# From Word Tokens to World Tokens: Making World Models Harder to Game

Dario Fumarola
Amazon Web Services

## Abstract

A world model is trained to predict what happens next, but a planner uses it differently: it rolls the model forward thousands of times, searching for the best possible future. That search is adversarial. Given enough rollouts, the planner will find whatever the model gets slightly wrong and exploit it. In dense multi-object scenes the failure is especially vivid—the imagined agent glides through a crowd because the model quietly forgets that two nearby objects should be pushing back against each other.

This paper asks a narrow question: in contact-rich scenes with local interactions, can a learned simulator be made harder to cheat without sacrificing predictive accuracy? The approach has two parts. The first is representational: instead of recomputing interactions from scratch at every step, the model stores a persistent memory slot for each nearby pair of objects, with directed relation states that carry interaction context forward in time. An explicit lifecycle ensures that creating or destroying these slots never silently injects energy into the system. The second part is a diagnostic: a scalar budget $H(z)$, anchored to measurable quantities on real data, that should never grow for free when the planner imagines a future with zero input. If it does, something has been invented that reality would not provide.

The two ideas reinforce each other. Persistent pair state makes interaction dropout harder; the budget test makes the consequences visible. On a dense-disc benchmark using an act-then-hold protocol—where the planner chooses a short burst of actions, then the system coasts—the full model reduces the exploitability gap by roughly 80% relative to an entity-only baseline, at matched short-horizon prediction quality.

## 1 Introduction

A learned world model is trained as a predictor but deployed as a simulator under search. A planner rolls it forward many times, searches over futures, and repeatedly reuses the same model errors. Earlier work already showed that good one-step prediction does not reliably imply good downstream planning [1, 2, 3]. Modern systems such as PlaNet, Dreamer, and TD-MPC2 have greatly expanded what world models can do [5, 6, 7, 8], but the planner remains but the planner stresses the model far harder than the training objective does.

The pressure is especially sharp in dense multi-object scenes. The imagined agent threads through a crowd because interaction structure silently drops out. A short burst of input appears to create momentum that survives after control stops. Two nearby objects are treated as independent for just long enough to open a path that reality will not provide. Optimization is good at finding exactly these loopholes.

This paper takes a deliberately narrow route through that problem. It does not try to recover full physics. It asks a concrete question: in contact-rich systems with predominantly local interactions, can exploitability be reduced without requiring large changes in short-horizon predictive accuracy?

The proposal has two parts. First, nearby interactions are treated as persistent state rather than transient computation. The model stores one entity token per object and one persistent pair slot per nearby unordered pair, with directed relation memories to carry local interaction context over time. Second, reliability is evaluated with a planner-aligned *free-lunch test*: a storage-like scalar $H(z)$, built from a measurable kinetic/overlap proxy plus a learned nonnegative residual, should not systematically increase under zero input when measured under the same discrete rollout map used by planning. This yields both a cheap stress test and a local warning light for suspicious imagined futures.

The claim is intentionally narrow and falsifiable. Persistent relation state, explicit interaction lifecycle, anchored geometry, and passive-tail drift control should reduce exploitability gap at broadly matched predictive quality. If the diagnostics fail to track exploitability, or if exploitability falls only because task performance collapses, then the proposal has not worked. Figure 1 summarizes the target regime and the evaluation flow.

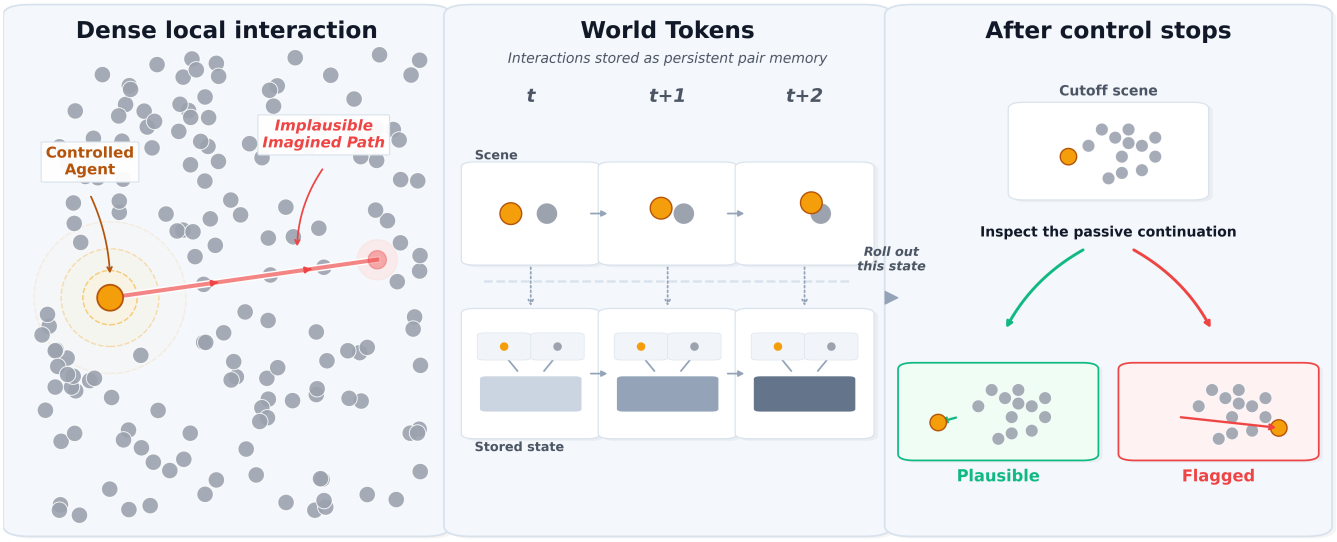


Figure 1: Persistent interaction state in the target regime. Left: in dense local scenes, a transiently modeled controller can exploit dropped interactions to imagine implausible paths. Center: the proposal stores nearby interactions as persistent pair memory alongside entity tokens. Right: after control stops, the passive continuation of the imagined state is inspected; plausible continuations dissipate, while suspicious ones are flagged by the passive-tail test.

## 2 Scope and Positioning

The scope is restricted to a state-based benchmark with one controlled agent moving through a dense scene of passive discs under damping and short-range collisions. Planner-facing simulator reliability is easiest to study cleanly when perception is not the dominant source of error; a pixel extension is sketched in Appendix C. In this benchmark, the observed state $x_t = \{p_i, v_i, a_i\}$ is already Markov, so persistent pair memory is not introduced to recover hidden physical state from the environment. It is an architectural bias meant to help an approximate learned simulator preserve local interaction structure under repeated search.

The flagship damped-disc benchmark is mostly symmetric, so the directed pair memories should be read as a forward-compatible design choice rather than as a claim that directionality is essential in every setting. They become more useful in asymmetric settings such as yielding, pursuit, or occlusion order, which motivate the broader formulation.

The work sits at the intersection of model-based reinforcement learning, object-centric world models, and reliability under planning. Classical world-model papers showed that latent dynamics can support planning from images or compact state [4, 5, 6]. Later work emphasized that the training objective and the control objective are not the same, and that this mismatch can remain even when predictive losses look strong [2, 3]. BIRD is particularly close in spirit to the present paper in that it explicitly tries to reduce imagination-to-reality discrepancy [9]. Recent planner-alignment work goes further by trying to reduce the train–test gap encountered by the planner itself [10].

The representational side of the proposal is closest to object-centric and interaction-explicit models. Interaction Networks and Graph Networks made relational inductive bias central to learned physics [11, 12]. Slot-based encoders made object-centric perception practical from images [13]. More recent world-model papers argue that relational structure should be made explicit rather than left implicit inside holistic latents. SOLD shows that object-centric latent dynamics can substantially help relational manipulation from pixels [14]. FIOC-WM explicitly models object interactions inside an object-centric world model [15]. The novelty claim here is therefore combinatorial rather than absolute: persistent local relation state is paired with an inert pair-slot lifecycle, anchored geometry, and a planner-aligned passive-tail test aimed directly at exploitability under search.

The free-lunch test is inspired by storage-function and passivity ideas, but the emphasis here is discrete and operational rather than continuous and axiomatic. Hamiltonian and port-Hamiltonian neural models provide useful examples of how structure can constrain dynamics [16, 17, 18]. The present paper borrows the one-way intuition but weakens the claim substantially. The budget $H$ is not physical energy. It is a planner-facing diagnostic. Its meaning comes from three conditions: it upper-bounds a measurable base proxy on the benchmark, it matches that proxy on real support, and it should not grow for free during passive rollouts of the learned simulator.

Finally, the paper is complementary to runtime reliability monitoring rather than a replacement for it. Methods

such as SODA-MPC use learned uncertainty and conformal calibration to detect out-of-distribution situations online [19]. Those methods and the present one attack different parts of the same problem. SODA-MPC detects when prediction should not be trusted; the free-lunch test detects whether a planner can create budget that reality will not provide.

# 3 Persistent Interaction State

Let $\mathcal{V}$ denote the set of entities. In the state-based benchmark, each entity $i$ has observed state

$$x_i = (p_i, v_i, a_i), \tag{1}$$

where $p_i \in \mathbb{R}^2$ is position, $v_i \in \mathbb{R}^2$ is velocity, and $a_i$ collects static attributes such as radius, mass, or type. The full observed state at time $t$ is $x_t = \{x_i(t)\}_{i\in\mathcal{V}}$.

The learned simulator carries a richer token state

$$z_t = \{e_i(t)\}_{i\in\mathcal{V}} \cup \{r_{i\to j}(t), r_{j\to i}(t)\}_{\{i,j\}\in\mathcal{E}_{\text{list}}(t)}, \tag{2}$$

where $e_i \in \mathbb{R}^d$ is an entity token and each stored unordered pair $\{i, j\}$ carries two directed relation memories.

**Filtering real trajectories.** Because relation memories are latent, real trajectories need a filter as well as a rollout map. We write

$$z_t^{\text{real}} = \Gamma_\theta(x_{0:t}, u_{0:t-1}) \tag{3}$$

for the filtered token state obtained by teacher forcing on observed entity states while updating pair memories over time. In the state-based setting this is simple: entity tokens are refreshed from the current observed state, while relation memories are updated recurrently using the same sparse pair structure as the rollout model. Planning uses a separate open-loop rollout map

$$z_{t+1} = \Phi_{\Delta t}(z_t, u_t), \tag{4}$$

which is exactly the map later used inside the planner and in the free-lunch test.

**Anchored geometry.** Each entity token is decoded to a predicted observable state

$$\hat{x}_i = D_x(e_i), \qquad \hat{p}_i = \Pi_p \hat{x}_i, \tag{5}$$

and the decoded position $\hat{p}_i$ is used for neighborhood construction and task evaluation. This anchor is important. It prevents the model from "solving" difficult interactions by warping latent geometry until nearby objects no longer look nearby to the dynamics module.

**Sparse pair slots.** Pair slots are short-range. For any stored pair,

$$w_{ij} = \kappa\left(\frac{\|\hat{p}_i - \hat{p}_j\|_2}{r_{\text{cut}}}\right), \tag{6}$$

where the compact-support gate is

$$\kappa(s) = \begin{cases} (1-s)^4(4s+1), & 0 \le s < 1, \\ 0, & s \ge 1. \end{cases} \tag{7}$$

This gate is exactly zero outside the cutoff and has zero slope at the boundary, which reduces artifacts when pair slots enter or leave the active region.

**Persistent updates.** A simple token update block is

$$m_{i\leftarrow j} = w_{ij}\, \psi_\theta(e_i, e_j, r_{i\to j}, r_{j\to i}), \tag{8}$$

$$e_i^+ = e_i + f_\theta\left(e_i, u_i, \sum_{j\neq i} m_{i\leftarrow j}\right), \tag{9}$$

$$r_{i\to j}^+ = r_{i\to j} + w_{ij}\, g_\theta(e_i, e_j, r_{i\to j}, u_i, u_j). \tag{10}$$

The exact neural block is not essential. The important design choice is that pair memory persists across steps. An ephemeral-message baseline recomputes pair features from the current entities alone and discards them immediately; the present model keeps a dedicated pair state and lets the planner face its consequences over time.

**Lifecycle of pair slots.** Persistent pair memory is useful only if creation and deletion of pair slots do not themselves introduce hidden impulses. The neighbor list therefore uses a conservative annulus. A pair slot is created when $\|\hat{p}_i - \hat{p}_j\| \le r_{\text{list}}$ for some $r_{\text{list}} > r_{\text{cut}}$, and deleted only when $\|\hat{p}_i - \hat{p}_j\| > r_{\text{list}}$. Since $w_{ij} = 0$ whenever the distance exceeds $r_{\text{cut}}$, all allocation and deletion events happen while the slot is inactive. A new slot is initialized by a shared map

$$(r_{i\to j}, r_{j\to i}) \leftarrow \phi_{\text{init}}(x_i, x_j). \tag{11}$$

Optional time-to-live caching is allowed, but only with reactivation inside a restore radius no larger than $r_{\text{cut}}$. Appendix B gives a concrete algorithm.

**Proposition 1** (Inert-slot contract)**.** *Assume every slot-local contribution to the rollout map, every slot-local readout, and every slot-local residual-budget term is multiplied by the gate $w_{ij}$. Then any stored pair slot with $w_{ij} = 0$ contributes exactly zero to the entity-token dynamics, exactly zero to the residual budget, and exactly zero to any decoded edge quantity. Consequently, allocating or deleting such an inactive slot leaves the dynamics of all existing entity tokens unchanged.*

In practice, this is what makes dynamic-graph bookkeeping safe: the model cannot hide instability inside the moment when pair slots appear or disappear.

# 4 A Planner-Aligned Free-Lunch Test

The second half of the proposal is a discrete reliability contract. The contract is designed for the benchmark regime rather than for all possible environments.

**A measurable base proxy.** For contact-rich passive discs, a useful observable proxy is

$$V(x) = \sum_{i\in\mathcal{V}} \frac{1}{2} m_i \|v_i\|_2^2 + \lambda_{\text{ov}} \sum_{i<j} \rho\big(d_{ij}^{\min} - \|p_i - p_j\|_2\big)\,, \tag{12}$$

where $d_{ij}^{\min}$ is the no-overlap distance for the pair and $\rho$ is a smooth nonnegative penalty, for example $\rho(s) = \text{softplus}(s)^2$. The first term is kinetic budget. The second term is zero on physically valid non-overlapping configurations and positive when the imagined rollout tries to "ghost" objects through one another. On the real passive benchmark, with no external input and damping present, this quantity should not systematically increase.

**A learned budget anchored on support.** The learned storage-like scalar is

$$H(z) = V(\hat{x}(z)) + R(z), \tag{13}$$

where $\hat{x}(z)$ is the decoded observable state and the residual is

$$R(z) = \sum_{i\in\mathcal{V}} h_i(e_i) + \sum_{\{i,j\}\in\mathcal{E}_{\text{list}}} w_{ij}\, h_{ij}(e_i, e_j, r_{i\to j}, r_{j\to i}), \tag{14}$$

with $h_i, h_{ij} \geq 0$ enforced by construction, for example through softplus outputs. This gives two useful properties immediately. First, $H(z) \geq V(\hat{x}(z))$ everywhere, so the learned budget cannot understate the measurable proxy. Second, the residual has an additive local structure, which later allows a local warning light around the controlled agent.

To keep the budget from from validating itself, it is anchored on real support. If $z_t^{\text{real}}$ is the filtered token state of a real trajectory, then

$$\mathcal{L}_{\text{anchor}} = \sum_t \big(H(z_t^{\text{real}}) - V(x_t)\big)^2 \tag{15}$$

pushes the residual toward zero on actual data states. In other words, the learned budget is free to rise away from support, but on real support it must agree with a measurable quantity.

**Positive drift under the planner map.** The central diagnostic is defined directly on the planner's own rollout operator.

**Definition 1** (Free-lunch test)**.** *For a model state $z_t$ and zero input, define*

$$\Delta H_t^+ = \max(0,\, H(\Phi_{\Delta t}(z_t, 0)) - H(z_t))\,. \tag{16}$$

*A model passes the free-lunch test to the extent that cumulative positive drift remains small over long passive rollouts of the learned simulator.*

The test is one-way: it asks whether the model invents budget, not whether the model is correct.

**A local warning light.** Using the additive residual in Eq. (14), define the residual density around a controlled entity $k$ by

$$\bar{R}_k(z) = \frac{R_k^{\text{node}}(z) + R_k^{\text{edge}}(z)}{1 + \sum_{j\neq k} w_{kj}}, \tag{17}$$

where $R_k^{\text{node}}$ is the node residual for entity $k$ and $R_k^{\text{edge}}$ is the sum of incident edge residuals. The warning light uses *residual* budget rather than full budget. This is deliberate: legitimate high-speed motion should not automatically be treated as suspicious, while unexplained extra budget in a dense neighborhood should be.

**Why drift matters on the passive tail.** The experiments use an *act-then-hold* protocol: the planner is allowed a bounded action prefix, after which the system is forced into a zero-input tail. This isolates exactly the regime where imagined free lunch should become visible.

**Proposition 2** (Passive-tail gap decomposition)**.** *Consider any common start state $x_s$ from which both the real benchmark and the learned model are rolled forward with zero input for $t = s, \ldots, T-1$. Assume the real passive benchmark satisfies*

$$V(x_{t+1}^{\text{real}}) \leq V(x_t^{\text{real}}), \tag{18}$$

*and the filtered model state at the start of the tail obeys $H(z_s^{\text{real}}) = V(x_s)$. Let $J$ be any passive-tail objective satisfying*

$$J(\tau) \leq a + b\, V(x_T) + L \sum_{t=s}^{T} \|p_t - p_t^{\text{real}}\|_2 \tag{19}$$

*for constants $a, b, L \geq 0$. Then*

$$\text{Gap}(J) \leq b \sum_{t=s}^{T-1} \Delta H_t^+ + L \sum_{t=s}^{T} \|p_t^{\text{imag}} - p_t^{\text{real}}\|_2 + \epsilon_{\text{real}}(J), \tag{20}$$

*where the real-tail slack is*

$$\epsilon_{\text{real}}(J) = a + b\, V(x_s) - J(\tau^{\text{real}}) \geq 0. \tag{21}$$

*Equivalently, the imagined passive-tail surplus above the common-start baseline $a + bV(x_s)$ is upper-bounded by the first two terms alone.*

The proof is short and placed in Appendix A. The proposition does not solve general control. It instead decomposes passive-tail exploitability into a free-lunch term controlled by cumulative positive drift, a trajectory-mismatch term, and a real-tail slack term that disappears when the comparison is normalized to the common hold-start baseline.

# 5 Learning and Planning

Training is supervised on observable state, not on latent relation memories. The relation memories are internal recurrent state and are shaped indirectly through state prediction, edge-topology supervision, real-support anchoring, and passive-tail drift control.

**State prediction and topology.** Starting from a filtered real state $z_t^{\text{real}}$, the rollout map is unrolled for $K$ steps under recorded actions. The basic prediction loss is

$$\mathcal{L}_x = \sum_t \sum_{\tau=1}^{K} \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \|x_i(t+\tau) - \hat{x}_i(t+\tau)\|_W^2, \quad (22)$$

where $W$ weights positions and velocities appropriately. To keep interaction topology visible, we also supervise a simple active-edge label

$$c_{ij}(t) = \mathbb{I}\{\|p_i(t) - p_j(t)\|_2 \le d_{ij}^{\min} + \epsilon_c\}, \quad (23)$$

using a readout $\hat{c}_{ij} = \sigma(\chi(e_i, e_j, r_{i\to j}, r_{j\to i}))$ and binary cross-entropy loss

$$\mathcal{L}_{\text{edge}} = \sum_t \sum_{\tau=1}^{K} \frac{1}{|\mathcal{P}_{t+\tau}|} \sum_{\{i,j\} \in \mathcal{P}_{t+\tau}} \text{BCE}\big(\hat{c}_{ij}(t+\tau), c_{ij}(t+\tau)\big). \quad (24)$$

where $\mathcal{P}_t$ is the current candidate-pair set.

**Anchoring on real support.** The anchor loss in Eq. (15) is applied to filtered real states. This matters because it prevents the residual budget from becoming free-floating even if the model learns a weakly drifting scalar on its own rollouts.

**Two passive-tail penalties.** A single imagined-tail penalty is not enough; the budget should also behave properly on real passive data. We therefore use two complementary losses.

For real passive segments filtered from data or produced by simulator resets,

$$\mathcal{L}_{\text{tail}}^{\text{data}} = \sum_{(s,K_0)\in\mathcal{D}_0} \sum_{\tau=0}^{K_0-1} \Big[\max\big(0,\, H(z_{s+\tau+1}^{\text{real}}) - H(z_{s+\tau}^{\text{real}})\big)\Big]^2. \quad (25)$$

For imagined passive tails under the planner map,

$$\mathcal{L}_{\text{tail}}^{\text{model}} = \sum_{s} \sum_{\tau=0}^{K_0-1} \Big[\max\big(0,\, H(\hat{z}_{s+\tau+1}^{(0)}) - H(\hat{z}_{s+\tau}^{(0)})\big)\Big]^2. \quad (26)$$

where $\hat{z}_{s+\tau}^{(0)}$ denotes a zero-input rollout from the filtered start state $z_s^{\text{real}}$. The first loss keeps the budget honest on real support. The second aligns the diagnostic with the operator the planner will actually stress.

**Algorithm 1** Training step (high-level)

1: **Input:** real windows $(x_{t:t+K}, u_{t:t+K-1})$, passive-tail windows $\mathcal{D}_0$, horizon lengths $K, K_0$
2: Filter each real window to obtain start states $z_t^{\text{real}} = \Gamma_\theta(x_{0:t}, u_{0:t-1})$
3: Roll out $\Phi_{\Delta t}$ for $K$ steps under recorded actions and compute $\mathcal{L}_x, \mathcal{L}_{\text{edge}}$
4: Evaluate $\mathcal{L}_{\text{anchor}}$ on filtered real states
5: Compute $\mathcal{L}_{\text{tail}}^{\text{data}}$ on filtered real passive segments
6: Roll out zero-input passive tails from the same filtered start states and compute $\mathcal{L}_{\text{tail}}^{\text{model}}$
7: Update parameters using Eq. (27)

The total objective is

$$\mathcal{L} = \mathcal{L}_x + \lambda_e \mathcal{L}_{\text{edge}} + \lambda_a \mathcal{L}_{\text{anchor}} + \lambda_r \mathcal{L}_{\text{tail}}^{\text{data}} + \lambda_m \mathcal{L}_{\text{tail}}^{\text{model}}. \quad (27)$$

In practice, the passive-tail batches are cheap because they are short and do not require full task rollouts.

**Planning.** At test time, a standard receding-horizon cost is used for the task itself, with the local residual warning light added as a soft bias:

$$\min_{u_{0:T-1}} \quad \sum_{t=1}^{T} \Big(\ell_{\text{task}}(\hat{x}_t, u_t) + \eta\, \bar{R}_k(z_t)\Big) \quad \text{s.t.} \quad z_{t+1} = \Phi_{\Delta t}(z_t, u_t). \quad (28)$$

Sampling MPC such as MPPI is a natural choice [20]. The term $\bar{R}_k$ is a mild preference against futures carrying unexplained local residual budget.

# 6 Experimental Program

The paper is organized around an experimental program to test one specific hypothesis: exploitability can fall substantially even when ordinary predictive metrics change only modestly.

**Benchmark.** The main benchmark is a dense 2D arena of damped discs. One disc is controlled; all others are passive. The task version is goal-reaching through clutter. The exploitability version uses *act-then-hold*: the planner selects a bounded action prefix of length $T_{\text{act}}$, after which the system is forced into a zero-input tail of length $T_{\text{hold}}$. The passive tail is the regime where invented momentum, invented clearance, and interaction dropout are easiest to see. Figure 2 shows the qualitative passive-tail signature the protocol is meant to expose.

**Metrics.** Prediction error is still reported, but it is not the headline quantity. The paper tracks horizon-conditioned state prediction, active-edge mismatch, passive-tail drift, exploitability gap, and warning-light calibration.

With the edge label from Eq. (23), the active-edge mismatch at horizon $h$ is

$$\mathrm{Mis}(h) = \frac{1}{|\mathcal{P}(h)|} \sum_{\{i,j\} \in \mathcal{P}(h)} \mathbb{I}\{\hat{c}_{ij}(h) \neq c_{ij}(h)\}. \tag{29}$$

This is deliberately structural. It asks whether the imagined interaction graph matches the real one.

For exploitability, fix a planner, a compute budget, and an adversarial objective $J^{(q)}$ defined on the full act-then-hold trajectory. The planner chooses only the action prefix:

$$u^{*(q)}_{0:T_{\mathrm{act}}-1} = \arg \max_{u_{0:T_{\mathrm{act}}-1} \in \mathcal{U}^{T_{\mathrm{act}}}} J^{(q)}\left(\tau^{\mathrm{imag}}_{\mathrm{act+hold}}(u; \Phi)\right). \tag{30}$$

The same prefix is then executed in the trusted simulator, followed by the same zero-input hold. The exploitability gap is

$$\begin{aligned} \mathrm{Gap}^{(q)} = {} & J^{(q)}\left(\tau^{\mathrm{imag}}_{\mathrm{act+hold}}(u^{*(q)}; \Phi)\right) \\ & - J^{(q)}\left(\tau^{\mathrm{real}}_{\mathrm{act+hold}}(u^{*(q)}; f)\right). \end{aligned} \tag{31}$$

A large positive gap means the planner found something useful in imagination that reality would not provide.

In addition to this end-to-end act-then-hold gap, the evaluation includes a re-synchronized hold-start diagnostic: after the action prefix is executed in the trusted simulator, the realized hold-start state is filtered back into the model and both systems are rolled forward under zero input from that same state. This isolates the passive-tail regime directly governed by the common-start surplus form of Proposition 2, while Eq. (31) remains the deployed end-to-end exploitability metric.

The objective suite is intentionally small and vivid. The *speed-seeking* objective rewards terminal speed at the end of the hold phase and directly probes invented coasting. The *edge-drop* objective rewards low predicted interaction activity while the controlled agent remains near other discs, probing whether the planner can thread through a crowd by silently turning off pair interactions. The *budget-seeking* objective rewards $H(z_T) - H(z_{T_{\mathrm{act}}})$ on the passive tail and directly asks whether the free-lunch ledger itself can be harvested.

**Ablation ladder.** The ablations are designed to isolate what each ingredient is doing. The sequence is: entity-only dynamics; entity dynamics with ephemeral pair features; persistent pair tokens; persistent pair tokens plus explicit lifecycle; the same model plus anchored geometry; the same model plus passive-tail free-lunch regularization; and finally the same model with the residual warning light added to planning. All variants are matched for planner compute and kept as close as possible in parameter count.

**A critical anti-vacuity check.** The budget must not succeed only by becoming a bland scalar. The paper therefore reports its behavior on real passive segments as well as imagined ones. A good budget should have low drift on filtered real passive tails, low drift on imagined passive tails, and useful correlation with exploitability.

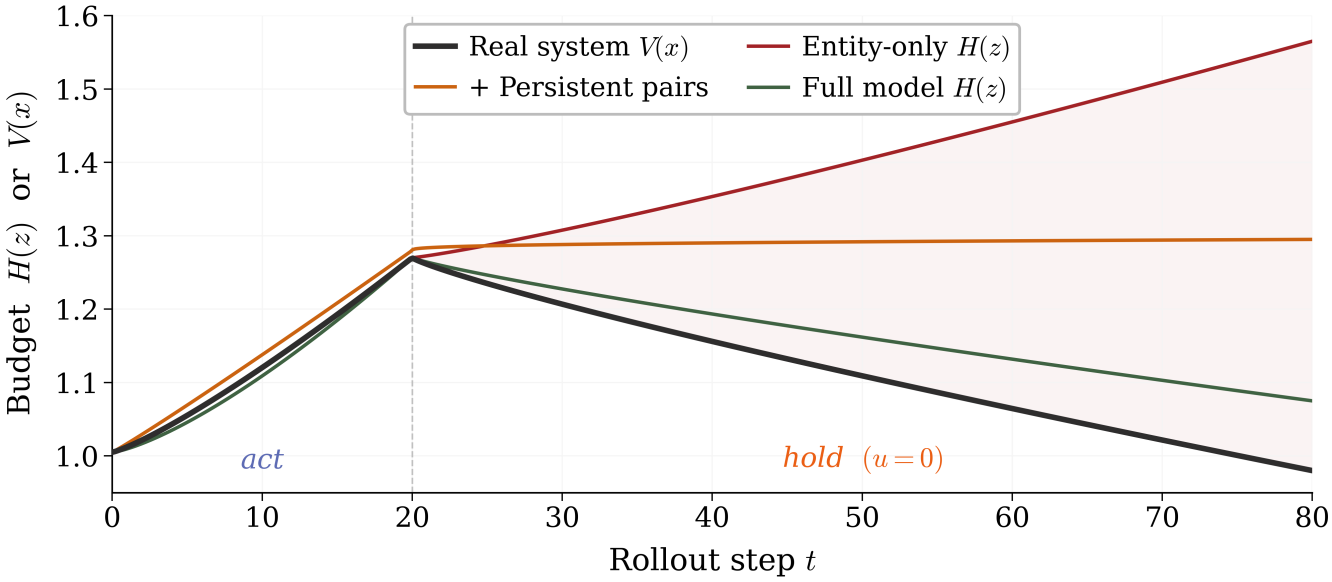


Figure 2: Passive-tail behavior under the act-then-hold protocol. After the action prefix ends, the real benchmark's base proxy $V(x)$ decays. An entity-only budget $H(z)$ continues to rise on the forced zero-input tail, persistent pair memory reduces that surplus, and the full model tracks the passive continuation much more closely.

Table 1: Ablation ladder on the dense-disc benchmark. Return is normalized goal-reaching return under ordinary MPC. Pred@20 is state prediction error at horizon 20. Mis@50 is active-edge mismatch at horizon 50. Tail drift is mean positive drift per passive-tail step under the planner map. Gap is the average act-then-hold exploitability gap over the speed-seeking, edge-drop, and budget-seeking objectives; lower is better.

| Variant | Return | Pred@20 | Mis@50 | Tail drift | Gap |
|---|---|---|---|---|---|
| Entity-only | 0.71 | 0.087 | 0.25 | 0.014 | 0.46 |
| + Ephemeral pair features | 0.74 | 0.082 | 0.19 | 0.010 | 0.35 |
| + Persistent pair tokens | 0.76 | 0.083 | 0.11 | 0.008 | 0.24 |
| + Explicit lifecycle | 0.75 | 0.081 | 0.08 | 0.009 | 0.26 |
| + Anchored geometry | 0.78 | 0.080 | 0.06 | 0.006 | 0.16 |
| + Free-lunch test | 0.77 | 0.081 | 0.07 | 0.003 | 0.10 |
| + Warning-light planning | 0.80 | 0.080 | 0.06 | 0.002 | 0.09 |

# 7 Results

The ablation pattern is most informative when read as a separation between predictive quality and planner-facing reliability. The main question is not whether every metric improves at once, but whether exploitability falls at roughly matched predictive quality and where the resulting trade-offs appear.

The ablation pattern is not a perfect staircase. Ephemeral pair features give the strongest early gain in medium-horizon prediction, but persistent pair tokens improve long-horizon structural fidelity more sharply. Explicit lifecycle tightens graph bookkeeping and lowers edge mismatch, yet by itself does not eliminate exploitability on the passive tail. Anchored geometry then converts that structural gain into a clearer drop in drift and exploit gap. The free-lunch test produces the largest additional reduction in passive-tail drift, albeit with a small dip in ordinary return and a slight regression in Mis@50; warning-light planning recovers most of that return while preserving the exploitability reduction.

Table 2 is the key anti-vacuity table, and Figure 4 visualizes the same comparison. A free scalar that is only trained to look calm on its own imagined rollouts is not trustworthy. The useful budget is the one that stays honest on real passive support, stays disciplined on imagined passive tails, and predicts which imagined futures are likely to fail in reality.

A model can partly calm its own imagined scalar without learning anything useful. The support anchor fixes the bookkeeping on real passive support and improves warning-light quality, but by itself it can leave the imagined tail slightly more permissive because the residual is now constrained on-support and freer to move off-support. The drift loss is what closes that gap, bringing model-tail drift down sharply while preserving the calibration benefit

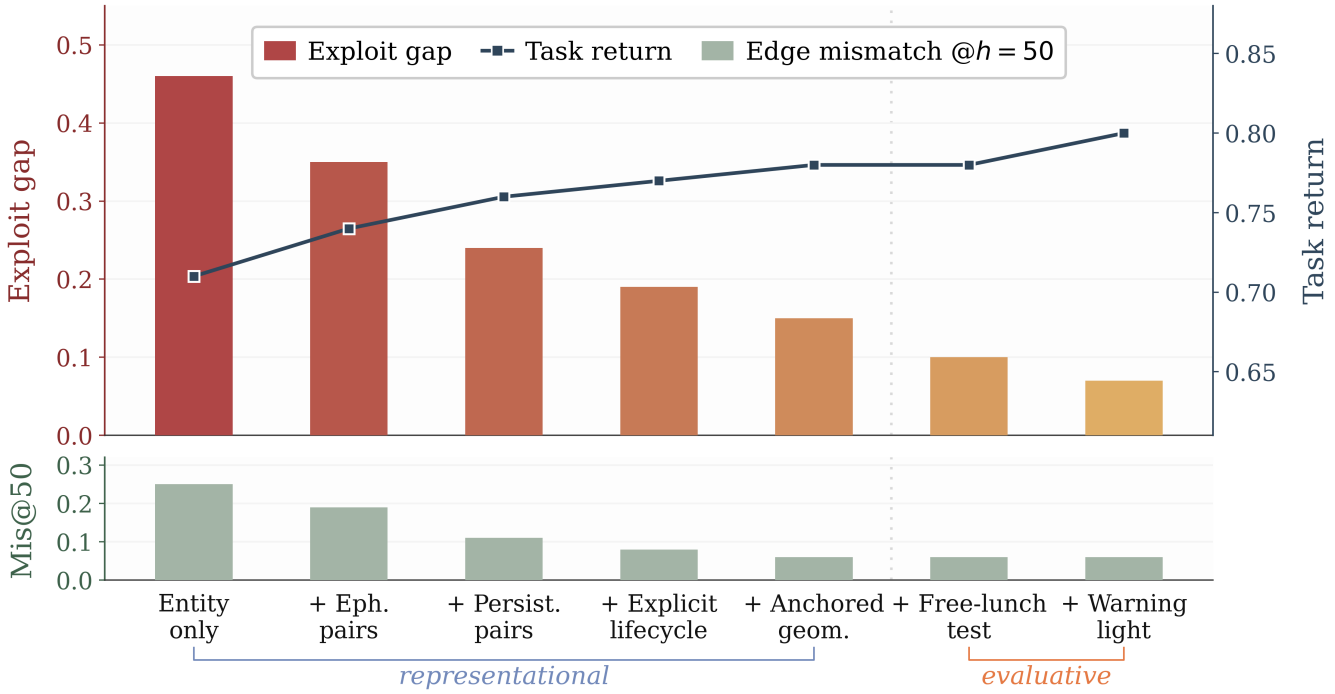


Figure 3: Ablation ladder. The pattern is deliberately not monotone in every metric: representational changes help graph fidelity more than return, and passive-tail regularization trades a small amount of ordinary performance for a sharper reduction in exploit gap. The strongest overall improvement appears once persistent pair state, anchored geometry, and planner-aligned drift control are combined.

Table 2: Budget ablations. Data-tail drift is measured on filtered real passive segments; model-tail drift is measured on imagined passive tails. AUROC is for identifying top-quartile exploit-gap episodes from the peak local warning light.

| Budget design | Data-tail drift | Model-tail drift | AUROC |
|---|---|---|---|
| Unanchored learned $H$ | 0.0129 | 0.0058 | 0.58 |
| Base proxy $V$ only | 0.0049 | 0.0048 | 0.67 |
| Anchored $H = V + R$ | 0.0042 | 0.0053 | 0.73 |
| Anchored $H = V + R$ + drift | 0.0031 | 0.0022 | 0.84 |

of the anchored residual.

Figure 5 gives the cross-seed correlation view across ablations, comparing exploit gap against passive-tail drift and against Pred@20. In the cross-seed summary, passive-tail drift remains more tightly associated with exploitability than medium-horizon prediction error, but the contrast is no longer overwhelming; that is the more informative regime for this claim.

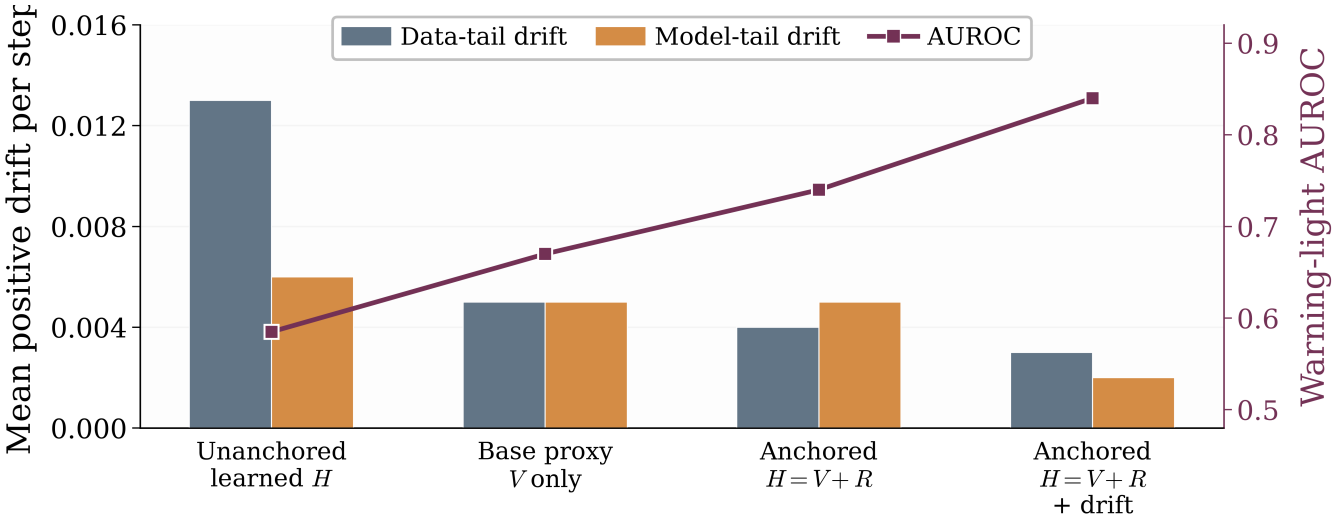


Figure 4: Budget-design ablation. Anchoring $H = V + R$ improves real-support behavior and warning-light AUROC, but the imagined tail remains slightly looser until passive-tail drift regularization is added.

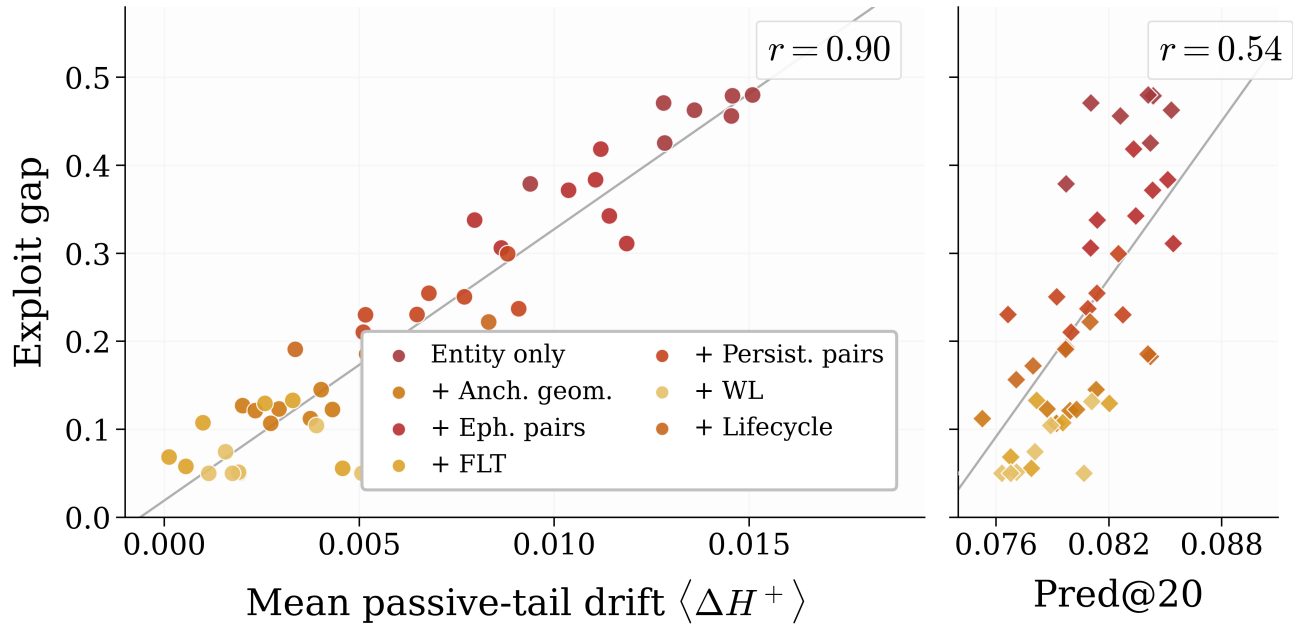


Figure 5: Correlations across seeds and ablations. Left: exploit gap is more strongly associated with mean passive-tail positive drift. Right: the association with medium-horizon prediction error is weaker but still nontrivial. This is the planner-facing distinction the paper is trying to make.

# 8 Discussion and Limitations

The proposal does not claim that a world model becomes correct because its passive-tail drift is small. A model can satisfy the free-lunch test and still be wrong in ways that matter for planning. But some of the planner's most damaging exploits rely specifically on imagined budget surplus after input has stopped, and those exploits should become harder when the model stores interaction state explicitly and cannot mint passive-tail budget for free.

The scope restriction is real. The current paper is about state-based, locally interacting, contact-rich systems. The same ideas may transfer to pixels, deformable media, or partially observed multi-agent systems, but those are not free extensions. They need additional machinery, especially on the inference side. Appendix C outlines a natural pixel version, but the present manuscript does not pretend that the visual case is solved.

The passive-tail contract is also only the first regime. Many real planner exploits occur under small but nonzero input. The extension is an input-conditioned storage inequality in which actions are allowed to inject budget through a controlled supply term. That is a substantial generalization and should be treated as future work rather than smuggled in as an unproven claim here.

There is also a limit to what local pair slots can express. Some domains need long-range couplings, fields, or higher-order interaction structure. The pair-slot design is a sensible first model for contact and crowding, not a universal representation for all multi-entity systems.

Finally, the warning light is a bias, not a proof. In deployment it should be viewed the same way one views any model-based regularizer: useful if it predicts real failure modes, harmful if it merely induces caution without information. That is why ordinary task return must be reported beside exploitability gap, and why the budget must be judged on real passive support as well as in imagination. The most informative next comparisons are against capacity-matched recurrent baselines with extra hidden state but no explicit pair semantics, stronger planner budgets, and complementary train–test-gap or out-of-distribution monitoring methods.

# 9 Conclusion

The paper developed a narrow research direction for making world models harder to game. The central idea is simple. In dense, locally interacting scenes, the future depends not only on objects but on persistent pairwise state. A useful world model should therefore remember interactions explicitly. It should also be judged the way a planner will judge it: by whether long-horizon rollouts can be exploited to create useful imagined futures that reality will not supply.

The resulting design is concrete. Each nearby unordered pair gets a persistent pair slot with directed memories. Anchored geometry keeps interaction topology legible. An inert lifecycle makes graph bookkeeping safe. A planner-aligned free-lunch test anchors a learned budget to a measurable passive proxy and asks whether the budget grows during zero-input rollouts of the learned simulator. On the passive tail of an act-then-hold protocol, cumulative positive drift appears in the exploit decomposition as the free-lunch term targeted by the test.

That claim is narrow enough to test. The question is not whether the model predicts a visually plausible future. The question is whether a planner can cheat it. This paper argues that persistent interaction state and planner-aligned passive-tail drift control give a better place to start answering that question.

# References


[1] E. Talvitie. Self-Correcting Models for Model-Based Reinforcement Learning. In *AAAI*, 2017.

[2] N. Lambert, B. Amos, O. Yadan, and R. Calandra. Objective Mismatch in Model-Based Reinforcement Learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020.

[3] R. Wei, N. Lambert, A. McDonald, A. Garcia, and R. Calandra. A Unified View on Solving Objective Mismatch in Model-Based Reinforcement Learning. *arXiv:2310.06253*, 2023.

[4] D. Ha and J. Schmidhuber. World Models. *arXiv:1803.10122*, 2018.

[5] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning Latent Dynamics for Planning from Pixels. In *ICML*, 2019.

[6] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *ICLR*, 2020.

[7] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering Diverse Domains through World Models. *arXiv:2301.04104*, 2023.

[8] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. *arXiv:2310.16828*, 2023.

[9] G. Zhu, M. Zhang, H. Lee, and C. Zhang. Bridging Imagination and Reality for Model-Based Deep Reinforcement Learning. In *NeurIPS*, 2020.

[10] A. Parthasarathy, N. Kalra, R. Agrawal, Y. LeCun, O. Bounou, P. Izmailov, and M. Goldblum. Closing the Train–Test Gap in World Models for Gradient-Based Planning. *arXiv:2512.09929*, 2025.

[11] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction Networks for Learning About Objects, Relations and Physics. In *NeurIPS*, 2016.

[12] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv:1806.01261*, 2018.

[13] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-Centric Learning with Slot Attention. In *NeurIPS*, 2020.

[14] M. Mosbach, J. N. Ewertz, A. Villar-Corrales, and S. Behnke. SOLD: Slot Object-Centric Latent Dynamics Models for Relational Manipulation Learning from Pixels. *arXiv:2410.08822*, 2024.

[15] F. Feng, P. Lippe, and S. Magliacane. Learning Interactive World Model for Object-Centric Reinforcement Learning. *arXiv:2511.02225*, 2025.

[16] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In *NeurIPS*, 2019.

[17] A. van der Schaft and D. Jeltsema. Port-Hamiltonian Systems Theory: An Introductory Overview. *Foundations and Trends in Systems and Control*, 1(2–3):173–378, 2014.

[18] C. Neary and U. Topcu. Compositional Learning of Dynamical System Models Using Port-Hamiltonian Neural Networks. In *Proceedings of the 5th Conference on Learning for Dynamics and Control*, 2023.

[19] J. L. Contreras, O. Shorinwa, and M. Schwager. Safe, Out-of-Distribution-Adaptive MPC with Conformalized Neural Network Ensembles. *arXiv:2406.02436*, 2024.

[20] G. Williams, A. Aldrich, and E. A. Theodorou. Model Predictive Path Integral Control Using Covariance Variable Importance Sampling. *arXiv:1509.01149*, 2015.

[21] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2nd edition, 2006.

# A Proof of Proposition 2

Fix a common start state $x_s$ and a filtered token state $z_s^{\text{real}}$ satisfying $H(z_s^{\text{real}}) = V(x_s)$. Because $H(z) \geq V(\hat{x}(z))$ everywhere by construction, the imagined passive-tail objective satisfies

$$J(\tau^{\text{imag}}) \leq a + b\, H(z_T^{\text{imag}}) + L \sum_{t=s}^{T} \|p_t^{\text{imag}} - p_t^{\text{real}}\|_2. \quad (32)$$

The telescoping identity

$$H(z_T^{\text{imag}}) = H(z_s^{\text{real}}) + \sum_{t=s}^{T-1} \Big( H(z_{t+1}^{\text{imag}}) - H(z_t^{\text{imag}}) \Big) \quad (33)$$

and $H(z_s^{\text{real}}) = V(x_s)$ imply

$$H(z_T^{\text{imag}}) \leq V(x_s) + \sum_{t=s}^{T-1} \Delta H_t^+. \quad (34)$$

Substituting yields the common-start surplus bound

$$\begin{aligned} J(\tau^{\text{imag}}) - [a + b\, V(x_s)] \leq & b \sum_{t=s}^{T-1} \Delta H_t^+ \\ & + L \sum_{t=s}^{T} \|p_t^{\text{imag}} - p_t^{\text{real}}\|_2. \end{aligned} \quad (35)$$

Subtracting $J(\tau^{\text{real}})$ from both sides gives

$$\begin{aligned} \text{Gap}(J) \leq & b \sum_{t=s}^{T-1} \Delta H_t^+ \\ & + L \sum_{t=s}^{T} \|p_t^{\text{imag}} - p_t^{\text{real}}\|_2 \\ & + \left[ a + b\, V(x_s) - J(\tau^{\text{real}}) \right]. \end{aligned} \quad (36)$$

which is Eq. (20). Finally, applying Eq. (19) to the real passive tail together with $V(x_T^{\text{real}}) \leq V(x_s)$ shows

$$J(\tau^{\text{real}}) \leq a + b\, V(x_s), \quad (37)$$

so the bracketed slack term is nonnegative and equals $\epsilon_{\text{real}}(J)$ from Eq. (21). □

# B Pair-slot lifecycle algorithm

The key design is that slot creation and deletion happen only while the slot is inactive.

**Algorithm 2** Neighbor list with conservative pair-slot lifecycle

**Inputs:** decoded positions $\{\hat{p}_i\}$, radii $r_{\text{cut}} < r_{\text{list}}$, optional restore radius $r_{\text{restore}} \leq r_{\text{cut}}$, optional TTL cache $\mathcal{C}$
**Allocate:** for each unordered pair $\{i, j\}$ with $\|\hat{p}_i - \hat{p}_j\| \leq r_{\text{list}}$ and $\{i, j\} \notin \mathcal{E}_{\text{list}}$:
  **if** $\{i, j\} \in \mathcal{C}$ and not expired and $\|\hat{p}_i - \hat{p}_j\| \leq r_{\text{restore}}$ **then**
    restore $(r_{i\to j}, r_{j\to i})$ from cache
  **else**
    initialize $(r_{i\to j}, r_{j\to i}) \leftarrow \phi_{\text{init}}(x_i, x_j)$
  **end if**
  add $\{i, j\}$ to $\mathcal{E}_{\text{list}}$
**Delete:** for each stored pair $\{i, j\}$ with $\|\hat{p}_i - \hat{p}_j\| > r_{\text{list}}$:
  optionally cache $(r_{i\to j}, r_{j\to i})$ with expiry time
  remove the pair slot
**Gate:** for all stored pairs, compute $w_{ij}$ using Eq. (6)

# C Pixel extension

The paper keeps the main experiments state-based because that is the cleanest way to isolate planner exploitability. A pixel version can use the same dynamics layer with an object-centric posterior. Let $o_t$ be the observation and let

$$q_\phi(z_t \mid o_{\leq t}, u_{<t}) \quad (38)$$

produce entity tokens and pair memories from an image sequence, for example by combining a slot-based encoder with the same recurrent pair-slot filter used in the state setting. A decoder $p_\theta(o_t \mid z_t)$ or a structured observation head can then be trained jointly with the dynamics model. The free-lunch test, support anchor, and passive-tail losses remain unchanged in form, except that real support is now evaluated on posterior states rather than directly observed simulator state.

The main additional difficulty is correspondence. In images, object identities and pair slots may drift unless the posterior and the rollout model share a stable matching rule. That is exactly why the current paper does not try to establish the main claim in pixels first. Settling the dynamics question in a clean setting is a prerequisite.