MemJEPA: Learning What a Latent World-Model Planner Must Remember

Steven Yang
Independent Researcher
steveny5@uci.edu

Abstract: Latent world models such as DINO-WM and LeWorldModel (LeWM) plan by imagining the outcomes of candidate actions from their last few camera frames. However, they fail after what the robot controls has been out of view for longer than that. We introduce MemJEPA to address this: it carries a recurrent state through the imagination of the cross-entropy method (CEM) planner and trains it on simulated data. MemJEPA is trained on 1) erasure, which removes the controlled entity from the input frames but keeps it in the prediction targets, and 2) counterfactual training, which replays recorded moments with altered actions to teach the model about actions it never took. On reaching, pushing and navigation tasks with LeWM's frozen encoder and the entity hidden, MemJEPA trained with erasure succeeds about 78% of the time, against 13% for LeWM's predictor fine-tuned on the same erased clips without memory, and loses nothing when the entity is visible. Erasure training allows the model to remember even during occlusion: trained on ordinary clips, the same recurrent state model does not perform well with the entity hidden.

Keywords: JEPA, World models, Memory, Occlusion, Partial observability

1 Introduction

When an object is occluded from a robot and leaves its field of view, it can affect the quality of the robot's ability to accurately plan and complete the task. There are several memory-based approaches to address this: Bayes filters [1], recurrent policies [2, 3], object tracking through occlusion [4, 5] and recurrent world models for planning [6]. Memory benchmarks for manipulation and navigation also test abilities such as recalling objects out of view [7, 8]. However, latent world models such as DINO-WM and LeWorldModel (LeWM) [9, 10] keep only a short window of recent frames. They imagine the outcome of candidate actions in an image encoder's latent space and pick those that land closest to the goal. Every planner in this lineage [9, 10, 11, 12, 13, 14] has such a short window and is evaluated with the controlled entity in view. Once the entity has been out of view for longer than that, the planner no longer knows where it is, and success falls sharply (Figure 1, bar 1).

Prior world models add memory, either a recurrent state [15] or a belief fed into the predictor [16]. Trained on ordinary clips, this memory does not help (Figure 1, bars 2 and 3). The missing piece is the training signal. In ordinary clips the entity is always visible, so the model can read it from the current frame and its state has no reason to remember it. We erase the entity from the input frames of training clips but keep the clean next latents as targets. The targets still depend on the entity, so the model must carry it from earlier frames.

MemJEPA builds on LeWM [10] (Figure 2), which plans with the cross-entropy method (CEM) and whose predictor sees at most the last three frames (15 environment steps). Appendix B gives the equations and hyperparameters.

CoRL 2026 Workshop on Memory for Robot Foundation Models, Austin, Texas, USA.

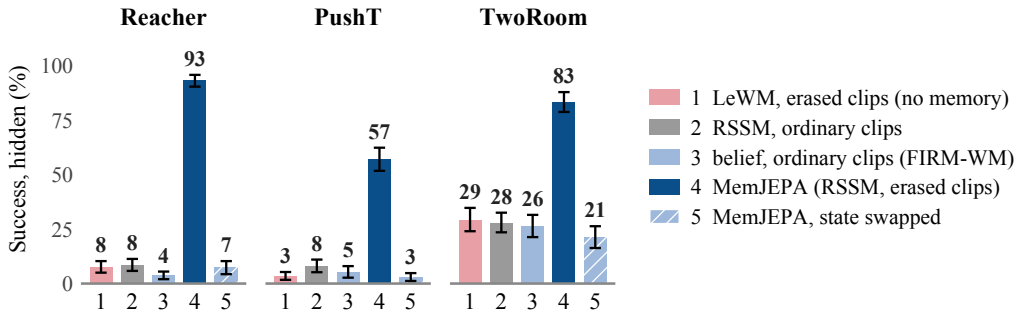


Figure 1: **What restores planning with the entity hidden.** Held-out sets, 95% intervals. Bar 1 is LeWM's predictor fine-tuned on the same erased clips, with no memory. Only MemJEPA (4) clearly beats it, and swapping its state across episodes (5) removes the gain.

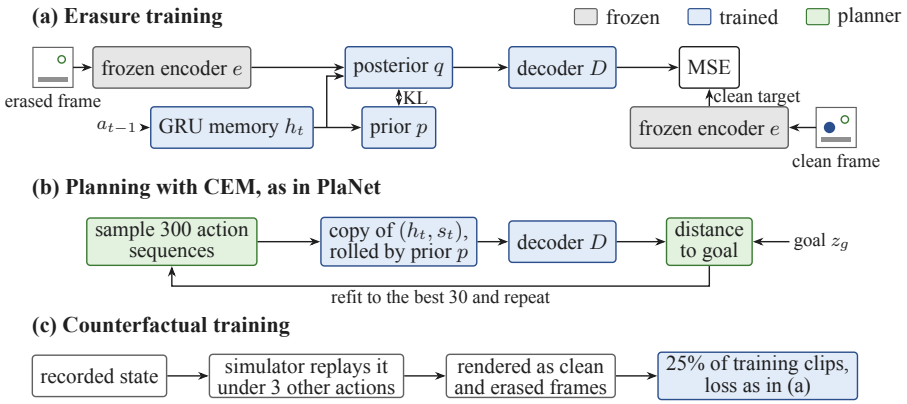


Figure 2: **MemJEPA.** (a) Erasure training: the RSSM sees erased frames but must predict the clean next latent, so it has to remember the entity. (b) Planning, as in PlaNet: each CEM candidate rolls its own copy of the state forward with the prior. (c) Counterfactual training: simulator replays under other actions become extra training clips.

Our contributions are as follows:

(1) A recurrent state for a JEPA planner. MemJEPA replaces LeWM's three-frame predictor with PlaNet's recurrent state-space model (RSSM) [15]. The RSSM updates its state from every frame, erased or not, and a decoder predicts the next latent from that state. Because the state persists across frames, it can still carry the entity after the entity leaves view. As in PlaNet, CEM gives each candidate action sequence its own copy of the current state and rolls it forward along that candidate's actions; the copies are discarded after scoring, and the real state carries over to the next replan.

(2) Erasure training. On half of the training clips, the simulator stops drawing the entity after the first two frames, as at test time (Appendix A). The state sees only these erased frames, but the decoder must predict the clean latent, which still shows the entity. To lower the loss, the state has to remember where the entity was and how it moved. The same model trained on ordinary clips is the standard RSSM; it can read the entity from every frame, so its state never has to hold it. A second design, a GRU belief added to LeWM's predictor, is trained the same way (B in tables; Appendix B).

(3) Counterfactual training. Recorded clips contain only the actions that were taken, but the planner scores many actions that were not. Following FIRM-WM's intervention branches [16], the simulator replays recorded moments under other actions and erases the entity in the same way, so the state also learns what those actions do.

Occlusion occurs when a robot loses sight of objects it is trying to detect. This includes its own parts, though with proprioception this may not be a critical issue. It is an issue in multi-robot workflows, where a robot cannot sense another robot's joints, and in tasks that cannot rely on proprioception. We address this problem of occlusion in latent-space planners and study how to maintain or improve their success on simulated tasks when the controlled entity is missing from the camera frames. In three tasks the simulator renders every frame after the first two without the entity (the arm in Reacher, the pusher in PushT, the agent in TwoRoom; Appendix A), and the planner gets no proprioception. We compare memory with an open-loop rollout and a proprioceptive oracle and, without retraining, also test a moving patch, a fixed occluded region and imperfect erasure masks (Appendix K).

Prior recurrent world models and beliefs [17, 18, 16, 19, 5] train against the observed frames, on ordinary clips, or without planning (Appendix N).

2 Experiments

Protocol. We use three public LeWM tasks: DeepMind Control Reacher [20] (arm), PushT [21] (pusher) and TwoRoom [22] (agent). A case succeeds if the environment's own test passes before a step budget runs out. Each memory model is compared with a *no-memory model*: LeWM's predictor fine-tuned with the same recipe, erased clips and budget, but no carried state. Held-out sets were opened only after case lists and success criteria were fixed (Appendices B, H); intervals are 95% case-clustered bootstraps, and TwoRoom's held-out results are second evaluations.

Two main results follow.

1. **Erasure training lets a latent-space planner use its memory.** Memory alone (the standard RSSM, trained on ordinary clips) and erasure alone (the no-memory model) both fail with the entity hidden. Together they raise hidden success from 13% to about 78%, with no loss in view (Fig. 1, Table 1). The GRU belief works nearly as well.
2. **When counterfactual training helps.** Where the hidden entity moves something visible (PushT) and MemJEPA still trails its in-view success, training on simulator replays under other actions adds 10.9 points, on every seed. On Reacher and TwoRoom, where MemJEPA already nearly matches its in-view success, it changes nothing (Appendix G).

Memory helps only when training demands it. On ordinary clips, reading the entity from the current frame is always cheaper, so the same RSSM never learns to hold it and fails as badly as having no memory; erasure removes that shortcut. A longer context window is no substitute: it works only while it still reaches a frame showing the entity (Appendix D).

The clean target is the essential half. Erasing the inputs is not enough. If the target is the erased frame as well, the state has little reason to carry the entity, and both designs fall to near the no-memory level (Appendix E). The target must contain what the input lacks.

The planner relies on the remembered history. Swapping the state between episodes with the same visible frames but different hidden histories sends MemJEPA back to the no-memory level (Figure 1, bar 5); a linear probe reads the hidden fingertip from the state, not the frame (Appendix E).

Table 1: **Planning success (%) with the entity hidden and in view.** Held-out sets, 200 cases $\times$ 3 seeds; intervals in Appendix C. The standard RSSM is trained on ordinary clips.

	Reacher (arm)		PushT (pusher)		TwoRoom (agent)	
	hidden	in view	hidden	in view	hidden	in view
No-memory model	7.5	93.8	3.3	51.7	29.3	84.3
10-frame context window	74.0	94.7	32.5	74.7	78.0	82.7
RSSM, standard training	8.5	94.2	7.8	74.3	27.8	81.0
GRU belief (B)	93.0	94.7	52.0	78.8	79.0	83.8
MemJEPA	93.2	93.0	57.0	72.0	83.3	87.0

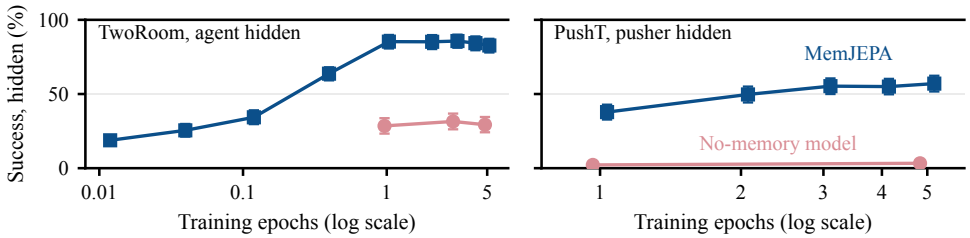


Figure 3: **Training compute.** Success with the entity hidden by training epoch (held-out sets, 3 seeds, 95% intervals; an epoch is 2.9 GPU-minutes on TwoRoom, 9.9 on PushT). The no-memory model never catches up.

When memory is worth it. If the hidden entity's motion follows from the commanded actions, the no-memory model can roll its predictor forward on those actions, and on Reacher and TwoRoom this open-loop rollout comes close to MemJEPA. A real robot's proprioception would likewise reveal its own arm: given that state, the no-memory model solves Reacher as well as MemJEPA does from pixels (Appendix K). Memory earns its place on PushT, where the hidden pusher moves a visible block. The effect of contact cannot be replayed from the actions alone, and MemJEPA beats the open-loop rollout by about 30 points. This is the case that matters for robots, which lose track of the objects they act on rather than of their own joints.

Counterfactual training teaches what recordings cannot. The planner scores action sequences the recordings never took, so a state trained only on recorded trajectories is least accurate exactly where the planner looks. Counterfactual training cuts the state's error most off the recorded trajectory (Appendix G).

Cost and return on compute. Memory is cheap: MemJEPA replans about twice as fast as the no-memory model, and more planning candidates cannot replace it (Appendix E). It also trains fast (Figure 3; Appendix E). Most of the return comes in the first epochs. On TwoRoom, one epoch (3 GPU-minutes) takes hidden success to 85%, and later epochs add nothing. PushT is near its final success by epoch 3. The no-memory model, fine-tuned on the same erased clips, stays near 29% on TwoRoom through five epochs. The gap comes from the carried state, which costs a few GPU-minutes of training. Limitations are in Appendix M.

3 Conclusion and Future Directions

With erasure training and clean targets, a recurrent state (an RSSM or a GRU belief) restores planning with the entity hidden. Memory matters most when the hidden entity moves something visible. On PushT, where the hidden pusher moves the block, MemJEPA beats an open-loop rollout by about 30 points, and counterfactual training adds about 11 more. All results are simulated (Appendix M). Future directions involve extending MemJEPA to realistic occlusion conditions using real robotics parts, particularly for long horizon tasks (Appendix J) and multi-robot workflows.

References

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[2] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015. arXiv:1507.06527.

[3] E. Cherepanov, N. Kachaev, D. Zelezetsky, A. Bulatov, A. Pshenitsyn, Y. Kuratov, A. Skrynnik, A. I. Panov, and A. K. Kovalev. μVLA: On recurrent memory for partially observable manipulation in VLA models, 2026. arXiv:2606.12497.

[4] Y. Huang, J. Yuan, C. Kim, P. Pradhan, B. Chen, F. Li, and T. Hermans. Out of sight, still in mind: Reasoning and planning about unobserved objects with video tracking enabled memory models. In *ICRA*, 2024. arXiv:2309.15278.

[5] M. Traub, F. Becker, S. Otte, and M. V. Butz. Learning object permanence from videos via latent imaginations. In *ICANN*, 2024. arXiv:2310.10372.

[6] Y. Du, P. Song, Y. Hu, and R. Detry. ELVIS: Ensemble-calibrated latent imagination for long-horizon visual MPC. In *RSS*, 2026. arXiv:2605.04709.

[7] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *ICLR*, 2026. arXiv:2502.10550.

[8] J. Pasukonis, T. Lillicrap, and D. Hafner. Evaluating long-term memory in 3D mazes. In *ICLR*, 2023. arXiv:2210.13383.

[9] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025. arXiv:2411.04983.

[10] L. Maes, Q. Le Lidec, D. Scieur, Y. LeCun, and R. Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. arXiv:2603.19312.

[11] Y. Wang, O. Bounou, Y. LeCun, and M. Ren. AdaJEPA: An adaptive latent world model, 2026. arXiv:2606.32026.

[12] K. Zhao, D. Nie, Y. Lin, Z. Luo, Y. Gu, D.-P. Fan, and D. Zeng. Sub-JEPA: Subspace Gaussian regularization for stable end-to-end world models, 2026. arXiv:2605.09241.

[13] P. Pham and A. Bera. Latent energy action planning with world models, 2026. arXiv:2609.03294.

[14] M. Karmann, S. Li, C. Internò, B. Andreis, D. Klindt, R. Balestriero, J. Gu, P. Torr, Q. Zhang, P.-T. Jiang, H. Zhang, B. Li, and O. Urfalioglu. MotionJEPA: Preventing temporal feature collapse by capturing visual changes in latent space, 2026. arXiv:2609.23881.

[15] 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. arXiv:1811.04551.

[16] Y. Wu, Y. Zhang, A. Li, M. Wang, H. Wu, and A. Zhang. FIRM-WM: State-factorized factual–interventional recurrent modeling for reward-free visual planning, 2026. arXiv:2609.22816.

[17] C. Gumbsch, M. V. Butz, and G. Martius. Sparsely changing latent states for prediction and planning in partially observable domains. In *NeurIPS*, 2021. arXiv:2110.15949.

[18] P. Becker and G. Neumann. On uncertainty in deep state space models for model-based reinforcement learning. *Transactions on Machine Learning Research*, 2022. arXiv:2210.09256.

[19] T. Y. Zhou and D. Zhang. Learning counterfactual world models for embodied reasoning under partial observability, 2026. arXiv:2609.05834.

[20] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind Control Suite, 2018. arXiv:1801.00690.

[21] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023. arXiv:2303.04137.

[22] V. Sobal, W. Zhang, K. Cho, R. Balestriero, T. G. J. Rudner, and Y. LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *NeurIPS*, 2025. arXiv:2502.14819.

[23] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering Atari with discrete world models. In *ICLR*, 2021. arXiv:2010.02193.

[24] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. arXiv:2301.04104.

[25] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[26] M. Huh, B. Cheung, T. Wang, and P. Isola. Position: The Platonic representation hypothesis. In *ICML*, 2024.

[27] M. R. Samsami, A. Zholus, J. Rajendran, and S. Chandar. Mastering memory tasks with world models. In *ICLR*, 2024. arXiv:2403.04253.

[28] L. Maes, Q. Le Lidec, D. Haramati, N. Massaudi, D. Scieur, Y. LeCun, and R. Balestriero. stable-worldmodel-v1: Reproducible world modeling research and evaluation, 2026. arXiv:2602.08968.

[29] M. Pleines, M. Pallasch, F. Zimmer, and M. Preuss. Memory Gym: Towards endless tasks to benchmark memory capabilities of agents. *Journal of Machine Learning Research*, 26(6): 1–40, 2025. arXiv:2309.17207.

[30] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *CoRL*, 2021. arXiv:2108.03298.

[31] R. Balestriero and Y. LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics, 2025. arXiv:2511.08544.

[32] F. Deng, J. Park, and S. Ahn. Facing off world model backbones: RNNs, transformers, and S4. In *NeurIPS*, 2023. arXiv:2307.02064.

[33] S. M. M. Islam and M. Huber. Dynamics-aligned representation learning under induced partial observability, 2026. arXiv:2610.25511.

[34] C. D. Freeman, L. Metz, and D. Ha. Learning to predict without looking ahead: World models without forward prediction. In *NeurIPS*, 2019. arXiv:1910.13038.

[35] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. In *CoRL*, 2022. arXiv:2206.14244.

[36] H. Nam, Q. Le Lidec, L. Maes, Y. LeCun, and R. Balestriero. Causal-JEPA: Learning world models through object-level latent masking. In *ICML*, 2026. arXiv:2602.11389.

Appendix

Table cells are recomputed from per-episode records by a tally script written separately from the experiment code.

Notation. MemJEPA (S in tables) is the erasure-trained RSSM of Section 1. B is a second state design, an erasure-trained GRU belief read at LeWM's predictor input. B_0 is B without its imagination-consistency term, which was added after the success criteria were fixed (Appendix B). Both designs use erasure training and advance their state inside every CEM candidate. "Twin" (in tables and text) is the no-memory model of Section 2, and "+ branches" marks counterfactual training.

A Erased frames and the fill rule

Erasure is a render-only operation (Figure 4). On Reacher the arm's geometry is made invisible for one render. On PushT the pusher's shapes are skipped when drawing. On TwoRoom the agent's dot is not drawn. The simulator is not stepped, and its state is unchanged. Nothing is painted over the frame, so the planner has no fill color, silhouette or mask channel to use. On Reacher and TwoRoom the erased frame is one image for every entity pose. On PushT it varies only with the visible block; the green T is a fixed marker, not the case's goal. The hidden pusher still moves the block: that is the task's dynamics, not a mask leak.

Check. We moved the entity to four other poses per case on the held-out sets and rendered again. All 5,600 erased frames were bit-identical to the original (Reacher 2,400, PushT 800 at the start state, TwoRoom 2,400), while the visible frames differed on every case. This is an idealized limit: no occluder is present, the simulator supplies the render, and imperfect erasure is tested only at test time (Appendix M).

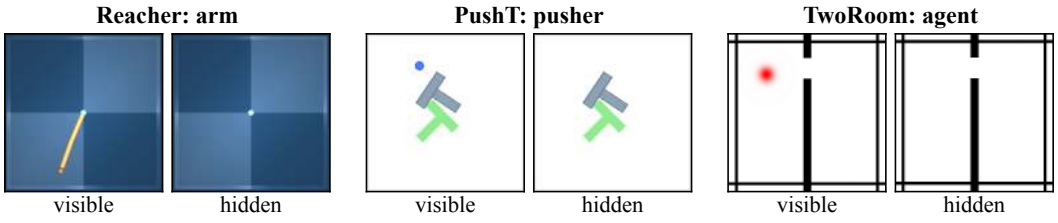


Figure 4: The three tasks with the controlled entity visible and hidden: the Reacher arm, the PushT pusher (blue dot) and the TwoRoom agent (red dot). Only the entity is erased. The cyan dot in the hidden Reacher frame is the arm's fixed base; the hidden PushT frame keeps the gray block and the fixed green T marker.

B Protocol details

Tasks and training. We release the protocol and the case lists. Success is each environment's own test. On PushT the pusher and block positions must both lie within 20 px of the goal state, and the block angle within $\pi/9$ rad. The goal is d steps later in an expert demonstration, and the budget is $2d$. Erased frames come from the simulator's render (Appendix A). Each training clip stays fully visible with probability $p_{\text{full}} = 0.5$ and is otherwise erased from frame $k_{\text{vis}} = 2$, for every variant and its twin. The no-erasure controls set $p_{\text{full}} = 1$. At test the planner receives only the erased frames and the goal image. Training uses one RTX 4090 per seed and takes 74 to 78 minutes on Reacher for MemJEPA or the GRU belief, or 15 to 45 minutes when only the GRU belief trains on the frozen predictor. Table 2 gives the hyperparameters.

Model. LeWM's encoder is a ViT-Tiny, frozen here; its predictor takes the last three latents here (LeWM uses one on TwoRoom) and action blocks of five primitive steps, so its window spans at most 15 environment steps, and CEM minimizes $\|\hat{z}_{t+H}(a) - z_g\|_2^2$ between the imagined horizon latent

and the goal latent. MemJEPA's RSSM has a deterministic state $h_t = \mathrm{GRU}(h_{t-1}, [s_{t-1}; a_{t-1}])$, a stochastic state s_t with prior $p(s_t \mid h_t)$ and posterior $q(s_t \mid h_t, \tilde{z}_t)$, and a decoder D. With erased inputs $\tilde{z}_t = e(\tilde{o}_t)$ and clean targets $e(o_t)$ it minimizes $\mathcal{L} = \sum_t \|D(h_t, s_t) - e(o_t)\|_2^2 + \mathrm{KL}(q \,\|\, p)$, with KL balancing 0.8 [23] and 1 nat of free bits [24]. In imagination each candidate's copy $(h_t^{(k)}, s_t^{(k)}) = (h_t, s_t)$ is rolled forward with the prior and decoded by D. The GRU belief instead adds a GRU state, updated with every observed latent, to each window token of LeWM's predictor, and is trained against the clean next latent in the same way.

Evaluation sets. A pre-registered prediction or criterion was written down before its evaluation ran. Design choices used development sets. The held-out sets (200 cases $\times$ 3 seeds; Table 1) were opened only after their case lists and the success criteria were pre-registered. The TwoRoom held-out set had first been evaluated with an earlier variant trained on its own predictions, so its contrasts are second evaluations. The never-opened sets were each evaluated once, in one batch (Appendix H): new start windows on held-out Reacher episodes, a new TwoRoom draw and a first never-opened PushT set of 200 cases. The fresh PushT set (300 cases in 150 pairs, excluded from every training) is a second never-opened PushT set, used for Table 5 and Appendices G and K. The PushT span set (Appendix J) reuses episodes of earlier sets with new windows, prefix paths and goals. The Reacher confirmation set (200 cases per seed) had already been opened by an evaluation with the state frozen; Appendix E uses it only to compare ways of advancing the state.

Statistics. Intervals come from a paired bootstrap (10,000 resamples, fixed seed) over cases or episodes; on Reacher the two cases cut from one episode (a case pair) are resampled together. Tests are exact two-sided sign tests on discordant pairs. The bootstrap is conditional on the trained models; the seed-level intervals of Appendix C are conditional on the case list. Three seeds cannot reach $p < 0.05$ on a rank test, so we report how many seeds meet each criterion.

Exclusions and post-hoc tests. The twin's failure on Reacher is not a tuning artifact: other horizons (3 and 10 blocks) and 20 CEM iterations reach at most 11.0%. Before any memory variant ran, two settings were excluded because hiding the entity cost the memoryless planner less than 20 points: TwoRoom with the door hidden (14 and 25% in view against 3 and 7% hidden, two door settings) and Cube, where the recipe-matched memoryless planner keeps its success with the arm erased (40.7% in view, 38.3% hidden). The 10-frame window, the state swap, the freeze and the two comparisons of Appendix E (where the predictor conditions on the belief, how the state is advanced) were evaluated after the primary variants, on the same or earlier-opened sets, so they are post hoc.

B_0 and B. B_0 is the base GRU belief: B without its added imagination-consistency term. The success criteria were fixed on B_0. The term was added afterwards: on half the clips, from a random position on, the belief is fed the model's own multi-step imagined latents. On the same evaluations as Table 1, B_0 scores 91.0, 47.7 and 74.5% hidden (first recorded at 91.2, 47.7 and 75.5%). B's extra 2.0, 4.3 and 4.5 points therefore come from a term chosen after these sets were opened. At seed level only TwoRoom's gain excludes zero. On the never-opened sets (Appendix H) the gain does not replicate ($+0.3$, $+2.8$ and $+2.3$, every interval including zero), so we make no claim for the term.

C Contrasts with confidence intervals

Table 3 gives the primary contrasts against the no-memory model with their intervals.

Seed-level intervals. Table 4 gives each contrast per seed (200 cases each, one shared case list) with three intervals. The case-clustered interval resamples cases (or case pairs) for the seed-averaged paired difference, conditional on the trained models. The seed-level interval is a t interval on the three per-seed differences (2 degrees of freedom), conditional on the case list. The two-stage interval resamples seeds, then cases or case pairs (10,000 draws; three seeds allow ten distinct seed draws).

Caveats. The table re-analyzes the per-episode records; the statistic was fixed in advance, the criteria are unchanged, and the outcome follows from the counts. The criteria were fixed on B_0, before B's added term (Appendix B). S trains from scratch, so its seed pairing is nominal. An unpaired Welch

Table 2: Training hyperparameters on the LeWM tasks (encoder frozen throughout). k_{vis}: frames visible at the start of an erased clip; p_{full}: probability that a training clip stays fully visible.

Hyperparameter	No-memory model	GRU belief (B)	MemJEPA (S)
Optimizer	AdamW, weight decay 10^{-3}		
Learning rate	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$3 \cdot 10^{-4}$
Learning-rate schedule	linear warm-up, then cosine annealing stepped per epoch		
Batch size (clips)	128		
Training steps (5 epochs)	Reacher 48,575; PushT 41,705; TwoRoom 13,045		
Precision, gradient clip	bf16, gradient-norm clip 1.0		
Clip length (frames)	11		
Context window (latents)	3	3	n/a
Frame skip (action block)	5 primitive steps per latent		
k_{vis}, p_{full}	2, 0.5		
Loss	next-latent MSE	next-latent MSE	MSE + KL[†]
Imagination-consistency term[‡]	no	p=0.5, $j \sim \mathcal{U}\{3, \ldots, 9\}$	no
State carried across clips	n/a	PushT, TwoRoom	no
Seeds	0, 1, 2		
CEM samples, elites	300 samples, top 30		
CEM iterations	Reacher 10; PushT and TwoRoom 30		
Planning horizon	5 blocks = 25 steps, whole plan executed before replanning		

[†]Decoder MSE to the clean latent plus KL (balance 0.8, free bits 1 nat per visible step). [‡]Each clip is selected with probability p; from input position j on, the inputs are the model's own multi-step imagined latents (detached).

interval in place of the seed-level one changes one result: in-view S − B on PushT then excludes zero ($[-13.5, -0.1]$).

In-view success and S against B. On PushT the erasure recipe lowers the twin's in-view success (51.7, against 74.3 for the standard-training RSSM, trained on clean clips), which partly explains the GRU belief's in-view lead there. That RSSM, strong in view, still falls to 7.8% with the pusher hidden. With the entity hidden, MemJEPA leads the GRU belief on TwoRoom ($+4.3$ $[+0.7, +8.2]$), is not separable from it on PushT and is level on Reacher. In view, the GRU belief is ahead by 6.8 on PushT (all three seeds; seed-level interval $[-7.1, +20.8]$), behind by 3.2 on TwoRoom ($[-3.9, -2.4]$), and ahead by 1.7 on Reacher, within seed variation ($[-2.1, +5.5]$).

D Hidden-span and context-window results

Hidden span. Figure 5 reports Reacher success by the number of strided frames hidden before planning, L. At 2 hidden frames the 10-frame window trails the GRU belief by 19.0 points. At 20, past the training clip, the 40-frame window comes within 6.3 points of the GRU belief (51.8 against 58.2%) but not of MemJEPA (84.5%), and the 10-frame window, which by then holds no frame showing the arm, and the cache have fallen to 7.7 and 5.7% (the bounds of Appendix L). At 33 (165 environment steps, the longest Reacher span tested) the 40-frame window leads the GRU belief B_0, 45.5 against 24.8%.

Context windows and caches. A cache helps only while the hidden entity stays still: in a recall test where the hidden object does not move, it beats B_0 by 35.5 points, as it should. The 10-frame window ties the GRU belief on TwoRoom (78.0 against 79.0%), where MemJEPA leads both, and trails it on Reacher and PushT; only the 40-frame window catches up past the training clip. On Reacher the post-hoc 10-frame window also beats the 40-frame window (74.0 against 60.0%). We cannot explain this; each window is a single untuned setting.

Table 3: Contrasts against the no-memory model (success %; 95% intervals, Reacher clustered by case pair). L: strided frames hidden before the first planning decision (Appendix D); the first Reacher hidden column is at $L = 2$. Last-visible cache: planning from the last latent in which the entity was visible. Windows: memoryless predictors with 10- or 40-frame context. †Post hoc, on sets already opened. ‡Sets already evaluated with B_0. §Runs on the GRU belief's model. –: not run.

	Reacher (arm)		Reacher, $L = 20$
Variant	hidden	in view	hidden
No-memory model	7.5	93.8	8.2
Last-visible cache	25.3	–	5.7§
10-frame window	74.0†	94.7†	7.7
40-frame window	60.0	91.8	51.8
More compute (1200 candidates)	9.5	96.2	–
GRU belief (B)‡	93.0	94.7	58.2
− twin	+85.5	+0.8	+50.0
95% interval	[+81.3, +89.3]	[−1.0, +2.7]	[+44.7, +55.3]
MemJEPA (S)	93.2	93.0	84.5
− twin	+85.7	−0.8	+76.3
95% interval	[+81.7, +89.5]	[−2.7, +0.8]	[+71.5, +81.0]

	PushT (pusher)		TwoRoom (agent)	
Variant	hidden	in view	hidden	in view
No-memory model	3.3	51.7	29.3	84.3
Last-visible cache	15.8	–	45.0	–
10-frame window	32.5†	74.7†	78.0†	82.7†
GRU belief (B)‡	52.0	78.8	79.0	83.8
− twin	+48.7	+27.2	+49.7	−0.5
95% interval	[+42.7, +54.7]	[+21.7, +33.0]	[+43.0, +56.0]	[−2.7, +1.5]
MemJEPA (S)	57.0	72.0	83.3	87.0
− twin	+53.7	+20.3	+54.0	+2.7
95% interval	[+48.3, +58.8]	[+14.3, +26.3]	[+47.7, +60.2]	[−0.3, +5.7]

E Further state controls

Where the predictor conditions on the belief. Adding the belief at the predictor's input rather than its output gains $+15.7\,[+12.2,\ +19.2]$ hidden points on Reacher.

Freeze. Holding the GRU belief's state fixed inside the imagined candidates gives 57.0, 19.2 and 75.2% hidden and 76.0, 24.5 and 82.7% in view on Reacher, PushT and TwoRoom (held-out sets). It costs $-36.0\,[-41.8,\ -30.3]$ hidden points on Reacher. With the state frozen, the GRU belief still keeps $+49.5$ over the twin.

State swap and probe (post hoc). On the held-out sets (200 cases $\times$ 3 seeds, entity hidden), exchanging the carried state once, at the first decision, between paired episodes that share the visible window but not the hidden history gives the GRU belief 7.7, 4.3 and 22.2% and MemJEPA 7.2, 2.8 and 21.2% on Reacher, PushT and TwoRoom. These runs used a different GPU and PyTorch version from the twin's, so the comparison with the twin is descriptive. A ridge probe fitted on 3,000 training clips decodes the fingertip on the held-out Reacher set (three seeds) from the state of B_0 and of MemJEPA with $R^2 \geq 0.92$ over 40 hidden steps, against about zero from the frame latent.

How the state is advanced in imagination (Reacher confirmation set; belief added at the predictor's output). Success when each candidate's copy of the state is advanced at every imagined step with both the imagined latent and the action: 78%; with the latent only, 65%; every fifth step, 54%; with the action only, 50%; never (frozen), 29%. Against advancing at every step, every fifth step costs $-23.7\,[-28.5,\ -18.8]$ and the action alone $-27.5\,[-32.5,\ -22.5]$. Training advances h at every position. A rollout that holds h at h_t therefore calls the predictor on pairs it never saw in training:

Table 4: Per-seed differences and seed-level intervals (success points). B_0: B without its imagination-consistency term; B: the GRU belief; S: MemJEPA; twin: the no-memory model; cache: the last-visible cache. Per seed: the three per-seed differences. Case-clustered, seed-level, two-stage: the intervals defined in the text. One-sided: 95% lower bound, for the in-view criterion. Seeds: how many seeds meet the criterion alone. ⋆The three differences are equal, so the seed-level interval is the point; use the two-stage column.

View	Contrast	Per seed	Case-clustered	Seed-level	One-sided	Two-stage	Seeds
Reacher (arm)							
hidden	$B_0 - $ twin	$+84.5, +84.0, +82.0$	$+83.5\,[+79.7, +87.2]$	$[+80.2, +86.8]$	–	$[+80.3, +86.7]$	3/3
hidden	B − twin	$+84.5, +88.0, +84.0$	$+85.5\,[+81.3, +89.3]$	$[+80.1, +90.9]$	–	$[+82.0, +89.0]$	3/3
hidden	S − twin	$+86.5, +86.0, +84.5$	$+85.7\,[+81.7, +89.5]$	$[+83.1, +88.3]$	–	$[+82.7, +88.5]$	3/3
hidden	B − cache	$+64.0, +68.0, +71.0$	$+67.7\,[+62.2, +73.0]$	$[+58.9, +76.4]$	–	$[+62.5, +72.5]$	3/3
hidden	S − cache	$+66.0, +66.0, +71.5$	$+67.8\,[+62.5, +73.0]$	$[+59.9, +75.7]$	–	$[+63.2, +72.5]$	3/3
in view	$B_0 - $ twin	$-2.5, +0.0, -1.0$	$-1.2\,[-3.0, +0.7]$	$[-4.3, +2.0]$	-3.3	$[-3.3, +1.0]$	3/3
in view	B − twin	$+0.0, +2.0, +0.5$	$+0.8\,[-1.0, +2.7]$	$[-1.8, +3.4]$	-0.9	$[-1.0, +2.8]$	3/3
in view	S − twin	$+0.0, -1.0, -1.5$	$-0.8\,[-2.7, +0.8]$	$[-2.7, +1.1]$	-2.1	$[-3.0, +1.2]$	3/3
hidden	S − B	$+2.0, -2.0, +0.5$	$+0.2\,[-2.2, +2.5]$	$[-4.9, +5.2]$	–	$[-2.7, +2.8]$	–
in view	S − B	$+0.0, -3.0, -2.0$	$-1.7\,[-3.2, -0.3]$	$[-5.5, +2.1]$	–	$[-3.5, +0.3]$	–
PushT (pusher)							
hidden	$B_0 - $ twin	$+45.0, +44.5, +43.5$	$+44.3\,[+38.3, +50.3]$	$[+42.4, +46.2]$	–	$[+40.2, +48.5]$	3/3
hidden	B − twin	$+46.5, +51.5, +48.0$	$+48.7\,[+42.7, +54.7]$	$[+42.3, +55.0]$	–	$[+43.7, +53.7]$	3/3
hidden	S − twin	$+55.5, +54.5, +51.0$	$+53.7\,[+48.3, +58.8]$	$[+47.8, +59.5]$	–	$[+49.0, +58.0]$	3/3
hidden	B − cache	$+36.0, +35.5, +37.0$	$+36.2\,[+29.7, +42.5]$	$[+34.3, +38.1]$	–	$[+31.5, +40.8]$	3/3
hidden	S − cache	$+45.0, +38.5, +40.0$	$+41.2\,[+35.0, +47.2]$	$[+32.7, +49.6]$	–	$[+35.7, +46.5]$	3/3
in view	$B_0 - $ twin	$+26.5, +29.5, +26.5$	$+27.5\,[+22.0, +33.2]$	$[+23.2, +31.8]$	$+24.6$	$[+23.2, +31.8]$	3/3
in view	B − twin	$+29.0, +26.0, +26.5$	$+27.2\,[+21.7, +33.0]$	$[+23.2, +31.2]$	$+24.5$	$[+23.0, +31.3]$	3/3
in view	S − twin	$+16.0, +24.0, +21.0$	$+20.3\,[+14.3, +26.3]$	$[+10.3, +30.4]$	$+13.5$	$[+14.5, +26.5]$	3/3
hidden	S − B	$+9.0, +3.0, +3.0$	$+5.0\,[-0.5, +10.5]$	$[-3.6, +13.6]$	–	$[-1.0, +11.0]$	–
in view	S − B	$-13.0, -2.0, -5.5$	$-6.8\,[-12.3, -1.3]$	$[-20.8, +7.1]$	–	$[-13.7, -0.3]$	–
TwoRoom (agent)							
hidden	$B_0 - $ twin	$+43.5, +47.0, +45.0$	$+45.2\,[+38.5, +51.7]$	$[+40.8, +49.5]$	–	$[+40.5, +49.7]$	3/3
hidden	B − twin	$+48.0, +51.0, +50.0$	$+49.7\,[+43.0, +56.0]$	$[+45.9, +53.5]$	–	$[+45.0, +54.2]$	3/3
hidden	S − twin	$+52.5, +55.5, +54.0$	$+54.0\,[+47.7, +60.2]$	$[+50.3, +57.7]$	–	$[+49.5, +58.5]$	3/3
hidden	B − cache⋆	$+34.0, +34.0, +34.0$	$+34.0\,[+27.7, +40.5]$	$[+34.0, +34.0]$	–	$[+29.8, +38.2]$	3/3
hidden	S − cache	$+38.5, +38.5, +38.0$	$+38.3\,[+32.2, +44.7]$	$[+37.6, +39.1]$	–	$[+34.2, +42.5]$	3/3
in view	$B_0 - $ twin⋆	$-2.0, -2.0, -2.0$	$-2.0\,[-4.2, +0.2]$	$[-2.0, -2.0]$	-2.0	$[-4.0, +0.0]$	3/3
in view	B − twin	$-0.5, +0.5, -1.5$	$-0.5\,[-2.7, +1.5]$	$[-3.0, +2.0]$	-2.2	$[-2.5, +1.7]$	3/3
in view	S − twin	$+2.5, +3.5, +2.0$	$+2.7\,[-0.3, +5.7]$	$[+0.8, +4.6]$	$+1.4$	$[+0.3, +5.2]$	3/3
hidden	S − B	$+4.5, +4.5, +4.0$	$+4.3\,[+0.7, +8.2]$	$[+3.6, +5.1]$	–	$[+1.3, +7.3]$	–
in view	S − B	$+3.0, +3.0, +3.5$	$+3.2\,[+0.3, +6.3]$	$[+2.4, +3.9]$	–	$[+1.0, +5.3]$	–

an imagined latent j steps ahead with a state j steps stale. The mismatch grows with j whenever the hidden variable moves.

Frozen predictor. Training only the belief (0.89M parameters) on a frozen predictor reaches 90.5% hidden on Reacher ($+83.0\,[+79.8, +86.0]$ over the twin; $-0.5\,[-3.0, +2.2]$ against the fine-tuned belief), 47.8% on PushT ($+0.2\,[-5.0, +5.3]$ against fine-tuned) and 71.7% on TwoRoom ($-2.0\,[-6.8, +2.7]$ against 73.7% for B_0 re-run on the same GPU and PyTorch version). Fine-tuning the predictor adds 6.7 in-view points on PushT.

Observed targets (pre-registered; PushT development set, 100 cases × 3 seeds). With the observed, erased latent as target, the GRU belief on the frozen predictor scores 2.7% [1.4, 5.2] hidden, against 43.0% with clean targets ($-40.3\,[-48.0, -33.0]$) and 1.0% for the twin ($+1.7\,[-1.0, +4.3]$ over it, about 4% of the clean-target gain of $+42.0$). In view, it scores 59.7%. The two beliefs were trained on two separately built caches of encoded latents. The RSSM with observed targets scores 9.0% [6.3, 12.8] hidden against 42.0% ($-33.0\,[-41.0, -25.3]$, every seed). It gains $+8.0\,[+4.0, +12.3]$ over the twin, below the $+20$-point success criterion, and scores 74.7% in view.

Severed recurrence. A parameter-matched control with its recurrence severed is level with the twin on a fresh Reacher set drawn from each seed's validation episodes ($-1.5\,[-3.8, +0.7]$; per seed $+4.0, -5.0, -3.5$).

Negative results. Training the belief on its own detached one-step predictions changes hidden success by $-4.7\,[-6.4, -3.0]$ on Reacher (five seeds), gains nothing on TwoRoom, and fails past the training clip. A planner term that steers toward imagined model disagreement does not help. A

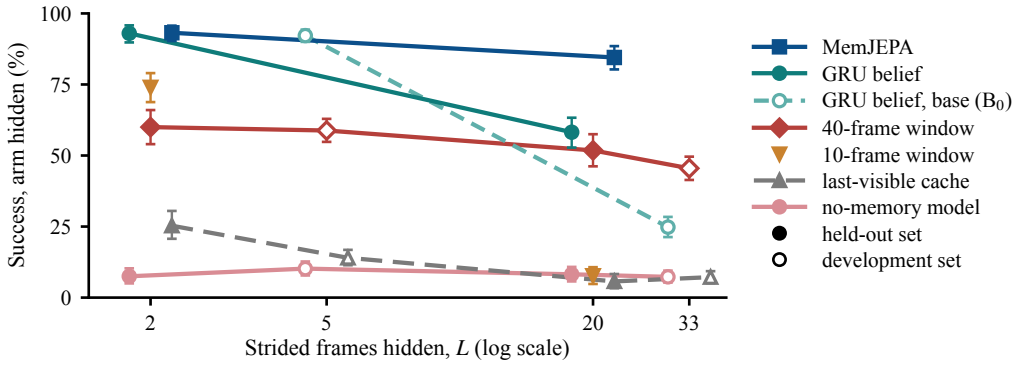


Figure 5: Reacher, arm hidden: success by strided frames hidden L (one frame is five environment steps; 200 cases $\times$ 3 seeds per point; 95% intervals). Filled markers: held-out sets ($L = 2$, 20); open: development sets ($L = 5$, 33). The GRU belief is shown at $L = 2$ and 20; the dashed series at $L = 5$ and 33 is B_0, the pre-registered belief it builds on. Points at one L are offset sideways to stay visible. Held-out values are in Table 3.

belief trained only on blank frames bridges a blank Reacher camera but not an erased arm, so the entity mask cannot be skipped. A mixed erasure schedule works on Reacher and TwoRoom but fails on PushT.

Planning cost. The GRU belief adds 892,884 parameters (4.95% of the model). At 300 candidates (RTX 4090), a replan takes 277.4, 303.3 and 129.7 ms for the twin, the GRU belief and MemJEPA. The GRU belief adds 2.6% FLOPs. With the arm hidden, the twin scores 10.8, 7.5, 8.2 and 9.5% at 150, 300, 600 and 1200 candidates, and B_0 scores 86.3, 91.0, 94.0 and 94.8%.

Training compute. Figure 3 evaluates the checkpoint saved after every training epoch on the held-out hidden sets (200 cases $\times$ 3 seeds), plus TwoRoom retrains of MemJEPA (three seeds each) stopped at 30, 100, 300 and 1,000 optimizer steps (one epoch is about 2,610 steps). On TwoRoom, MemJEPA rises from 18.8% after 30 steps to 63.7% after 1,000 steps (1.1 GPU-minutes) and 85.3% after one epoch (2.9 GPU-minutes on an RTX 4090) and stays flat through epoch 5 (82.7%). The no-memory model stays at 28.5, 31.5 and 29.3% at epochs 1, 3 and 5 (MemJEPA $-$ no-memory model: $+56.8\,[+50.3, +63.0]$, $+54.2\,[+47.5, +60.7]$ and $+53.3\,[+47.0, +59.5]$). On PushT, MemJEPA rises from 37.8% at epoch 1 to 55.3% at epoch 3 and 57.0% at epoch 5 (about 50 GPU-minutes), against 2.2 and 3.3% for the no-memory model at epochs 1 and 5. More training does not give the no-memory model, trained on the same erased clips, success with the entity hidden, so the gap comes from the state, not from training length. The no-memory model was evaluated at epochs 1, 3 and 5 on TwoRoom and 1 and 5 on PushT. The epoch-5 results reproduce the main records (PushT case for case; TwoRoom within 4 of 200 cases per seed, 82.7 against 83.3%).

F A second backbone: DINO-WM

Results. Each design and its recipe-matched twin are retrained with DINO-WM's frozen DINOv2 encoder and its own training code. Where the entity covers under 5% of the patches, every variant and its twin use an entity-mask-weighted loss ($\lambda = 10$). With the entity hidden, the GRU belief beats the trained twin by $+13.0\,[+9.0, +17.0]$, $+5.0\,[+1.8, +8.3]$ and $+34.2\,[+28.0, +40.2]$ points on Reacher, PushT and TwoRoom, and MemJEPA by $+40.0\,[+35.0, +44.8]$, $+19.3\,[+14.8, +23.8]$ and $+43.2\,[+38.0, +48.3]$. In view, the GRU belief changes success by -5.2, $+7.7$ and $+0.0$, and MemJEPA by -26.5, -32.3 and -15.2. Three of the six hidden contrasts meet the $+20$-point criterion. Only the GRU belief meets the in-view criterion: on TwoRoom, and on PushT when seed variation is ignored (seed-level lower bound -10.7; each of the three seeds meets it alone). The gains are smaller than on LeWM, and MemJEPA pays an in-view cost on this backbone.

Does the erasure schedule cause the in-view cost? A pre-registered follow-up on PushT asked this. MemJEPA retrained without erased clips ($p_{\text{full}} = 1$) scores 28.8% [25.4, 32.6] in view, against 23.3% for MemJEPA and 55.7% for the twin (600 episodes each). That is $+5.5\,[+1.5,\ +9.5]$ over MemJEPA (per seed $+10.0$, $+7.0$, -0.5). With the pusher hidden it scores 2.5%, level with the twin (3.0%). The schedule therefore accounts for 5.5 of the 32.3-point gap. The rest is unexplained. A second follow-up, to run only if the gain reached 10 points, did not run.

G Counterfactual training

Branches. Start states are every tenth start of a non-excluded training episode. At each start state the simulator is restored to the recorded state and run under three action sequences. Two are near-factual: the recorded actions plus AR(1) noise ($\sigma = 0.1$ of the action standard deviation, $\rho = 0.9$). One is a donor: the actions of a clip from another episode. Each branch is rendered clean and erased, then encoded by the frozen encoder. Branch clips replace 25% of clip draws, chosen deterministically from the seed and training step, so every variant sees the same draws. S + branches keeps S's loss, and twin + branches keeps the twin's. S + energy score replaces S's decoder MSE by the energy score [25] on particles drawn from its state. S + both adds to S + energy score the target set {factual, two near-factual} at near-factual start states, with donor branches as ordinary clips. Re-simulating the recorded actions reproduces the data bit-exactly on TwoRoom (200 of 200 checked start states). On Reacher the physics is bit-exact, with a one-level intensity difference on 0.2% of frames. PushT's stored states lack velocities, so there the re-simulated frames instead pass a check on the cosine similarity of their frozen-encoder latents to the recorded ones. S retrained on the regenerated PushT training latents is bit-identical to S at seed 0, so the regeneration explains none of the gain.

Planning evaluations. Success in Table 5 pools three seeds. The fresh PushT set is 300 cases (150 pairs), never opened before and excluded from every training; there S scores 69.6% in view. The TwoRoom set is the 200-case never-opened set of Appendix H, with S and the twin retrained on the same GPU and PyTorch version. The Reacher set is the 200-case never-opened set of new windows. Each evaluation was pre-registered and run once. The pre-registered contrast (S + branches) $-$ S ($\geq +5$ points, interval above 0) passes on PushT and fails on the other two tasks. The PushT gain therefore did not transfer to TwoRoom or Reacher, and one task supports no mechanism claim. S + branches still beats twin + branches by $+50.5\,[+44.2,\ +56.8]$ on TwoRoom and $+88.7\,[+84.8,\ +92.0]$ on Reacher, and it stays within the in-view criterion ($+1.2$ and -0.3 against twin + branches).

Energy score. When the planner conditions on the state's mean, the energy score adds nothing on top of branches on the fresh PushT set: the gain of S + both over S + energy score, minus the gain of S + branches over S, is $-3.3\,[-8.0,\ +1.3]$. Branches also lift the energy-score variant: with four sampled futures, S + both beats S + energy score by $+10.0\,[+6.7,\ +13.3]$, positive on every seed. Sampling futures at planning time has no detectable effect for S + energy score or S + both (difference in differences against S: S + energy score $-1.2\,[-4.8,\ +2.3]$, S + both $+1.2\,[-2.4,\ +4.9]$). Training on simulator branches raises the planner's success when it conditions on the mean; sampling adds nothing beyond S. Energy-score training removes the floor on the state's spread and makes the spread track the error where the planner computes its cost. The entity position decoded from the state, however, stays under-dispersed: it fails the pre-registered coverage and spread-to-error criteria. The energy score did not produce calibrated spread.

Belief diagnostic. The planner's one-step prediction from erased frames is rolled 1 to 9 strided steps from held-out start states. They lie in episodes excluded from training and disjoint from the branch cache (PushT 300, TwoRoom 300, Reacher 182). Rollouts follow the recorded actions (on) and three new branches (off). The error is the MSE per latent dimension relative to S, with intervals over start states. We pre-registered that S + branches and S + both would lower the error off the trajectory and leave it unchanged on it, and that S + energy score would change neither. On PushT the off-trajectory part held and the on-trajectory part failed. Branch training lowers the error on both, by 19.7 percentage points more off the trajectory ($-19.7\,[-23.5,\ -16.1]$), and the gap grows with

Table 5: Training variants of MemJEPA (Section 2), with intervals; variants as defined in Appendix G. Top: hidden success and the paired difference to the reference variant (ref.). Bottom: the state's latent error relative to S (%) on held-out clips, along recorded actions (on) or counterfactual branches (off). Intervals are 95%. Seeds improving: seeds with higher success or lower error. [d] 100 development start states. Variants not run on a task are omitted.

Task	Variant	Success (%)	vs.	Difference	Interval	Seeds improving
Reacher	S	94.8	ref.			
	S + branches	96.0	S	$+1.2$	$[-0.5, +2.8]$	2/3
	twin	8.7	ref.			
	twin + branches	7.3	twin	-1.3	$[-3.7, +1.0]$	0/3
PushT	S	54.2	ref.			
	S + energy score	57.3	S	$+3.1$	$[-0.6, +6.8]$	3/3
	S + branches	65.1	S	$+10.9$	$[+7.3, +14.6]$	3/3
	S + both	64.9	S + energy score	$+7.6$	$[+4.0, +11.1]$	3/3
	twin	5.4	ref.			
	twin + branches	8.1	twin	$+2.7$	$[+0.3, +5.0]$	3/3
TwoRoom	S	85.2	ref.			
	S + branches	85.2	S	0.0	$[-2.2, +2.2]$	1/3
	twin	31.2	ref.			
	twin + branches	34.7	twin	$+3.5$	$[-0.2, +7.3]$	2/3

Task	Rollout	Variant	Latent error vs. S (%)	Interval	Seeds improving
Reacher	off	S + branches	-2.1	$[-4.3, +0.3]$	3/3
PushT	on	S + energy score	-2.5	$[-5.3, +0.4]$	3/3
		S + branches	-12.1	$[-15.3, -8.8]$	3/3
		S + both	-7.3	$[-10.8, -3.7]$	3/3
	off	S + energy score	-0.9	$[-2.4, +0.5]$	3/3
		S + branches	-31.8	$[-33.9, -29.6]$	3/3
		S + both	-23.4	$[-26.0, -20.8]$	3/3
TwoRoom	off	S + energy score	$+8.0$[d]	$[+1.5, +13.7]$	0/3
		S + branches	-11.7	$[-13.6, -9.8]$	3/3

the horizon (step 9, off: S 56.8, S + branches 36.4, $\times 10^{-3}$). A re-simulated factual control matches the on-trajectory set (-13.5%), so the branch pipeline is not the cause.

On TwoRoom, as on PushT, the gain is larger off the recorded trajectory than on it (difference $-9.2\,[-11.4, -6.9]$ percentage points, every seed). There S + energy score raised off-trajectory error, so it is not a null there ($+8.0\,[+1.5, +13.7]$ off, $+12.4\,[+1.7, +23.0]$ on, 100 development start states). On Reacher the gain off the recorded trajectory is not larger than on it, so the PushT and TwoRoom asymmetry did not replicate (off $-2.1\,[-4.3, +0.3]$, on $-1.3\,[-4.8, +2.5]$, difference $-0.8\,[-3.3, +1.6]$). S's own error is 2.5 times higher on branches than on recorded clips on PushT, 1.05 times on TwoRoom and 1.06 times on Reacher.

Representation (pre-registered, post hoc). On the 300 PushT diagnostic start states we measured mutual 10-nearest-neighbor alignment [26]. From erased frames, the state recovers, above chance, the in-view neighbors that its erased frame misses: 0.584 [0.567, 0.600] of them for S, against 0.033 by chance, 0.153 for an untrained network and 0.595 for S's own state in view. The state also recovers the neighbors that DINO-WM's encoder loses under erasure (0.321, against 0.184 for the erased LeWM latent). What it recovers is therefore visible to an encoder it never trained on. Branch training lowers the state's latent error off the trajectory but does not raise how many of the erased frame's missed in-view neighbors it recovers ((S + branches) $-$ S: $+0.01\%\,[-0.41, +0.45]$). The predictions of S + branches, by contrast, recover $+3.0\%\,[+2.6, +3.4]$ more. Branches change the predictions, not the state's geometry. S + energy score changes the state's recovery of the missed neighbors ($+2.8\%$ on the recorded trajectory, $+3.3\%$ off it), although it changed neither planning nor latent error on PushT.

Table 6: Never-opened sets, each evaluated once (200 cases $\times$ 3 seeds per variant). Top: success (%). Bottom: paired hidden differences with 95% intervals. B_0 is the GRU belief the criteria were fixed on; B adds the imagination-consistency term; S is MemJEPA. Cache: the last-visible cache, which keeps the latent of the last frame showing the entity (Appendix C). The PushT row is the first never-opened PushT set (200 cases). It is a different set from the held-out PushT set of Table 1 and from the fresh PushT set (300 cases) of Tables 5 and 9, so its B and S values differ from theirs.

	Hidden					In view		
Task	B	S	B_0	Twin	Cache	B	S	Twin
Reacher	94.3	94.8	94.0	8.7	27.7	96.5	96.2	96.5
PushT	55.7	59.3	52.8	4.2	18.2	74.5	73.8	57.7
TwoRoom	76.8	85.2	74.5	31.2	55.2	82.3	86.5	84.0

Task	B − twin	S − twin	B − B_0
Reacher	$+85.7\,[+83.0,\ +88.3]$	$+86.2\,[+83.2,\ +88.8]$	$+0.3\,[-1.7,\ +2.5]$
PushT	$+51.5\,[+45.5,\ +57.3]$	$+55.2\,[+49.7,\ +60.7]$	$+2.8\,[-0.3,\ +6.0]$
TwoRoom	$+45.7\,[+39.0,\ +52.3]$	$+54.0\,[+47.7,\ +60.3]$	$+2.3\,[-0.5,\ +5.2]$

H Never-opened sets and evaluation reproducibility

Table 6 tests whether the primary contrasts replicate. Each task's never-opened set was evaluated once, in one blind batch, with every criterion fixed in advance: new windows on held-out Reacher episodes, a PushT set never opened, and a new TwoRoom draw with every variant retrained on one GPU and PyTorch version.

Hardware sensitivity. Evaluation depends on the GPU and PyTorch version. On the same 600 previously opened TwoRoom cases, B_0 scores 453 successes on the original RTX 4090 setup and 439 on the RTX 5090 setup used for the new evaluations (-2.3 points), the twin 171 and 158 (-2.2), and B 474 and 478 ($+0.7$). B_0's 453 and the twin's 171 are their first recorded evaluations (75.5 and 28.5%, Appendix B); the evaluations of Table 1 use a later re-run of both (74.5 and 29.3%; B_0 in Appendix B). The two setups also differ in PyTorch version. B's seed-0 weights are bit-identical across them, so the drift is in evaluation. It is up to about 2 points, against contrasts of $+45$ to $+54$. Every contrast in this paper is computed within one setup; comparisons across setups are descriptive.

I External baselines

Scope. The external comparison is partial. Released memoryless planners (LeWM; DINO-WM on PushT), the released GCBC and behavior cloning trained on our latents with and without memory all run on our cases with our evaluator (Table 7). No released planner with memory runs over a frozen encoder. R2I [27] as published is a pixel-space, Dreamer-style agent trained with an actor-critic. Moved onto our encoder, planner and clips, it becomes a re-implementation, not a baseline. We specified and pre-registered such a port, with nine deviations from R2I fixed in advance, among them the frozen encoder, CEM in place of the actor-critic and short training clips. We did not run it, so no external method with memory is compared here. Instead, R2I's target ported to latents, the erased input's own latent, is tested on our own designs (Section 2, Appendix E). On the PushT development set it leaves the GRU belief at 2.7% hidden, within $+1.7\,[-1.0,\ +4.3]$ of the twin, and the RSSM at 9.0%. On Reacher and TwoRoom the erased frame is one image, so an observed target carries no entity.

GCBC. We run stable-worldmodel's [28] released goal-conditioned behavior cloning, with one patch for our data split. It is memoryless, is queried every environment step and is trained for N of its 100 epochs. A pre-registered fidelity check compares its in-view success (seed 0, 100 episodes) with the published value. On TwoRoom $N = 3$, and the check scored 100/100 (LeWM reports 100 [10]). On PushT $N = 2$: the check missed at $N = 1$ (56/100) and passed on its one pre-registered retry (74/100, published 75). By pre-registration it is not run on Reacher. On 3 seeds

$\times$ 200 never-opened cases it scores 1.3% [0.7, 2.6] hidden and 53.2% [49.2, 57.1] in view on PushT, and 14.7% [12.1, 17.7] and 99.2% [98.1, 99.6] on TwoRoom (Wilson intervals).

Behavior cloning on our latents. The recurrent BC is a GRU policy on the frozen LeWM latents trained with the erasure recipe. We also query it at the planner's decision rate. The memoryless BC is the same recipe without memory (897,130 parameters). Table 7 gives the pre-registered outcome per task. A contrast is a win or a loss when the 95% interval of the difference excludes 0; it meets the memory criterion at $+20$ or more.

TwoRoom. With the entity hidden, B and S both win against GCBC and the memoryless BC and meet the memory criterion. Both lose against the recurrent BC: B 76.8 against 95.3% ($-18.5\,[-23.8,\ -13.3]$) and S 85.2 against 95.3% ($-10.2\,[-15.5,\ -5.0]$). In view every BC variant is near ceiling (99.2 to 99.7%), B and S trail by 12.7 to 17.3 points, and no in-view contrast clears -5. At 3 to 5$\times$ the planner's decision rate the recurrent BC reaches 95.3%, against 76.8% for B and 85.2% for S; at the planner's rate it also reaches 95.3%. The rate makes no difference there because the erased frame is constant. Memory and erasure training suffice there. PLDM reports GCBC at 86.0 on good data and 8.4 without door-passing trajectories (its Table 2) [22], so this result is conditional on data quality, which this paper did not vary.

Reacher. B and S win against the recurrent BC by about 9 points, below the $+20$ memory criterion, and against the memoryless BC by 94, meeting it. Every in-view lower bound is above -5.

PushT. B and S win against GCBC and the memoryless BC and meet the criterion, with in-view lower bounds of $+14.8$ to $+19.7$. The recurrent BC did not reach its pre-registered in-view level after its one retry (development set, 9% in view against a criterion of 35%). Its contrasts are reported, but no win is claimed over it, and the pre-registered check that the recurrent BC comes within 10 points of GCBC in view fails on PushT.

Memory inside a policy, the recurrent minus the memoryless BC with the entity hidden, is $+60.5$ on TwoRoom and $+85.0$ on Reacher. On PushT the two are tied at 5 to 7%. Of eight pre-registered predictions, five held and three failed: GCBC in view at least the twin's on PushT (53.2 against 57.7%, interval covering 0), GCBC hidden within 15 points of the twin on TwoRoom (16.5), and recurrent BC hidden on PushT in 25 to 60% (7.2). All three datasets are near-expert, the setting that favors BC. BC's training goals also differ from the evaluated $+25$-step goal, as the pre-registration states.

J Longer hidden spans

The pre-registered hard-occlusion test keeps erasure and lengthens the span L, the number of strided frames the entity has been hidden at the first planning decision (PushT 2, 10, 20, 40; TwoRoom 2, 6, 10; Reacher 2, 20, 33, the most the episodes admit). The entity stays hidden to the end, and the planner gets 50 steps. A development check, fixed before any evaluation ran, picks per task the shortest span at which S is below 60% hidden and at least 10 points under its in-view success, else the longest span. Development numbers choose the span and never decide the test.

Development check (S hidden / in view, 3 seeds pooled). TwoRoom: 85.7 / 87.7, 83.5 / 82.7 and 86.5 / 83.3% at 2, 6 and 10 frames. Reacher: 96.2 / 98.1, 86.9 / 91.5 and 81.7 / 94.4% at 2, 20 and 33. Neither task qualified, so the branch tests below were not run there, and the S-only evaluations on their fresh sets are not used here. PushT: 42.0 / 69.3, 22.7 / 42.3, 16.0 / 36.7 and 7.3 / 39.7% at 2, 10, 20 and 40. Every span qualified, so the chosen span is $L = 2$.

Does S fail at the chosen span? We pre-registered that at $L = 2$ S would be below 60% hidden and at least 10 points under its in-view success. S hidden is 61.7% (Table 8), at or above 60% on every seed, so the hidden half fails, while the in-view gap holds ($+11.3\,[+7.2,\ +15.5]$). S's development failure did not replicate at that span, so the branch results below are reported, not treated as a test.

Branches at the chosen span. This is a replication at the pre-registered span, not a hard-occlusion result. (S + branches) $-$ S is $+6.3\,[+1.7,\ +11.0]$, positive on every seed. The same branches raise

Table 7: External baselines and behavior cloning on our cases with our evaluator. Top: success (%) with the entity hidden / in view on the never-opened sets (3 seeds $\times$ 200 cases, 600 episodes per variant and view). Recurrent BC: a GRU policy on the frozen latents trained with the erasure recipe, queried at 3 to 5$\times$ the planner's decision rate or at that rate (at planner rate); memoryless BC: the same recipe without memory. GCBC, LeWM and DINO-WM are the released memoryless baselines; GCBC is trained for N of its 100 epochs (PushT 2, TwoRoom 3). †200 cases of each never-opened set. ‡PushT's held-out set (600 episodes). Bottom: hidden differences with 95% intervals and the pre-registered outcome; BC memory gain is recurrent minus memoryless BC. Win or loss: the interval lies above or below 0; tie: it includes 0; none: no claim. Every win against the memoryless BC or GCBC is also at least $+20$ points, the memory criterion; the Reacher wins against the recurrent BC are not. *The recurrent BC did not reach its in-view level on PushT after its one retry, so no claim is made against it there. –: not run.

	Reacher	PushT	TwoRoom
GRU belief (B)	94.3 / 96.5	55.7 / 74.5	76.8 / 82.3
MemJEPA (S)	94.8 / 96.2	59.3 / 73.8	85.2 / 86.5
No-memory model	8.7 / 96.5	4.2 / 57.7	31.2 / 84.0
Recurrent BC	85.3 / 98.5	7.2 / 19.3*	95.3 / 99.7
at planner rate	85.5 / –	5.3 / –*	95.3 / –
Memoryless BC	0.3 / 96.8	5.0 / 48.7	34.8 / 99.5
GCBC (released)	–	1.3 / 53.2	14.7 / 99.2
LeWM (released)†	8.5 / 96.5	3.0 / 68.0	27.5 / 81.5
DINO-WM (released)‡	–	1.0 / 88.5	–
B $-$ recurrent BC	$+9.0\,[+6.2, +11.8]$ win	$+48.5\,[+42.7, +54.3]$ none*	$-18.5\,[-23.8, -13.3]$ loss
S $-$ recurrent BC	$+9.5\,[+6.7, +12.3]$ win	$+52.2\,[+46.5, +57.7]$ none*	$-10.2\,[-15.5, -5.0]$ loss
B $-$ memoryless BC	$+94.0\,[+92.0, +95.8]$ win	$+50.7\,[+44.8, +56.5]$ win	$+42.0\,[+34.0, +49.8]$ win
S $-$ memoryless BC	$+94.5\,[+92.7, +96.2]$ win	$+54.3\,[+48.8, +59.7]$ win	$+50.3\,[+41.8, +58.5]$ win
B $-$ GCBC	–	$+54.3\,[+48.3, +60.3]$ win	$+62.2\,[+55.5, +68.7]$ win
S $-$ GCBC	–	$+58.0\,[+52.5, +63.3]$ win	$+70.5\,[+63.5, +76.8]$ win
BC memory gain	$+85.0\,[+82.0, +87.8]$ win	$+2.2\,[-0.5, +4.8]$ tie*	$+60.5\,[+53.8, +66.8]$ win

Table 8: Longer hidden spans on the PushT span set (3 seeds $\times$ 200 cases, 600 episodes per variant and span): success (%) with Wilson 95% intervals. Not run by pre-registration: twin and B in view; twin, twin + branches and B hidden at 10 and 20 frames (B also at 40). A branch-trained B was pre-registered but never trained.

	Hidden, frames hidden at the first decision				In view
Variant	2	10	20	40	2
S	61.7 [57.7, 65.5]	32.8 [29.2, 36.7]	18.7 [15.8, 22.0]	10.2 [8.0, 12.8]	73.0 [69.3, 76.4]
S + branches	68.0 [64.2, 71.6]	46.3 [42.4, 50.3]	28.8 [25.4, 32.6]	15.0 [12.4, 18.1]	79.5 [76.1, 82.5]
twin	5.2 [3.7, 7.2]	–	–	5.3 [3.8, 7.4]	–
twin + branches	10.3 [8.1, 13.0]	–	–	6.2 [4.5, 8.4]	69.2 [65.4, 72.7]
B	49.0 [45.0, 53.0]	–	–	–	–

the twin by $+5.2\,[+2.2,\ +8.3]$, so we take the gain net of that: $+1.2\,[-4.5,\ +6.8]$, 2 of 3 seeds positive. The criterion is not met.

Long span (40 frames). S hidden is 10.2%, and the twin's branch effect is $+0.8\,[-1.5,\ +3.2]$. (S + branches) − S is $+4.8\,[+1.5,\ +8.2]$, positive on all three seeds but below the $+5$ criterion: where erasure-trained memory fails under long occlusion on PushT, counterfactual branch training does not repair it. The in-view gap at 40 frames was not measured on the span set (development: 32.3 points).

Gain against span. The gain is $+6.3$, $+13.5$, $+10.2$ and $+4.8$ at 2, 10, 20 and 40 frames (intervals $[+1.7, +11.0]$, $[+8.3, +18.5]$, $[+6.2, +14.0]$, $[+1.5, +8.2]$). It does not grow with the hidden span. On PushT the span is confounded with the prefix action size (0.07 at 2 frames to 0.0067 at 40), so this is not taken as an effect of span alone.

In view. (S + branches) − (twin + branches) in view is $+10.3\,[+5.0,\ +15.8]$, with one-sided lower bound $+5.8$: the pre-registered in-view check passes. Reported without a criterion: (S + branches) − B $+19.0\,[+13.3,\ +24.7]$ and S − B $+12.7\,[+6.5,\ +18.8]$ at 2 frames; (S + branches) − (twin + branches) $+57.7\,[+51.7,\ +63.5]$ at 2 and $+8.8\,[+5.2,\ +12.5]$ at 40; S − twin $+56.5\,[+50.5,\ +62.2]$ at 2 and $+4.8\,[+1.7,\ +8.0]$ at 40. With the branch test failing on PushT, no hard-occlusion claim is made.

Disclosures. The PushT span set reuses episodes from sets opened earlier, with new windows, prefix paths and goals. No variant was tuned on them. The branch-trained B was never trained, so it has no row. Evaluations that depended on other spans or on a task that did not qualify were never run.

K Open-loop and occluder baselines

Four pre-registered tests, each evaluated once, ask whether an open-loop rollout of the no-memory model matches the memory variants and how the variants fare under occluders they were not trained on. Predictions stated as a range or a threshold are judged on the point estimate, with the interval printed beside it. On PushT the intervals resample cases individually rather than by pair; a pair-clustered interval, not pre-registered, gives $[+27.3, +38.3]$ for B − open-loop rollout.

Open-loop rollout (Table 9) needs no training. From the first hidden frame on, every hidden slot of the no-memory model's window holds the twin predictor's own latent. That latent is rolled open-loop with the executed action blocks and never re-anchored while the entity is hidden. On PushT this discards the visible block at hidden slots, so the PushT open-loop row and the PushT differences against it include that handicap by construction. S + visibility oracle is S told which frames are hidden: it updates its state from the observation on visible frames and only predicts at hidden ones. It is a reference only.

Region occlusion (Table 10): on TwoRoom the agent is erased whenever its true position lies in a fixed region, the lower half of its starting room, so it reappears when it leaves the region. B, S and the twin are the TwoRoom checkpoints of the never-opened set (Appendix H), evaluated on a new set of 71 cases (no more episodes meet the rule).

Moving patch (Table 11): the moving-patch occluder of stable-worldmodel [28], with sides 0.4 to 0.5 of the frame, is applied to the in-view frames with erasure off. No variant was trained with it. It covers the entity wholly at a decision window in 174 to 183 of 600 runs per variant on Reacher, 8 to 12 of 600 on TwoRoom and 14 to 39 of 300 on the PushT development set. The PushT patch evaluations were first run for Appendix M and are reused here unchanged.

Proprioceptive oracle (pre-registered; never-opened Reacher and TwoRoom sets, fresh PushT set). The twin's recipe with the controlled entity's z-scored true state appended to every real frame's action token (absent at imagined positions; dropped out with probability 0.5 in training). Hidden / in view: Reacher 94.5 / 96.0 (twin 8.7 / 96.5; MemJEPA 94.8; GRU belief 94.3), PushT 43.0 / 65.8 (twin 5.4 / 56.3; MemJEPA 54.2; GRU belief 55.8), TwoRoom 34.3 / 83.8 (twin 31.2 / 84.0; MemJEPA 85.2; GRU belief 76.8). On Reacher the oracle comes within $-2.0\,[-3.8,\ -0.2]$ of the

Table 9: Open-loop rollout against the memory variants, entity hidden: successes k/n and success (%) with Wilson 95% intervals, 3 seeds pooled; below, paired differences with 95% intervals. Reacher and TwoRoom: the never-opened sets (200 cases); PushT: the fresh set of Table 5. Open-loop rollout: the no-memory model rolled open-loop on the executed actions. S + visibility oracle: S told which frames are hidden.

	Reacher	PushT	TwoRoom
Open-loop rollout (twin)	578/600, 96.3 [94.5, 97.6]	207/900, 23.0 [20.4, 25.9]	471/600, 78.5 [75.0, 81.6]
No-memory model	52/600, 8.7 [6.7, 11.2]	49/900, 5.4 [4.1, 7.1]	187/600, 31.2 [27.6, 35.0]
GRU belief (B)	566/600, 94.3 [92.2, 95.9]	502/900, 55.8 [52.5, 59.0]	461/600, 76.8 [73.3, 80.0]
MemJEPA (S)	569/600, 94.8 [92.8, 96.3]	488/900, 54.2 [51.0, 57.5]	511/600, 85.2 [82.1, 87.8]
S + visibility oracle	566/600, 94.3 [92.2, 95.9]	498/900, 55.3 [52.1, 58.6]	497/600, 82.8 [79.6, 85.6]
Open-loop rollout − twin	$+87.7\,[+85.0, +90.2]$	$+17.6\,[+13.2, +22.0]$	$+47.3\,[+40.7, +53.8]$
B − open-loop rollout	$-2.0\,[-3.7, -0.5]$	$+32.8\,[+27.4, +38.2]$	$-1.7\,[-5.3, +1.8]$
S − open-loop rollout	$-1.5\,[-3.2, +0.2]$	$+31.2\,[+26.2, +36.4]$	$+6.7\,[+2.8, +10.7]$
S + visibility oracle − S	$-0.5\,[-2.0, +1.0]$	$+1.1\,[-2.6, +4.8]$	$-2.3\,[-4.5, -0.2]$

Table 10: Region-occluded TwoRoom (71 cases $\times$ 3 seeds = 213 per variant): successes k/n and success (%), overall with Wilson 95% intervals and by stratum as point estimates: the 5 cases with a visible frame before the first decision and the 66 without. B, S and the twin are the TwoRoom checkpoints of the never-opened set. The open-loop rollout row is descriptive only; in 69 of the 71 cases on each seed its rollout starts from the latent of an erased frame.

	All cases		Region occluded, by stratum	
	Region occluded	In view	Seen before (5)	Never seen (66)
B	23/213, 10.8 [7.3, 15.7]	188/213, 88.3 [83.2, 91.9]	11/15, 73.3	12/198, 6.1
S	29/213, 13.6 [9.6, 18.9]	198/213, 93.0 [88.7, 95.7]	12/15, 80.0	17/198, 8.6
Twin	34/213, 16.0 [11.7, 21.5]	193/213, 90.6 [85.9, 93.8]	5/15, 33.3	29/198, 14.6
Open-loop rollout	106/213, 49.8 [43.1, 56.4]	–	10/15, 66.7	96/198, 48.5

Table 11: Moving patch over the in-view frames: successes k/n and success (%), 3 seeds pooled; with the patch on, Wilson 95% intervals; with it off, point estimates from each variant's archived in-view evaluation. PushT rows are on the development set. †Seed 0 only.

Task	Variant	Patch on	Patch off
Reacher	S	68/600, 11.3 [9.0, 14.1]	577/600, 96.2
	S + branches	76/600, 12.7 [10.2, 15.6]	574/600, 95.7
	twin	79/600, 13.2 [10.7, 16.1]	579/600, 96.5
	twin + branches	78/600, 13.0 [10.5, 15.9]	576/600, 96.0
	B	81/600, 13.5 [11.0, 16.5]	579/600, 96.5
TwoRoom	S	439/600, 73.2 [69.5, 76.6]	519/600, 86.5
	S + branches	433/600, 72.2 [68.4, 75.6]	512/600, 85.3
	twin	416/600, 69.3 [65.5, 72.9]	504/600, 84.0
	twin + branches	410/600, 68.3 [64.5, 71.9]	505/600, 84.2
	B	412/600, 68.7 [64.8, 72.2]	494/600, 82.3
PushT (development)	S	1/300, 0.3 [0.1, 1.9]	208/300, 69.3
	S + branches	3/300, 1.0 [0.3, 2.9]	84/100, 84.0†
	twin	0/300, 0.0 [0.0, 1.3]	135/300, 45.0
	twin + branches	2/300, 0.7 [0.2, 2.4]	63/100, 63.0†
	B	0/300, 0.0 [0.0, 1.3]	195/300, 65.0

twin in view, and MemJEPA recovers the same share of the gap from pixels (ratio 1.004 [0.980, 1.029]). On PushT the oracle recovers part of the in-view level ($-13.3\,[-18.1,\ -8.6]$) and also gains in view ($+9.4\,[+5.3,\ +13.7]$), so the remaining gap is about using the state, not memory. On TwoRoom the oracle did not learn to use the state input ($+3.2\,[0.0,\ +6.5]$ over the twin, hidden).

Contrasts under the patch (paired, 95% intervals). Reacher: S − twin $-1.8\,[-5.2,\ +1.7]$, B − twin $+0.3\,[-3.2,\ +3.7]$, (S + branches) − S $+1.3\,[-1.2,\ +3.8]$, (twin + branches) − twin $-0.2\,[-2.7,\ +2.3]$, (S + branches) − (twin + branches) $-0.3\,[-3.5,\ +2.8]$. TwoRoom, in the same order: $+3.8\,[-2.0,\ +9.8]$, $-0.7\,[-5.2,\ +3.8]$, $-1.0\,[-4.0,\ +1.8]$, $-1.0\,[-4.2,\ +2.0]$, $+3.8\,[-1.7,\ +9.3]$. PushT development set: $+0.3\,[0.0,\ +1.0]$, $0.0\,[0.0,\ 0.0]$, $+0.7\,[-0.7,\ +2.3]$, $+0.7\,[0.0,\ +1.7]$, $+0.3\,[-1.3,\ +2.3]$. Differences in differences against patch off, S − twin and B − twin: Reacher $-1.5\,[-5.2,\ +2.0]$ and $+0.3\,[-3.3,\ +4.0]$; TwoRoom $+1.3\,[-4.8,\ +7.5]$ and $+1.0\,[-3.2,\ +5.5]$; PushT development set $-24.0\,[-33.7,\ -14.3]$ and $-20.0\,[-33.3,\ -7.0]$. On that set the in-view gaps S − twin and B − twin are $+24.3$ and $+20.0$, so the PushT differences in differences are the patch removing an in-view gap, not a hidden-view memory gain.

Open-loop rollout: predictions and outcomes. We predicted that the open-loop rollout beats the twin on Reacher with the interval above 0, by $+10$ to $+40$. It beats it by $+87.7\,[+85.0,\ +90.2]$, outside the predicted range: hidden on Reacher, the open-loop rollout (96.3%) equals the twin's own in-view success (96.5%). We predicted that B and S each beat the open-loop rollout by at least $+10$ with the interval above 0 on every task, and that the prediction fails if the open-loop rollout comes within 5 points of B or S on any task. It holds on PushT and fails on Reacher (B -2.0, S -1.5) and TwoRoom (B -1.7). On Reacher the open-loop rollout is above B by 2.0 with an interval excluding 0. On TwoRoom S − open-loop rollout is $+6.7\,[+2.8,\ +10.7]$, below $+10$ but not within 5. On these two tasks a memoryless predictor rolled open-loop through the hidden span with the executed actions matches the belief, so it accounts for most of the memory variants' gain there, and the memory claim is confined to PushT. We expected the open-loop rollout to stay within ±10 of the twin on PushT and TwoRoom (descriptive); it did not ($+17.6$ and $+47.3$). We predicted that the visibility oracle changes S by $[-5, +5]$ on Reacher and TwoRoom and $[0, +10]$ on PushT; it held (-0.5, -2.3 and $+1.1$). On TwoRoom the oracle is slightly below S ($-2.3\,[-4.5,\ -0.2]$).

Region occlusion: predictions and outcomes. We predicted that B and S beat the twin under region occlusion by at least $+10$ with the interval above 0. Both fail: $-5.2\,[-13.6,\ +3.3]$ and $-2.3\,[-10.8,\ +5.6]$. The pre-registered reading of this failure was that the twin recovers once the agent is visible again and the belief's advantage is not measurable, on a set where 66 of 71 cases never show the agent before the first decision, so no earlier observation exists to remember. In fact no variant recovers on this set (region 10.8 to 16.0% against in view 88.3 to 93.0%). A pre-registered qualifier interprets the failure as planning from a never-seen start, so it is not attributed to memory. We also predicted that region occlusion costs at most 10 points against in view. It fails for B ($-77.5\,[-85.9,\ -68.1]$) and S ($-79.3\,[-86.9,\ -70.9]$): the belief does not recover when the entity reappears, with the same qualifier. Descriptively, the open-loop rollout beats the twin under region occlusion by $+33.8\,[+21.6,\ +45.5]$. On the 5 cases with a frame seen before the decision, B and S lead the twin by $+40.0\,[+6.7,\ +73.3]$ and $+46.7\,[+13.3,\ +80.0]$. Of those 5 cases, 2 start outside the region and 3 start inside it and leave it for 1 to 2 prefix steps; 69 of 71 cases start inside the region.

Moving patch: predictions and outcomes. We predicted every variant below 15% under the patch, both overall and where the entity is never more than half covered; the prediction fails if any variant keeps at least 60% on a task, and results in between neither hold nor fail. It fails on TwoRoom, where every variant keeps 68 to 73%. On Reacher every variant is below 15% overall but at 24 to 31% where the entity is never more than half covered, so it neither holds nor fails there. It holds on the PushT development set. We predicted, expecting failure, that S and B beat the twin under the patch by at least $+20$; this fails everywhere. We predicted (S + branches) − S under the patch within ±5; it held ($+1.3$, -1.0 and $+0.7$).

L What erasure training makes the state carry

Let o_t be the rendered frame, $z_t = e(o_t)$ its latent and s_t the state of the task entity. A k-window predictor imagines a rollout from the last k latents and actions. A sampling planner such as CEM scores each candidate by a function of the rollout and the goal. It returns a distribution that depends on the candidates only through their scores.

Proposition 1 (Window blindness)**.** *If two histories share the goal and the last k observations and actions, with the entity hidden throughout, the k-window planner returns the same action distribution on both. Hence, on episodes in which the entity has been hidden for at least k frames, its success rate is at most that of the best policy whose action at each replanning time is measurable with respect to the goal and the last k frames and actions.*

Each imagined step depends only on the window, earlier imagined latents and candidate actions, so equal windows and goals give every candidate the same rollout and score. A planner that keeps the latent of the last frame showing the entity is bounded likewise, by that frame and the actions since. Its estimate of s_t goes stale when the entity moves in ways those actions do not determine.

For training, input frames are erased after a visible prefix and the targets are the clean next latents $z_{t+1} = e(o_{t+1})$. Write H_t for the erased history and W_t for its last k frames and actions. A model with a state predicts with a function of H_t, and a k-window model with a function of W_t. Both minimize $\mathbb{E}\,\|z_{t+1} - f\|^2$.

Proposition 2 (What erasure training makes the state carry)**.** *The squared error is minimized by $m_H = \mathbb{E}[z_{t+1} \mid H_t]$ over functions of H_t and by $m_W = \mathbb{E}[z_{t+1} \mid W_t]$ over functions of W_t, and*

$$\mathbb{E}\,\|z_{t+1} - m_W\|^2 - \mathbb{E}\,\|z_{t+1} - m_H\|^2 = \mathbb{E}\,\|m_H - m_W\|^2 = \mathbb{E}\big[\,\mathrm{tr}\,\mathrm{Var}(m_H \mid W_t)\big] \geq 0.$$

The gap is zero (i) when the entity is visible in every input frame, so that s_t is a function of $\tilde{o}_t$ and z_{t+1} depends on the past only through $(s_t, \tilde{o}_t, a_t)$, and (ii) when the targets are erased too and the rest of the scene evolves independently of the entity.

The proof is the orthogonality of the residual $z_{t+1} - m_H$ to $m_H - f$ for any function f of W_t, and the tower property. With the entity erased throughout the window, the gap is the expected variance of m_H under the posterior of s_t given the window. It is positive exactly when the window leaves s_t uncertain among states that give different clean next latents. Case (i) is why standard training leaves the state empty. Case (ii) is why the targets must stay clean. The propositions establish necessity, not sufficiency. Whether a trained state recovers enough of s_t, and whether the planner uses it, are the empirical questions of Section 2.

M Limitations in full

Summary of limits. MemJEPA does not yet hold over long hidden spans on PushT, where success decays as more frames are hidden and counterfactual training does not stop it (Appendix J). It does not transfer to occluders unseen in training, such as a moving patch (Appendix K), and on DINO-WM's encoder it costs in-view success (Appendix F). Released planners without memory fail with the entity hidden; only recurrent behavior cloning trained with the same erasure is competitive, winning on TwoRoom (Appendix I).

Erasure is idealized. Our experiments use three simulated tasks with a frozen encoder. Erasure is idealized and simulator-privileged. The simulator renders the frame without the entity, the limit of segmentation plus inpainting with no occluder present; on real data a segmenter and inpainter would have to supply it. The entity is also known in advance. At training and at test every condition erases the entity we chose, so which entity the state must carry is given, not learned.

The hidden entity is the controlled body. On a real robot, proprioception and forward kinematics would report the arm, pusher or agent we erase, and a no-memory model given that state matches MemJEPA on Reacher (Appendix K). Erasing the controlled body is therefore a controlled proxy

for hidden state, not a realistic condition. The robotics case it points to is a hidden object moved by contact. We designed that test (PushT with the block erased, the pusher visible, and contact while the block is hidden) but have not run it.

An off-the-shelf occluder does not replace erasure. Applied at test time with no training, the moving-patch occluder of stable-worldmodel [28] collapses every planner on PushT (development set, 300 episodes each). Without the patch, the GRU belief, MemJEPA and the no-memory model score 65.0, 69.3 and 45.0%. At the patch's default size, which covers the pusher on about 1% of steps, they score 3.7, 2.7 and 3.3%; with sides of 0.4 to 0.5 of the frame, which cover it on 24% of steps, they score 0.0, 0.3 and 0.0%.

Imperfect erasure at test time (never-opened sets; TwoRoom on training-split episodes). An inpainted fill of the exact mask, false-positive masks and an HSV color segmenter leave PushT within about 2 points of exact erasure. On TwoRoom they cost the GRU belief 8 to 10 points and MemJEPA 3, while the twin also rises (35.7 to 47–52%), so part of the agent leaks through the fill. On Reacher, where the trained erasure is one exact image, they cost B 45 to 47 points and S 18 to 20 (94.3 to 48.2 and 94.8 to 76.3 with the inpainted fill). Eroding the mask by one pixel leaks the entity (the Reacher twin rises from 8.7 to 35.7%), so those evaluations do not test memory. Under erosion the GRU belief and MemJEPA lose 54.3 and 28.8 points on Reacher, gain 9.5 and 6.5 on PushT, where the leak helps every variant and the twin most, and change by $+1.8$ and -5.2 on TwoRoom. Temporally inconsistent masks serve each erased frame un-erased with probability 0.1, with the same draw for every variant. They change hidden success by -3.7 and -0.2 points for the GRU belief and MemJEPA on Reacher, $+1.7$ and $+2.2$ on PushT and $+2.0$ and 0.0 on TwoRoom (paired differences against exact erasure on the same sets). The un-erased frames also lift the twin (Reacher 8.7 to 33.3%, TwoRoom 35.7 to 54.0%). The state bridges the erasure it was trained on. Whether natural occlusion in the training data trains the state is untested.

Branches, long spans and demonstrations. Counterfactual branches need a simulator that can be reset to a recorded state, and they raised success on one task of three. Memory decays with the hidden span: on PushT S's hidden success falls from 61.7% at 2 hidden frames to 10.2% at 40. Where erasure-trained memory fails under long occlusion on PushT, counterfactual branch training does not repair it ($+4.8\,[+1.5,\ +8.2]$; Appendix J). With near-expert demonstrations an erasure-trained recurrent BC beats MemJEPA and the GRU belief with the entity hidden on TwoRoom (Appendix I). We therefore make no claim that the planner is the practical route when such demonstrations exist. Reward-free or non-expert data, where a planner could still earn its place, is untested here.

Memory. Memory clearly beats every memoryless alternative only on PushT, where the hidden pusher drives a visible block through contact. On Reacher a memoryless predictor rolled open-loop through the hidden span with the executed actions matches MemJEPA and the GRU belief, and on TwoRoom it matches the GRU belief while MemJEPA leads by $+6.7\,[+2.8,\ +10.7]$, so the open-loop rollout accounts for most of the memory variants' gain there (Appendix K). On a region-occluded TwoRoom set the belief does not recover when the entity reappears, on a set where 66 of 71 cases never show the agent before the first decision, so no earlier observation exists to remember. That failure measures planning from a never-seen start and is not attributed to memory.

Context windows and uncertainty. Context windows catch up past the training clip. A 40-frame window beats the GRU belief (B_0) at 33 hidden Reacher frames (45.5 against 24.8%), though not MemJEPA, which scores 81.2% on the same cases (Appendix D), and on TwoRoom the post-hoc 10-frame window matches B (78.0 against 79.0%). A stochastic state trained with the energy score did not help planning (Section 2). A state whose uncertainty the planner can act on, tested on real robots under self-occlusion, remains open.

Backbone and scale. On DINO-WM's backbone MemJEPA fails the in-view criterion, losing 15.2 to 32.3 points. The erasure schedule explains 5.5 of the 32.3 on PushT; the rest is unexplained (Appendix F). The primary contrasts rest on three training seeds per variant, a frozen encoder and three simulated tasks.

Settings that failed a precondition. Three established settings failed a precondition before any memory test could run. A memory contrast needs a base latent that encodes the controlled entity, a base planner that matches the released planner in view, and planning that succeeds in view. Memory can repair missing history, but not an unusable representation or an incapable planner. On Searing Spotlights from Memory Gym [29], the frozen encoder does not encode the agent (a position probe errs by 23.9 px against a 6 px criterion). On DINO-WM's Wall our base planner reached 23.3% against the released planner's 96%. On robomimic [30] Lift and Can the behavior-cloning policy used to initialize CEM beat the planner on its own, and plain CEM scored 0 of 50 on both, so no memory contrast could be measured. CLWM's Occluded Push [19] is unreleased. FIRM-WM [16] is not released as a planner over a frozen encoder; ported, it becomes a CEM planner with a recurrent state on LeWM latents. The standard-training RSSM and the clean-clip belief cover that port, except its intervention branches (the ancestor of our counterfactual branches) and its label-supervised configuration. Appendix N compares both with MemJEPA.

N Related work

DINO-WM plans with CEM on frozen DINOv2 features [9]. LeWM trains the encoder and predictor end to end with LeJEPA's SIGReg [10, 31]. LEAP changes the planner [13]. All of these work under full observability. PlaNet and Dreamer advance a recurrent state in imagination [15, 24], and S4WM and R2I use state-space memory [32, 27], but none uses a frozen encoder. CLWM and FIRM-WM condition the predictor on a belief [19, 16]. CLWM hides a pushable object and trains its world model with a contrastive counterfactual objective over imagined rollouts rather than erasure; FIRM-WM evaluates without occlusion. Under occlusion, ELVIS plans in a recurrent state-space model [6], μVLA carries a recurrent memory inside a vision-language-action policy [3], and object permanence has been learned through latent imagination during occlusion [5] and handled by tracking-based memory [4]. GateL0RD plans through a wall occlusion with recurrent states trained on partially observed sequences [17]. VRKN trains a recurrent state-space model on occluded inputs [18], and recent work trains one on randomly masked pixels for CEM control, evaluated unmasked [33]. Observational dropout [34], feature masking [35] and object-slot masking in Causal-JEPA [36] shape representations or robustness. Loci-Looped trains through frame blackouts against the real next frame [5]. DRQN learns a recurrent Q-network rather than a world model [2]. None of these combines clean-target memory training for a pretrained predictor, erasure of the controlled entity itself, and a recipe-matched no-memory model.