal representation z_g and uses the resulting costs to select actions.

Autoregressive Latent Rollout

A standard autoregressive (AR) latent world model learns an action-conditioned one-step predictor P_ϕ,

$$\hat{z}_{t+1} = P_\phi(z_t, a_t), \tag{5}$$

and constructs a planning trajectory by recursively applying the same predictor:

$$\hat{z}_{t+k+1} = P_\phi(\hat{z}_{t+k}, a_{t+k}), \qquad k = 1, \dots, H-1. \tag{6}$$

The resulting rollout is therefore a chain of predicted latent representations. Each $\hat{z}_{t+k}$ serves two interfaces: it is a planner-facing readout used for supervision and goal-conditioned scoring, and it is also the input to the subsequent transition. We refer to this shared interface as *state–readout coupling*.

Error Feedback under State–Readout Coupling

To characterize the resulting feedback path, let

$$e_k = \hat{z}_{t+k} - z^{\mathrm{tgt}}_{t+k} \tag{7}$$

denote the latent readout error at rollout step k. Since

$$\hat{z}_{t+k} = z^{\mathrm{tgt}}_{t+k} + e_k, \tag{8}$$

the next prediction can be expressed as

$$\hat{z}_{t+k+1} = P_\phi\left(z^{\mathrm{tgt}}_{t+k} + e_k, a_{t+k}\right). \tag{9}$$

A first-order expansion of the predictor around the target latent gives

$$e_{k+1} \approx J_k e_k + \epsilon_k, \tag{10}$$

where

$$J_k = \left.\frac{\partial P_\phi(z, a_{t+k})}{\partial z}\right|_{z=z^{\mathrm{tgt}}_{t+k}} \tag{11}$$

is the predictor Jacobian evaluated at the target latent, and

$$\epsilon_k = P_\phi\left(z^{\mathrm{tgt}}_{t+k}, a_{t+k}\right) - z^{\mathrm{tgt}}_{t+k+1} \tag{12}$$

is the local transition error obtained when the predictor receives the target latent as input.

Thus, the previous latent readout error is an explicit input to the next transition and is transformed by the subsequent predictor Jacobian. Across multiple rollout steps, this creates a direct computational path through which intermediate readout errors can influence later predictions. This relation identifies the direct feedback mechanism induced by state–readout coupling, whose effect depends on the local predictor Jacobians and transition errors.

Methodology

Overview

Given the current latent representation z_t and candidate action sequence $a_{t:t+H-1}$, SRD treats the planning-horizon hidden state trajectory as the native output of the rollout predictor. Starting from a recurrent state initialized from z_t, the predictor advances the hidden state through the complete action sequence. Finally, a shared readout maps the resulting hidden state sequence to the latent trajectory used by the planner.

This formulation aligns three parts of the rollout computation. The hidden state sequence carries dynamics across the horizon, the shared readout exposes planner-facing predictions at every step, and the prediction objective supervises the complete readout trajectory. All other components, including the encoder architecture, latent space, candidate action generation, goal cost, CEM planner, execution protocol, and replanning loop, remain unchanged.

State–Readout Decoupling

SRD uses a separate recurrent hidden state as the recurrent carrier. A learned initializer maps the current latent representation to the initial hidden state:

$$h^{\text{init}} = g_\phi(z_t), \tag{13}$$

where g_ϕ denotes the state initializer. Each candidate action is mapped to an action embedding,

$$u_k = e_a(a_{t+k}), \qquad k = 0, \dots, H-1, \tag{14}$$

where e_a is the action encoder. Starting from h^{init}, the transition model advances the hidden state through the candidate action sequence:

$$h_0 = F_\phi(h^{\text{init}}, u_0), \tag{15}$$

$$h_k = F_\phi(h_{k-1}, u_k), \qquad k = 1, \dots, H-1. \tag{16}$$

A shared readout maps each post-action state to the latent space of the underlying backbone:

$$\hat{z}_{t+k+1} = R_\phi(h_k), \qquad k = 0, \dots, H-1. \tag{17}$$

The recurrent transition therefore contains only hidden states and action embeddings. In particular, $\hat{z}_{t+k+1}$ does not appear as an argument of the following transition $F_\phi(h_k, u_{k+1})$. Latent predictions remain available at every step for supervision and planner-side goal comparison, but they do not directly determine later state transitions. This removes the explicit latent readout feedback path identified in the preceding error feedback analysis.

SRD therefore changes more than the input interface of an otherwise one-step predictor. Given the complete candidate action sequence, its native computation produces the state trajectory $h_{0:H-1}$ and the corresponding latent trajectory $\hat{z}_{t+1:t+H}$. The planning horizon is thus part of the predictor's output structure rather than being constructed only by repeatedly calling a one-step latent transition.

Horizon-Aligned Readout Supervision

Because the native output of SRD is a planning-horizon state sequence, the predictor produces one planner-facing latent readout for every state along that sequence. We supervise the complete readout trajectory using latent-space mean squared error:

$$\mathcal{L}_{\text{SRD}} = \frac{1}{H} \sum_{k=1}^{H} \text{MSE}\big(\hat{z}_{t+k}, z^{\text{tgt}}_{t+k}\big)\,. \tag{18}$$

SRD replaces the original latent prediction objective while retaining all backbone-specific anti-collapse and representation-learning objectives:

$$\mathcal{L} = \mathcal{L}_{\text{SRD}} + \mathcal{L}_{\text{aux}}, \tag{19}$$

where $\mathcal{L}_{\text{aux}}$ denotes the original weighted sum of the remaining backbone-specific objectives, kept unchanged from LeWM or PLDM.

H is the native output span of the rollout predictor. This objective differs from attaching an auxiliary multi-step loss to a one-step autoregressive predictor: each supervised latent is the readout of a corresponding state in SRD's native hidden state trajectory. Supervising all readouts anchors the trajectory to the latent space used by the planner, while the hidden states remain responsible for carrying dynamics through the action horizon.

Matched Hidden State Rollout Predictor

The SRD interface naturally admits an efficient recurrent realization. Because only the hidden state is propagated across rollout steps, the transition model can advance a compact persistent state instead of repeatedly applying a one-step predictor to its previous planner-facing output. The GRU can provide a compact implementation matched to this persistent state interface. We therefore instantiate F_ϕ with an L_g-layer GRU, whose hidden state serves as the recurrent carrier of the SRD rollout. The GRU processes the full action sequence while preserving one persistent state across the horizon, and the shared readout converts all post-action states to planner-facing latent predictions. This structure matches the native SRD computation: the GRU produces the state trajectory, the readout produces the latent trajectory, and the horizon-aligned objective supervises that trajectory at every step.

Given the action-embedding sequence $U_{0:H-1} = e_a(a_{t:t+H-1})$, a sequence-level recurrent evaluation produces all post-action states and planner-facing latent predictions:

$$\begin{aligned} h_{0:H-1} &= \text{GRU}_\phi\big(U_{0:H-1}, h^{\text{init}}\big)\,, \\ \hat{z}_{t+1:t+H} &= R_\phi(h_{0:H-1}). \end{aligned} \tag{20}$$

This realization also makes the computational contrast between the two rollout formulations explicit. In the original AR implementation, the one-step Transformer predictor is invoked recursively over the action sequence, and each planner-facing latent prediction is returned as the input to the following transition. SRD instead initializes a persistent hidden state once and advances that state through the complete action sequence. The shared readout exposes a latent prediction after every state transition, but these predictions are not returned to the GRU. It changes the representation that carries the rollout: AR recursively propagates planner-facing latent predictions, whereas SRD propagates a hidden state trajectory and uses latent predictions only as readouts. Exact parameter counts and measured evaluation times are reported in Table 1 and Figure 2(c).

Local Transition Supervision

We study SRD-AR as an optional variant that supplements horizon-level SRD with **local one-step** transition supervision. An auxiliary predictor estimates

$$\hat{z}^{\text{AR}}_{t+1} = P^{\text{AR}}_\psi(z_t, a_t), \tag{21}$$

and is trained with

$$\mathcal{L}_{\text{AR}} = \text{MSE}\big(\hat{z}^{\text{AR}}_{t+1}, z^{\text{tgt}}_{t+1}\big)\,. \tag{22}$$

The resulting training objective is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{SRD}} + \lambda_{\text{AR}}\mathcal{L}_{\text{AR}} + \mathcal{L}_{\text{aux}}, \tag{23}$$

where λ_{AR} controls the contribution of the local transition objective.

At evaluation, SRD-AR uses the same decoupled rollout as SRD; the auxiliary local transition objective therefore does not reintroduce latent readout feedback into the test-time rollout.

Experiments

We organize the experiments around three questions:

1. Does SRD improve goal-conditioned planning relative to autoregressive latent rollout?
2. How do the two rollout formulations behave under the joint demands of longer predicted trajectories and more distant goals?
3. Which parts of SRD—horizon-level supervision, recurrent architecture, and the representation used as the rollout carrier—account for its observed behavior?

Experimental Setup

We evaluate on four visual goal-reaching tasks: TwoRoom, PushT, OGB-Cube, and Reacher. These tasks were selected to cover qualitatively different visual dynamics, from navigation to contact-rich and continuous-control settings. We use LeWM (Maes et al. 2026) and PLDM (Sobal et al. 2025), two recent end-to-end joint-embedding world model backbones, to test whether the effect of SRD generalizes across model families.

All methods use the same dataset splits, visual encoder architecture, target construction, goal representations, planner, and evaluation budget. Only the rollout predictor and its training objective are changed. Each checkpoint is evaluated for 50 closed-loop episodes; we report the mean and sample standard deviation across training seeds. Every model is trained for 10 epochs on the full training set, which is same as baseline. SRD and SRD-AR predict the planning-horizon trajectory with $H_{\text{train}} = 5$. The AR baseline retains the original one-step objective and repeated rollout protocol of each backbone.

We report goal-reaching success, rollout predictor parameters, evaluation time, and open-loop latent prediction error. The long-horizon TwoRoom study uses

$$H_{\text{eval}} \in \{5, 10, 15, 20\}. \tag{24}$$

H_{eval} denotes the number of model rollout steps, predicted for each candidate action sequence within a single planning update. We use a frame skip of five, such that each model action is held fixed for five consecutive environment steps. Thus, one model rollout step corresponds to five environment steps. Each value of H_{eval} defines a separate closed-loop evaluation setting. The goal distance is increased consistently with the evaluation horizon so that longer rollouts correspond to more distant navigation targets. All other evaluation settings, including the planner, candidate budget, replanning protocol, and evaluation budget, are held fixed.

Goal-Reaching Performance

Across the eight backbone–task settings, SRD improves success in seven while consistently reducing predictor size and evaluation time.

Table 1 compares repeated autoregressive rollout (AR) with state-readout decoupling rollout (SRD) across two backbones and four tasks. SRD improves goal-reaching success in seven of the eight backbone–task settings. Its largest improvement occurs on LeWM-TwoRoom, where success increases from 88.67% to 96.00%, a gain of 7.33 percentage points. LeWM-Reacher and PLDM-PushT each improve by 6.67 points. On PLDM, SRD improves over AR on all four tasks, although the gains on TwoRoom and Reacher are comparatively small at 1.00 and 0.67 points, respectively.

LeWM-PushT is the only setting in which SRD underperforms AR: success decreases from 94.67% to 87.33%, a reduction of 7.34 points. PushT emphasizes precise local contact geometry and object pose, suggesting that the stability of a horizon-level state rollout does not always compensate for reduced fidelity in short-range contact transitions. The contrasting PLDM-PushT result, where SRD improves by 6.67 points, further indicates that this trade-off depends on both the task and the backbone.

SRD-AR adds a one-step transition objective to the decoupled rollout. It achieves the highest success rate in four of the eight settings: LeWM-TwoRoom, LeWM-OGB-Cube, PLDM-TwoRoom, and PLDM-PushT. Its largest gain occurs on PLDM-PushT, where success increases from 78.00% to 89.33%, an improvement of 11.33 percentage points. SRD-AR also reaches 99.33% success on PLDM-TwoRoom, the highest absolute success rate in the table. The auxiliary objective is not uniformly beneficial: SRD remains stronger on LeWM-PushT, LeWM-Reacher, PLDM-OGB-Cube, and PLDM-Reacher. We therefore use SRD as the default, while SRD-AR is an optional extension that may improve success at the cost of additional training parameters and computation.

In addition to its success-rate improvements, SRD consistently reduces model size and evaluation cost. Its predictor uses 53.0–58.2% fewer parameters than AR across all eight settings, with the largest reduction on LeWM-TwoRoom. SRD also evaluates faster in every setting, reducing evaluation time by 24.2–62.8% and by approximately 44% on average. The largest fixed-horizon runtime reduction occurs on PLDM-TwoRoom, where evaluation time decreases from 33.21 to 12.35 seconds. SRD-AR retains substantial runtime savings, but its auxiliary predictor increases the parameter count: it remains smaller than AR for LeWM and becomes slightly larger than AR for PLDM.

Long-Horizon Diagnostic on TwoRoom

As the planning horizon and goal distance increase jointly, SRD degrades more slowly than autoregressive rollout. Its widening success advantage is accompanied by lower open-loop latent error and substantially slower growth in evaluation time.

We use the LeWM backbone on TwoRoom as a diagnostic setting because its navigation dynamics avoid contact-induced discontinuities and fine-grained object-pose transitions. This provides a controlled setting for examining how the rollout formulations behave in increasingly difficult long-range planning problems. The evaluation horizon and goal distance are increased together, while the planner, candidate budget, replanning protocol, and evaluation budget are held fixed. We use this protocol as a combined stress test of robustness under the joint demands of longer predicted trajectories and more distant goals.

Success Rate Success decreases for every method as the evaluation horizon and goal distance increase, which is

Backbone	Task	Method	Success Rate (%)	ΔS (pp)	Parameters (M)	ΔP (%)	Time (s)	ΔT (%)
LeWM	TwoRoom	AR	88.67 ± 4.16	–	18.03	–	25.25	–
		SRD	96.00 ± 1.63	$+7.33$	**7.54**	$\mathbf{-58.2}$	**13.75**	$\mathbf{-45.5}$
		SRD-AR	$\mathbf{97.33 \pm 1.15}$	$\mathbf{+8.66}$	11.51	-36.2	14.42	-42.9
	PushT	AR	$\mathbf{94.67 \pm 3.06}$	–	18.03	–	33.78	–
		SRD	87.33 ± 2.31	-7.34	**7.69**	$\mathbf{-57.3}$	**17.12**	$\mathbf{-49.3}$
		SRD-AR	86.67 ± 2.31	-8.00	11.51	-36.2	17.93	-46.9
	OGB-Cube	AR	73.33 ± 6.11	–	18.03	–	47.92	–
		SRD	76.67 ± 1.15	$+3.34$	**7.69**	$\mathbf{-57.3}$	**36.32**	$\mathbf{-24.2}$
		SRD-AR	$\mathbf{79.33 \pm 4.16}$	$\mathbf{+6.00}$	11.51	-36.2	37.44	-21.9
	Reacher	AR	81.33 ± 4.62	–	18.03	–	55.58	–
		SRD	$\mathbf{88.00 \pm 5.29}$	$\mathbf{+6.67}$	**7.69**	$\mathbf{-57.3}$	27.87	-49.9
		SRD-AR	87.33 ± 4.16	$+6.00$	11.51	-36.2	**26.78**	$\mathbf{-51.8}$
PLDM	TwoRoom	AR	97.00 ± 1.21	–	18.03	–	33.21	–
		SRD	98.00 ± 0.00	$+1.00$	**8.33**	$\mathbf{-53.8}$	12.35	-62.8
		SRD-AR	$\mathbf{99.33 \pm 1.15}$	$\mathbf{+2.33}$	19.12	$+6.0$	**12.28**	$\mathbf{-63.0}$
	PushT	AR	78.00 ± 5.07	–	18.03	–	38.32	–
		SRD	84.67 ± 3.06	$+6.67$	**8.48**	$\mathbf{-53.0}$	20.00	-47.8
		SRD-AR	$\mathbf{89.33 \pm 1.15}$	$\mathbf{+11.33}$	19.27	$+6.9$	**18.04**	$\mathbf{-52.9}$
	OGB-Cube	AR	65.00 ± 2.90	–	18.83	–	49.11	–
		SRD	$\mathbf{67.33 \pm 4.16}$	$\mathbf{+2.33}$	**8.48**	$\mathbf{-55.0}$	**33.66**	$\mathbf{-31.5}$
		SRD-AR	65.33 ± 2.31	$+0.33$	19.27	$+2.3$	34.19	-30.4
	Reacher	AR	78.00 ± 5.80	–	18.83	–	45.81	–
		SRD	$\mathbf{78.67 \pm 1.15}$	$\mathbf{+0.67}$	**8.48**	$\mathbf{-55.0}$	26.65	-41.8
		SRD-AR	78.00 ± 5.30	0.00	19.27	$+2.3$	**26.26**	$\mathbf{-42.7}$

Table 1: Comparison of autoregressive rollout (AR) and state-readout decoupling rollout (SRD) across LeWM and PLDM. Boldface marks the highest success rate, the fewest predictor parameters, or the shortest evaluation time within each backbone–task setting, together with the corresponding change relative to AR.

shown in Figure 2(a). However, AR degrades substantially faster than SRD and SRD-AR. From $H = 5$ to $H = 20$, AR success falls from approximately 88% to 14%, whereas SRD decreases from 96% to 46%. SRD-AR follows a similar trend, starting at approximately 98% and retaining 46% success at the longest horizon.

The resulting advantage of SRD over AR grows from approximately 8 percentage points at $H = 5$ to 32 points at $H = 20$, and exceeds 30 points at both $H = 15$ and $H = 20$. Across all evaluated horizons, SRD degrades more slowly than AR under the combined demands of longer predicted trajectories and more distant goals.

The following open-loop analysis examines whether this widening closed-loop gap is accompanied by a corresponding difference in latent prediction quality.

Open-Loop Latent Error For each model, we initialize the rollout from its ground-truth encoded context and provide the subsequent action sequence from the held-out dataset, without incorporating future observations. Since one model rollout step corresponds to five environment steps, raw step $s = 5k$. We report the mean latent-space MSE across N evaluation trajectories:

$$E_k = \frac{1}{N} \sum_{i=1}^{N} \mathrm{MSE}\left(\hat{z}_{t+k}^{(i)}, z_{t+k}^{\mathrm{tgt},(i)}\right). \tag{25}$$

Figure 2(b) shows that SRD and SRD-AR maintain lower latent error than AR throughout the measured open-loop trajectory. The separation is largest around raw step 65, where SRD reduces MSE from 0.348 to 0.218, a reduction of 37.3%. The gap narrows toward the end of the rollout, but SRD still reduces terminal MSE from 0.417 to 0.345, or 17.2%.

The lower open-loop error is consistent with the widening closed-loop success advantage. SRD remains closer to the target latent trajectory while also retaining higher planning success at longer horizons.

Evaluation Time Figure 2(c) shows that the runtime gap widens rapidly as the rollout horizon increases. At $H = 20$, AR requires 515 seconds, compared with 68 seconds for SRD and 66 seconds for SRD-AR. The corresponding reductions are 86.8% and 87.2%, respectively. Thus, SRD and SRD-AR remain close in runtime, while both grow substantially more slowly than the autoregressive baseline.

This pattern reflects the matched implementations evaluated here. AR constructs each candidate trajectory by invoking its one-step Transformer predictor separately at every rollout step. SRD instead initializes a persistent hidden state once, advances it through the complete action sequence, and reads out the resulting latent trajectory. The widening gap therefore reflects both the computation required at each step and the overhead of repeatedly invoking the AR predictor.

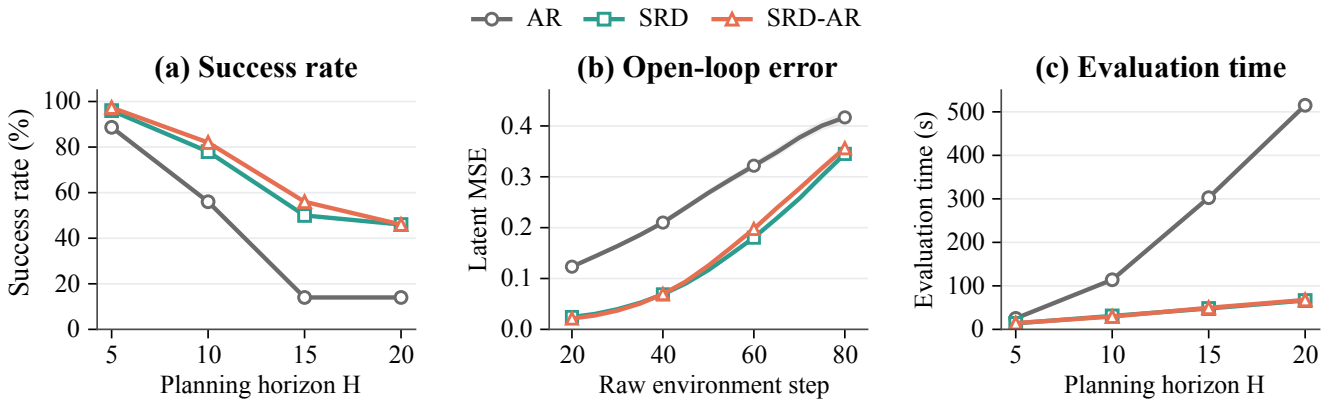

Figure 2: Long-horizon analysis on TwoRoom. **(a)** Goal-reaching success rate under different planning horizons. **(b)** Open-loop latent prediction error over raw environment steps. **(c)** Evaluation time under different planning horizons.

Method	Success Rate (%)	ΔS (pp)	Parameters (M)	ΔP (%)	Time (s)	ΔT (%)
AR baseline	88.67 ± 4.16	–	18.034	–	25.25	–
AR-MS	88.00 ± 2.00	-0.67	18.034	0.0	31.98	$+26.6$
AR-GRU	94.00 ± 0.00	$+5.33$	7.577	-58.0	17.03	-32.6
SRD-Transformer	93.33 ± 1.15	$+4.67$	18.036	$+0.01$	18.58	-26.4
SRD	96.00 ± 1.63	$+7.33$	**7.540**	**-58.2**	**13.75**	**-45.5**
SRD-AR	**97.33 ± 1.15**	**$+8.66$**	11.510	-36.2	14.42	-42.9

Table 2: Alternative rollout formulations on TwoRoom. The variants retain selected properties of SRD while changing the representation that is recursively propagated. Boldface marks the highest success rate, the fewest parameters, and the shortest evaluation time, together with the corresponding changes relative to AR.

The three diagnostics show a consistent long-horizon pattern: SRD retains higher planning success, remains closer to the target latent trajectory in open loop, and grows more slowly in evaluation cost. The first two observations are consistent with removing the direct path through which planner-facing latent errors are returned to later transitions. The runtime result additionally reflects the structurally matched GRU implementation and the repeated Transformer calls used by the AR baseline.

Ablation Analysis

Table 2 separates three components of SRD: horizon-level supervision, recurrent architecture, and the representation used as the rollout carrier. In this TwoRoom setting, no isolated modification reproduces the complete SRD result.

Adding multi-step supervision alone has little effect. AR-MS retains the autoregressive latent rollout and achieves 88.00% success, compared with 88.67% for AR, while increasing evaluation time. Horizon-level supervision therefore does not account for the SRD gain when planner-facing latent predictions continue to carry the rollout dynamics.

Changing either the predictor architecture or the rollout carrier recovers part of the improvement. AR-GRU retains latent feedback but raises success to 94.00% and reduces evaluation time by 32.6%. SRD-Transformer instead preserves the decoupled hidden state rollout and reaches 93.33% success with approximately the same parameter count as AR. These two variants show that both recurrent prediction and the choice of rollout carrier contribute, while neither alone matches the complete SRD configuration.

The complete GRU-based SRD combines a persistent hidden state rollout, horizon-aligned readout supervision, and a recurrent predictor matched to that state interface. It reaches 96.00% success while using 58.2% fewer predictor parameters and 45.5% less evaluation time than AR. SRD-AR further increases success to 97.33% by adding local one-step supervision, but introduces additional training-time parameters. We therefore use SRD as the default configuration and treat SRD-AR as an optional extension.

Conclusion

We propose State–Readout Decoupling (SRD), which replaces autoregressive latent rollout with a planning-horizon hidden state trajectory and generates planner-facing latent predictions through a separate readout, preventing predicted latents from being directly reused in subsequent transitions. With a structurally matched GRU predictor and horizon-aligned supervision, SRD achieves higher goal-reaching success in seven of eight LeWM and PLDM settings, uses substantially fewer predictor parameters, reduces evaluation time, and degrades more slowly under long-horizon planning. These results support organizing reconstruction-free latent world models around horizon-level hidden state rollout rather than recursively propagated latent predictions.

References

Assran, M.; Duval, Q.; Misra, I.; Bojanowski, P.; Vincent, P.; Rabbat, M.; LeCun, Y.; and Ballas, N. 2023. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15619–15629.

Bardes, A.; Garrido, Q.; Ponce, J.; Chen, X.; Rabbat, M.; LeCun, Y.; Assran, M.; and Ballas, N. 2024. Revisiting Feature Prediction for Learning Visual Representations from Video. *Transactions on Machine Learning Research*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28, 1171–1179. Curran Associates, Inc.

Chen, J.; Xu, L.; Venugopal, A.; and Schneider, J. 2025. Policy-Driven World Model Adaptation for Robust Offline Model-based Reinforcement Learning. *arXiv preprint arXiv:2505.13709*.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Ding, Z.; Lai, X.; Chen, W.; Zhang, X.-P.; and Chen, J. 2026. CausalVAE as a Plug-in for World Models: Towards Reliable Counterfactual Dynamics. *arXiv preprint arXiv:2604.07712*.

Finn, C.; and Levine, S. 2017. Deep Visual Foresight for Planning Robot Motion. In *IEEE International Conference on Robotics and Automation*, 2786–2793.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2170–2179. PMLR.

Ha, D.; and Schmidhuber, J. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2450–2462. Curran Associates, Inc.

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2020. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2555–2565. PMLR.

Hafner, D.; Lillicrap, T.; Norouzi, M.; and Ba, J. 2021. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2025. Mastering Diverse Control Tasks through World Models. *Nature*, 640: 647–653.

Hafner, D.; Yan, W.; and Lillicrap, T. 2025. Training Agents Inside of Scalable World Models. *arXiv preprint arXiv:2509.24527*.

Hansen, N.; Su, H.; and Wang, X. 2024. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations*.

Hansen, N. A.; Su, H.; and Wang, X. 2022. Temporal Difference Learning for Model Predictive Control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 8387–8406. PMLR.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 32, 12519–12530. Curran Associates, Inc.

Lambert, N.; Amos, B.; Yadan, O.; and Calandra, R. 2020. Objective Mismatch in Model-Based Reinforcement Learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, 761–770. PMLR.

Li, W.; Li, G.; Maeda, K.; Ogawa, T.; and Haseyama, M. 2026. Predictive but Not Plannable: RC-aux for Latent World Models. *arXiv preprint arXiv:2605.07278*.

Maes, L.; Le Lidec, Q.; Scieur, D.; LeCun, Y.; and Balestriero, R. 2026. LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels. *arXiv preprint arXiv:2603.19312*.

Oh, J.; Singh, S.; and Lee, H. 2017. Value Prediction Network. In *Advances in Neural Information Processing Systems*, volume 30, 6118–6128. Curran Associates, Inc.

Rao, P.; Zhang, W.; Balestriero, R.; LeCun, Y.; and Loianno, G. 2026. SkyJEPA: Learning Long-Horizon World Models for Zero-Shot Sim-to-Real Control of Quadrotors. *arXiv preprint arXiv:2606.23444*.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; Lillicrap, T.; and Silver, D. 2020. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839): 604–609.

Schwarzer, M.; Anand, A.; Goel, R.; Hjelm, R. D.; Courville, A.; and Bachman, P. 2021. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *International Conference on Learning Representations*.

Silver, D.; van Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D.; Rabinowitz, N.; Barreto, A.; and Degris, T. 2017. The Predictron: End-to-End Learning and Planning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3191–3199. PMLR.

Sobal, U.; Zhang, W.; Cho, K.; Balestriero, R.; Rudner, T. G. J.; and LeCun, Y. 2025. Learning from Reward-Free Offline Data: A Case for Planning with Latent Dynamics Models. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.

Talvitie, E. 2017. Self-Correcting Models for Model-Based Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1): 2597–2603.

Zhou, G.; Pan, H.; LeCun, Y.; and Pinto, L. 2025. DINO-WM: World Models on Pre-Trained Visual Features Enable Zero-Shot Planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 79115–79135. PMLR.