TRAJ-LEWM: PATH-AWARE WORLD-MODEL PLANNING VIA LATENT TRAJECTORY COST

Anonymous authors
Paper under double-blind review

ABSTRACT

LeWM is a lightweight visual world model that learns latent dynamics end-to-end from pixels and ranks candidate action sequences by the distance between their predicted endpoints and the goal. However, LeWM has two limitations. First, during training, it learns local next-step transitions without evaluating complete trajectories relative to the task goal. Second, during planning, it ranks candidates solely by predicted endpoint distance. Because model predictions may differ from actual execution outcomes, the candidate whose predicted endpoint is closest to the goal may not perform best when executed in the environment. The evolution of the complete predicted trajectory can therefore provide complementary information beyond endpoint distance. To address these limitations, we propose Traj-LeWM, which retains LeWM's local-dynamics objective and endpoint score while introducing a goal-conditioned latent trajectory cost (LTC) that aggregates trajectory-level information as a complementary signal. During training, LTC-based trajectory-preference supervision complements next-step prediction in shaping the shared representation. During planning, LTC is combined with endpoint distance to incorporate intermediate-path information into candidate ranking. With joint endpoint-plus-LTC scoring, Traj-LeWM outperforms LeWM on Push-T, OGBench-Cube, Reacher, and Two-Room by 3, 14, 7, and 7 percentage points, respectively. Controlled experiments and ablations further verify the complementary roles of trajectory-level representation shaping and path-aware candidate ranking.

1 INTRODUCTION

Joint-embedding predictive architectures (JEPAs) support model-based control by predicting future observations in compact latent spaces rather than reconstructing decision-irrelevant pixel details (LeCun, 2022; Assran et al., 2023). LeWM (Maes et al., 2026) extends this paradigm into a lightweight visual world model trained end-to-end from pixels and achieves strong performance in goal-conditioned planning. However, this performance does not remain consistent on more challenging tasks: LeWM's success rate drops from 96% on Push-T to 74% on OGBench-Cube, whereas DINO-WM reports 86% success on Cube (Zhou et al., 2024). Across tasks, we empirically find that candidates with the same start and goal and similar endpoint costs can produce different execution outcomes, showing that scalar endpoint distance does not fully determine execution outcomes. This motivates using complete-trajectory information in both representation learning and candidate ranking.

Two properties of LeWM motivate trajectory-level supervision and scoring. First, next-step prediction supervises local transitions, while SIGReg regularizes the latent distribution; neither directly supervises the shared encoder on complete trajectories relative to a task goal. Second, endpoint distance cannot distinguish candidates by their intermediate evolution. A predicted endpoint close to the goal does not guarantee successful execution, motivating a complementary trajectory-level signal.

We propose Traj-LeWM, which retains LeWM's prediction objective and endpoint score while adding a goal-conditioned latent trajectory cost (LTC) (Fig. 1). LTC maps goal-relative latent evolution to a scalar cost. Pairwise preferences favor goal-matched expert trajectories over synthetic negatives and closed-loop execution failures; synthetic preferences also shape the shared encoder.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

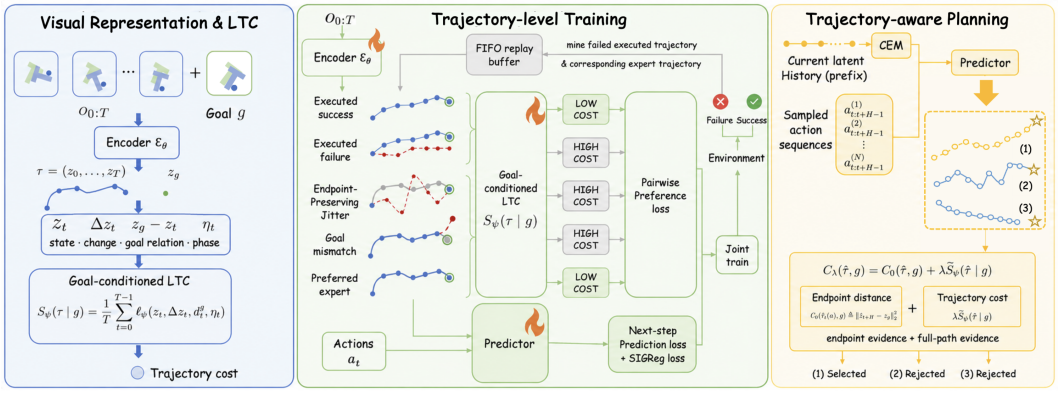

Figure 1: Overview of Traj-LeWM. Stage 1 encodes the observation trajectory and goal into latent representations used by LTC. Stage 2 trains LTC with synthetic and mined closed-loop trajectory preferences while retaining LeWM's next-step prediction objective. Stage 3 ranks CEM candidates using the joint endpoint-plus-LTC score.

During planning, LTC supplements endpoint distance, allowing candidate ranking to use both goal proximity and the evolution of the predicted trajectory. Traj-LeWM outperforms LeWM on four simulated tasks and shows feasibility on a physical Franka FR3. Controlled experiments and ablations examine the effects of training supervision and planning-time scoring on representations and candidate rankings.

In summary, our contributions are threefold:

- We introduce a learned goal-conditioned latent trajectory cost that evaluates complete predicted trajectories. By aggregating goal-relative latent evolution over time, LTC complements endpoint distance with a path-sensitive planning signal.
- We develop a trajectory-preference learning mechanism based on synthetic negative trajectories and closed-loop execution failures. Synthetic preferences introduce trajectory-level, goal-conditioned supervision into the shared representation, while mined failures further train LTC without changing LeWM's original predictor objective.
- We conduct a comprehensive evaluation on four simulated tasks and a physical Franka FR3. Compared with LeWM, Traj-LeWM improves success rates by 3, 14, 7, and 7 percentage points on Push-T, OGBench-Cube, Reacher, and Two-Room, respectively, and increases real-robot success from 50% to 70% over the same 20 tasks. Controlled diagnostics and ablations characterize the distinct effects of training supervision and planning-time scoring.

2 Related Work

World models and JEPA-based planning. World models support decision-making by learning predictive dynamics and planning through imagined rollouts. Representative approaches include generative latent agents such as Dreamer (Hafner et al., 2020; 2023), value-guided planners such as TD-MPC (Hansen et al., 2022; 2024), and reward-free latent planners such as PLDM and DINO-WM (Sobal et al., 2025; Zhou et al., 2024). Universal Planning Networks jointly learn representations and differentiable latent-space planning for image-conditioned control (Srinivas et al., 2018). Joint-embedding predictive architectures (JEPAs) instead predict masked or future content directly in representation space (Assran et al., 2023; LeCun, 2022). Recent systems extend this principle to action-conditioned physical planning: V-JEPA 2-AC plans from image goals on robots (Assran et al., 2025), while a systematic study of JEPA world models analyzes how representation, prediction, and planner design affect physical planning (Terver et al., 2026). LeWM (Maes et al., 2026) learns its encoder and predictor end-to-end from pixels, then ranks action sequences by predicted endpoint distance. Value-guided JEPA planning further shapes embedding distances to approximate a goal-conditioned value function (Destrade et al., 2026). Our method instead learns a cost over complete goal-conditioned latent rollouts and uses it alongside, rather than as, the endpoint distance.

Trajectory-level planning signals. Planning signals beyond endpoint distance include goal-conditioned value functions and reachability estimates (Eysenbach et al., 2019; Kaelbling, 1993), as well as visual representations trained to provide dense goal-conditioned rewards (Ma et al., 2023). Learned reward predictions support latent imagination in agents such as DreamerV3 (Hafner et al., 2023), whereas optimal cost design learns surrogate objectives tailored to the finite horizon and re-planning behavior of model predictive control (Jain et al., 2021). A separate line treats trajectories as generated objects: denoising diffusion models provide the underlying generative machinery (Ho et al., 2020), Diffuser adapts it to classifier-guided trajectory planning (Janner et al., 2022), and energy-based models define scalar landscapes over structured inputs (Du & Mordatch, 2019). Goal-conditioned behavior cloning and offline value-learning methods, including GCBC, GCIVL, and GCIQL, instead learn policies or values directly from offline data (Park et al., 2025; Kostrikov et al., 2022). LTC does not replace LeWM's dynamics model or candidate generator. It scores the same CEM candidates from their complete predicted latent paths and supplements the endpoint score during ranking.

Preference-based cost learning. Pairwise preference models learn scalar rankings from comparisons, with the Bradley–Terry model providing the standard logistic form (Bradley & Terry, 1952); the same principle underlies reward learning from human comparisons (Christiano et al., 2017). At the trajectory level, T-REX learns a reward from ranked demonstrations (Brown et al., 2019), and D-REX automatically constructs rankings by perturbing a behavior-cloned policy to generate demonstrations of different quality (Brown et al., 2020). Energy-based models likewise represent relative plausibility through a learned scalar landscape (LeCun et al., 2006; Du & Mordatch, 2019). We use the standard pairwise objective rather than proposing a new preference loss. Our preferences act on complete goal-conditioned latent trajectories: negatives combine goal mismatches and endpoint-preserving perturbations with failures mined from closed-loop execution, and the learned cost both shapes the shared encoder during training and ranks predicted trajectories during planning.

3 METHOD

Overview. Traj-LeWM extends LeWM with a goal-conditioned latent trajectory cost (LTC), $S_\psi(\tau \mid g)$, that evaluates complete latent trajectories. As illustrated in Fig. 1, trajectory-level preferences train LTC and shape the shared representation, while during planning LTC supplements endpoint distance with a path-sensitive signal for ranking candidate action sequences.

3.1 MOTIVATION: TWO GAPS IN LEWM

LeWM training and planning. The agent receives pixel observations $o_t \in \mathcal{O}$ and executes actions $a_t \in \mathcal{A}$. Following LeWM (Maes et al., 2026), an encoder $\mathcal{E}_\theta : \mathcal{O} \to \mathbb{R}^d$ maps each observation to a latent state $z_t = \mathcal{E}_\theta(o_t)$, and an action-conditioned predictor $\mathcal{F}_\phi$ predicts the next latent state. We write the goal observation as g and its representation as $z_g = \mathcal{E}_\theta(g)$. LeWM trains the encoder and predictor with a next-step latent prediction loss $\mathcal{L}_{\text{pred}}$ and uses SIGReg, denoted by $\mathcal{R}_{\text{sigreg}}$, to prevent representation collapse:

$$\mathcal{L}_{\text{LeWM}} \triangleq \mathcal{L}_{\text{pred}} + \lambda_{\text{sig}} \mathcal{R}_{\text{sigreg}}. \tag{1}$$

At inference time, LeWM freezes the world model and autoregressively rolls out each candidate action sequence $a_{t:t+H-1}$. With a real-prefix length L, the context-augmented predicted latent trajectory is

$$\hat{\tau}_t(a) = \big(z_{t-L+1:t}, \hat{z}_{t+1:t+H}\big). \tag{2}$$

LeWM scores a candidate solely by the distance between its predicted endpoint and the goal representation:

$$C_0(\hat{\tau}_t(a), g) \triangleq \big\|\hat{z}_{t+H} - z_g\big\|_2^2. \tag{3}$$

It then uses CEM to minimize Eq. (3) and executes the selected action sequence in a receding-horizon manner. We omit the current-time subscript below and write the predicted trajectory as $\hat{\tau}$.

Gap 1: Missing trajectory-level, goal-conditioned supervision. The prediction loss constrains local transitions, whereas SIGReg shapes the distribution of latent representations. Neither term in Eq. (1) jointly uses an ordered complete trajectory and its task goal. The original objective therefore provides no explicit trajectory-level, goal-conditioned supervision to the shared representation.

Gap 2: Endpoint scoring is insensitive to intermediate paths. Eq. (3) reads only the predicted endpoint. For any two intermediate latent-state sequences $u_{1:H-1}$ and $v_{1:H-1}$ that share an initial state z_0 and endpoint z_H,

$$C_0\big((z_0, u_{1:H-1}, z_H), g\big) = C_0\big((z_0, v_{1:H-1}, z_H), g\big). \tag{4}$$

The endpoint term therefore cannot directly use information about how a candidate reaches its endpoint. Intermediate dynamics may nevertheless provide additional evidence about candidate quality under the specified goal when endpoint distance alone is insufficient. Together, these gaps leave trajectory-level, goal-conditioned information without an explicit role in either representation learning or candidate scoring.

3.2 Goal-Conditioned Trajectory Cost

Designing a goal-conditioned path functional. To use information along the complete predicted path rather than only its endpoint, we formulate trajectory quality as a goal-conditioned path functional over an ordered sequence of latent states and instantiate it as a latent trajectory cost (LTC). LTC maps a predicted trajectory to a learned scalar cost by aggregating latent states, their temporal changes, and goal-relative information over time. Intermediate-path information therefore contributes directly to trajectory evaluation rather than being discarded by endpoint-only scoring. The ranking semantics of this cost are learned from trajectory preferences: goal-matched expert trajectories should receive lower costs than synthetic negative trajectories and execution failures.

We implement this functional using four components: **(i) latent state and local evolution**, represented by the current state z_t and the first difference $\Delta z_t = z_{t+1} - z_t$ between adjacent latent states; **(ii) goal conditioning**, introduced at every step through the goal-relative quantity $d_t^g = z_g - z_t$; **(iii) temporal position**, represented by the normalized phase $\eta_t = t/\max(T-1, 1)$; and **(iv) learned trajectory aggregation**, in which ℓ_ψ maps the information at each step to a nonnegative contribution and these contributions are averaged over the trajectory. The goal-relative term allows the same latent state and change to receive different evaluations under different goals, while the phase term distinguishes where an evolution occurs along the trajectory. Averaging over time produces a length-normalized cost for the complete trajectory.

Definition 1 (Goal-Conditioned Latent Trajectory Cost)**.** *Given a latent trajectory $\tau = (z_0, \dots, z_T)$ with $T \geq 1$ and goal representation z_g, define*

$$S_\psi(\tau \mid g) = \frac{1}{T} \sum_{t=0}^{T-1} \ell_\psi(z_t, \Delta z_t, d_t^g, \eta_t)\,. \tag{5}$$

Here $\ell_\psi : \mathbb{R}^{3d+1} \to \mathbb{R}_{\geq 0}$ is a learnable nonnegative per-step contribution function, whose aggregation over time defines the cost of the complete trajectory.

Under the learned trajectory preferences, a lower S_ψ indicates a trajectory preferred as goal-matched expert behavior, whereas a higher value indicates a less preferred trajectory, such as a synthetic negative or an execution failure. Unlike C_0, which reads only the endpoint, LTC is structurally able to use intermediate latent states, their changes, and their relations to the goal.

3.3 Trajectory Preference Supervision

Write a goal-conditioned example as $x = (\tau, g)$ and abbreviate $S_\psi(x) = S_\psi(\tau \mid g)$. For a preference pair (x^+, x^-), define the cost margin as $m_\psi(x^+, x^-) = S_\psi(x^-) - S_\psi(x^+)$ and use the pairwise logistic loss

$$\ell_\beta(x^+, x^-) = \log\left(1 + \exp\left[-\frac{m_\psi(x^+, x^-)}{\beta}\right]\right), \tag{6}$$

where $\beta > 0$ is a temperature. Minimizing Eq. (6) encourages $S_\psi(x^+) < S_\psi(x^-)$, so that a goal-matched expert trajectory is preferred to its corresponding negative.

Synthetic preferences. For an expert trajectory $\tau^+ = (z_0^+, \dots, z_T^+)$, let the goal representation be $z_g = z_T^+$ and define the positive example $x^+ = (\tau^+, g)$. The first negative replaces the goal with a batchwise cyclically shifted goal g':

$$x^-_{\mathrm{gm}} = (\tau^+, g'). \tag{7}$$

The second negative adds noise to the intermediate states. Here s_{batch} is the scalar latent standard deviation over the current batch and time dimensions, and $\sigma = 0.05$ is the jitter scale. Draw $\epsilon_t \sim \mathcal{N}(0, \sigma^2 s_{\text{batch}}^2 I)$ and fix $\epsilon_0 = \epsilon_T = 0$; the perturbed states are

$$z_t^- = z_t^+ + \epsilon_t, \qquad 0 \le t \le T. \tag{8}$$

Let $\tau_{\text{jit}}^- = (z_0^-, \dots, z_T^-)$ and $x_{\text{jit}}^- = (\tau_{\text{jit}}^-, g)$. Goal mismatch trains LTC to assign an expert trajectory a lower cost with its corresponding goal than with an unrelated goal. Endpoint-preserving perturbations train LTC to distinguish different intermediate paths even when the initial and final states are fixed.

We stop gradients through the synthetic negative branch while retaining the current encoder's computation graph for the expert branch. Synthetic preferences therefore update LTC directly and update the encoder through the expert branch; their effect on negative encodings is mediated indirectly by the cost landscape learned by LTC. The corresponding loss is

$$\mathcal{L}_{\text{path}} = \mathbb{E}\left[\frac{w_{\text{gm}}\ell_\beta(x^+, x_{\text{gm}}^-) + w_{\text{jit}}\ell_\beta(x^+, x_{\text{jit}}^-)}{w_{\text{gm}} + w_{\text{jit}}}\right]. \tag{9}$$

Closed-loop failure preferences. Synthetic negatives expose predefined structural differences but cannot cover errors produced by the model during actual planning. After each training epoch, we therefore use the current model to perform endpoint-only CEM planning and execute the resulting plan in a resettable training environment. We deliberately omit LTC during failure mining so that it does not filter out failures selected by endpoint-only scoring; these episodes provide negative examples of planning errors that the endpoint term alone cannot identify. The endpoint-only planner is used only for collecting training failures, whereas final planning uses the joint endpoint-plus-LTC score defined below. The environment's success criterion supplies a binary episode-level label. For each failed episode i, we encode the executed observations frame by frame as $\tau_i^- = (\mathcal{E}_\theta(o_{i,0}^{\text{exec}}), \dots, \mathcal{E}_\theta(o_{i,T_i^-}^{\text{exec}}))$ and pair it with an expert trajectory τ_i^+ having the same initial condition and goal:

$$(x_i^+, x_i^-) = \big((\tau_i^+, g_i), (\tau_i^-, g_i)\big). \tag{10}$$

Positive and negative trajectories may differ in length because Eq. (5) averages each trajectory over its own number of transitions. We insert these preference pairs sequentially into a bounded FIFO buffer $\mathcal{B}$. The corresponding buffer loss is

$$\mathcal{L}_{\text{mined}} = \mathbb{E}_{(x^+, x^-)\sim\mathcal{B}}\ell_\beta(x^+, x^-). \tag{11}$$

The trajectory and goal representations inserted into the buffer are detached, so $\mathcal{L}_{\text{mined}}$ updates only LTC in the current backward pass. It first changes LTC parameters and then indirectly influences the encoder through $\mathcal{L}_{\text{path}}$ in subsequent batches.

We intentionally exclude the mined observations from predictor training, because using them as transition targets would introduce online dynamics adaptation and confound it with the effect of LTC. Closed-loop mining is instead an integral part of the proposed trajectory-preference module: it supplies failure preferences only for calibrating LTC, while LeWM's predictor data and objective remain unchanged.

3.4 Joint Training and Gradient Flow

Writing the LeWM objective defined in Eq. (1) as $\mathcal{L}_{\text{LeWM}}$, the complete training objective is

$$\mathcal{L}_{\text{Traj-LeWM}} = \mathcal{L}_{\text{LeWM}} + \lambda_{\text{path}}\mathcal{L}_{\text{path}} + \lambda_{\text{mined}}\mathcal{L}_{\text{mined}}. \tag{12}$$

Table 1 summarizes the direct gradient flow from each loss in one backward pass. This design preserves the predictor's training signal while allowing trajectory preferences to shape the shared encoder through the synthetic expert branch. Closed-loop failure preferences primarily calibrate LTC against the model's own failure modes, while the synthetic expert branch supplies trajectory-level supervision to the encoder.

Each epoch has two stages. First, we jointly optimize Eq. (12) on offline expert batches: we construct goal-mismatched and endpoint-preserving perturbations to compute $\mathcal{L}_{\text{path}}$ and, when the buffer is nonempty, sample closed-loop preferences to compute $\mathcal{L}_{\text{mined}}$. We then disable gradients, perform closed-loop planning with the current model, and add newly discovered failed trajectories to $\mathcal{B}$. Training alternates between these two stages.

Loss	Encoder $\mathcal{E}_\theta$	Predictor $\mathcal{F}_\phi$	LTC S_ψ
$\mathcal{L}_{\text{pred}}$	✓	✓	
$\mathcal{R}_{\text{sigreg}}$	✓		
$\mathcal{L}_{\text{path}}$	✓		✓
$\mathcal{L}_{\text{mined}}$			✓

Table 1: Direct gradient flow from each loss term.

3.5 Trajectory-Sensitive Candidate Scoring

During planning, LTC reads, as defined in Eq. (2), the context-augmented predicted latent trajectory $\hat{\tau}$. To eliminate the scale difference between LTC output and endpoint distance, we first define the calibrated LTC score

$$\widetilde{S}_\psi(\hat{\tau} \mid g) = \frac{\mathrm{IQR}(C_0)}{\max(\mathrm{IQR}(S_\psi), \epsilon)} S_\psi(\hat{\tau} \mid g). \tag{13}$$

Here $\epsilon > 0$ is a small constant for numerical stability. $\mathrm{IQR}(x) = Q_{0.75}(x) - Q_{0.25}(x)$ denotes the interquartile range; the interquartile ranges of C_0 and S_ψ are both estimated over the same set of candidates produced by endpoint-only CEM. The combined score is defined as

$$C_\lambda(\hat{\tau}, g) = C_0(\hat{\tau}, g) + \lambda \widetilde{S}_\psi(\hat{\tau} \mid g), \tag{14}$$

where C_0 evaluates only the predicted endpoint, $\widetilde{S}_\psi$ reads the complete predicted latent trajectory, and $\lambda \geq 0$ controls the relative strength of the LTC signal. When $\lambda = 0$, candidates are ranked solely by the endpoint term, although the encoder may still have been shaped by trajectory preference training.

Thus, the combined score retains endpoint-based goal matching while incorporating information from complete predicted trajectories into candidate selection.

4 Experiments

We first evaluate closed-loop planning across four simulated environments. Controlled experiments compare scoring functions under matched supervision and examine representation shaping and candidate ranking. Ablations separate trajectory supervision during training from LTC scoring during planning. Finally, we evaluate both methods on a physical robot.

4.1 Main Results: Closed-Loop Planning in Simulation

Experimental setup. We evaluate closed-loop planning using the public LeWM datasets (Maes et al., 2026) for Push-T (Chi et al., 2025; Zhou et al., 2024), OGBench-Cube (Park et al., 2025), Reacher (Tassa et al., 2018), and Two-Room (Sobal et al., 2025), covering object manipulation, continuous control, and obstacle navigation. LeWM (Maes et al., 2026) is our primary controlled baseline. LeWM and Traj-LeWM use the same offline dataset, encoder–predictor backbone, predictor objective, optimization schedule, MPC framework, and CEM sampling budget. Their intended planning-time difference is the candidate-scoring objective: LeWM uses endpoint distance, whereas Traj-LeWM combines endpoint distance with LTC.

Traj-LeWM additionally runs 200 closed-loop training queries after each epoch and uses success/failure labels to construct preferences. The resulting observations are excluded from predictor transition training, but this still adds interaction and supervision unavailable to LeWM. The comparison therefore evaluates the complete training-and-planning system. Section 4.2 separately compares learned endpoint and trajectory scorers with matched capacity and supervision on frozen backbones.

Following LeWM, we report task success rates under a shared evaluation protocol. For broader comparison, Table 2 also includes PLDM (Sobal et al., 2025), DINO-WM (Zhou et al., 2024), and the goal-conditioned methods GCBC, GCIVL, and GCIQL (Park et al., 2025; Kostrikov et al., 2022). Additional implementation and evaluation details, including the model architecture, training, failure mining, planning settings, and multi-seed evaluation counts, are provided in the appendix.

Model	Push-T	Cube	Reacher	Two-Room
GCBC	75	84	–	**100**
GCIVL	33	56	–	**100**
GCIQL	20	64	–	**100**
PLDM	78	65	78	97
DINO-WM	74	86	79	**100**
LeWM	96	74	86	87
Traj-LeWM (Ours)	**99**	**88**	**93**	94

Table 2: Closed-loop planning success rates (%) across four environments. For Traj-LeWM, results are averaged over three evaluation seeds and rounded to the nearest percentage point. Per-seed counts and evaluation-seed standard deviations appear in Appendix B.1; these do not quantify training-seed variation. Bold indicates the best result for each task.

Main results. With joint scoring, Traj-LeWM achieves 99%, 88%, 93%, and 94% success on Push-T, Cube, Reacher, and Two-Room: gains of 3, 14, 7, and 7 percentage points over LeWM. It leads the reported comparisons on the first three tasks; Two-Room remains below alternatives at 100%.

LTC adds fewer than 1M parameters to the lightweight backbone. These results measure the combined effect of additional supervision and the learned planning objective.

4.2 Controlled Tests of the Two Gaps

Capacity- and supervision-matched scorer controls. We compare LTC with a learned endpoint scorer while holding the world-model backbone, scoring capacity, and preference supervision fixed. The encoder and predictor are frozen, and the endpoint scorer receives the full start, terminal, and goal latent vectors. Both scorers contain 824,833 parameters and use the same synthetic preferences and shared failure examples. Each scorer is combined with endpoint distance using a validation-selected weight, with identical CEM budgets during evaluation. This experiment uses a separate data split and calibration protocol from Table 2; details and the full six-scheme comparison appear in Appendix A.3.

Task	Endpoint	LTC	Δ (pp)	Paired 95% CI (pp)
Push-T	87.66	89.53	$+1.88$	$[-1.25, +5.00]$
Cube	66.41	72.66	$+6.25$	$[+1.56, +10.94]$
Reacher	59.69	65.94	$+6.25$	$[+1.56, +10.94]$
Two-Room	94.06	98.13	$+4.06$	$[+0.63, +7.50]$

Table 3: Frozen-backbone scorer controls. Success rates (%) aggregate 640 executions per scheme: five training seeds $\times$ 128 shared test queries per task. Validation-selected weights remain fixed during testing. Differences and paired 95% intervals are in percentage points (pp).

LTC improves success on all four tasks; paired 95% intervals exclude zero on Cube, Reacher, and Two-Room, but include zero on Push-T. These controls use frozen representations; the following diagnostics examine jointly trained models.

Testing Gap 1: representation shaping. We execute shared candidate action sequences to obtain success labels, then score them using endpoint distance in each model's predicted representation, with LTC excluded from planning. The difference in success-discrimination AUC measures the effect of trajectory-preference training on endpoint ranking.

Compared with LeWM, Traj-LeWM improves the endpoint-score AUC by 0.4, 1.0, and 4.3 percentage points on Push-T, Cube, and Two-Room, respectively, with statistically significant gains on Cube and Two-Room. The change on Reacher is a nonsignificant -0.4 percentage points (Fig. 2(a)). In Fig. 3, Traj-LeWM also attends to the partition and doorway, whereas LeWM largely follows the agent. This qualitative view complements the endpoint-ranking results (Appendix B.5).

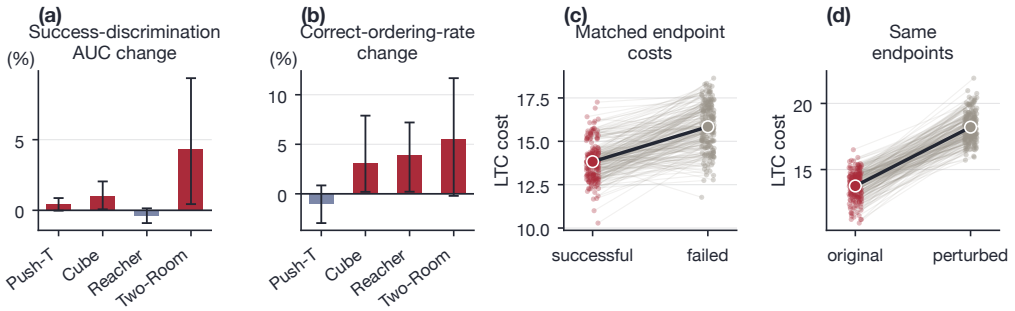

Figure 2: Representation and planning diagnostics. **(a)** Endpoint-only AUC change from LeWM to Traj-LeWM. **(b)** Correct-ordering change after adding LTC for candidates with large prediction–execution discrepancies. **(c)** On Reacher, query-averaged ordering accuracy is 94.4% across 207 endpoint-matched success–failure pairs. **(d)** Intermediate-state perturbations increase LTC cost in all 200 Reacher pairs with fixed endpoints. Bars in (a,b) are query means with paired-query bootstrap 95% confidence intervals (20,000 resamples); lines in (c,d) connect paired costs.

Testing Gap 2: path use and candidate ranking. We compare endpoint-only and joint scoring on success–failure candidate pairs with large discrepancies between predicted endpoints and encoded executed endpoints. Adding LTC changes the correct-ordering rate by $+3.1$, $+3.8$, and $+5.5$ percentage points on Cube, Reacher, and Two-Room, respectively, and by -1.0 on Push-T (Fig. 2(b)).

Figure 2(c,d) shows endpoint-matching and perturbation diagnostics; Appendix B.2 details their protocols and limits for interpreting valid-path reasoning.

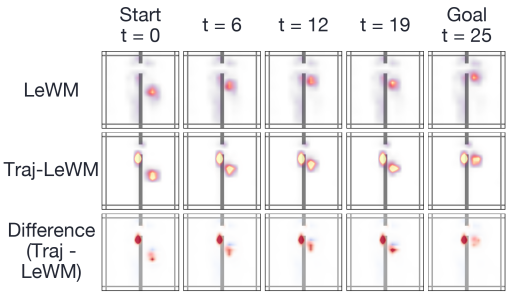

Figure 3: Two-Room CLS-to-patch attention: LeWM, Traj-LeWM, and their difference (top to bottom). Red/blue indicates higher/lower attention under Traj-LeWM.

4.3 Ablation Studies

Table 4 separates training and planning effects: the first three rows vary supervision under endpoint scoring; the last three vary the score for the same trained Traj-LeWM model.

Training	Score	Push-T	Cube	Reacher	Two-Room	Avg.
LeWM	Endpoint	96	74	86	87	85.8
Traj-LeWM-syn	Endpoint	96	79	80	**94**	87.3
Traj-LeWM	Endpoint	98	80	84	**94**	89.0
Traj-LeWM	LTC	78	69	**94**	80	80.3
Traj-LeWM	Joint	**99**	**88**	93	**94**	**93.5**

Table 4: Ablation of training supervision and planning score (success rate, %). Bold indicates the best result for each task.

Training-time supervision. Under endpoint-only scoring, synthetic preferences raise the average success rate from 85.8% to 87.3%, and adding closed-loop failures raises it to 89.0%. Because LTC is absent at test time, these gains reflect trajectory supervision of the shared representation. The fixed endpoint score separates this training effect from the contribution of planning-time LTC ranking.

Effect of planning-time LTC scoring. For the same trained model, adding LTC to endpoint scoring changes success by $+1$, $+8$, $+9$, and 0 points on Push-T, Cube, Reacher, and Two-Room, raising

the average from 89.0% to 93.5%. Thus, the full Two-Room gain and two of the three Push-T points are already present under endpoint-only scoring. LTC-only scoring reaches 94% on Reacher but averages 80.3% across tasks. Joint scoring has the highest average; its benefit varies substantially by task.

4.4 Real-Robot Evaluation

We evaluate goal-conditioned visual planning on a physical Franka FR3 using 20 paired short-horizon start–goal tasks. Both methods receive the same initial observation and a visual goal recorded 1.5 seconds later in the same demonstration. Trajectories are recorded at 30 Hz from three synchronized RGB views. Success requires collision avoidance and a minimum seven-joint L_2 distance to the goal of at most 0.11 rad. This criterion measures closest approach, rather than sustained goal reaching. Figure 4 shows two tasks; Appendix C gives the recording and paired evaluation protocol.

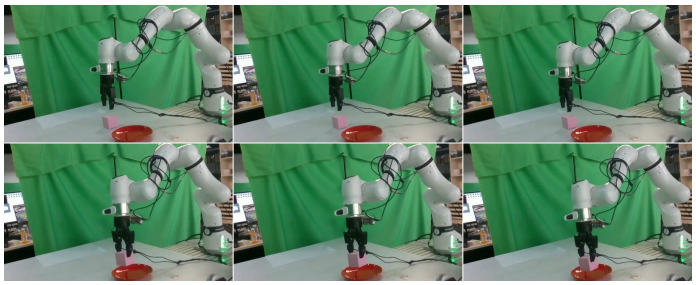

Figure 4: Franka FR3 setup and two tasks. Each row shows the start, an intermediate observation, and the visual goal 1.5 seconds later.

Model	Success	Target miss	Collision
LeWM	10/20 (50%)	6/20 (30%)	4/20 (20%)
Traj-LeWM	14/20 (70%)	4/20 (20%)	2/20 (10%)

Table 5: Real-robot outcomes on the same 20 start–goal tasks. Both methods are evaluated once per task.

As shown in Table 5, Traj-LeWM succeeds in 14 trials compared with 10 for LeWM. Target misses decrease from six to four and collision-induced terminations from four to two. These 20 paired tasks provide preliminary evidence of real-robot feasibility; the sample is insufficient to establish a conclusive performance advantage.

5 Conclusion

Traj-LeWM combines trajectory-preference training with a learned latent trajectory cost for LeWM planning. The method retains local next-step prediction and endpoint-based goal matching while adding trajectory-level supervision and a complementary score over predicted latent trajectories. LTC aggregates latent states, transitions, goal-relative displacements, and temporal position. Synthetic preferences shape the shared encoder, while closed-loop failures further train the cost without adding their observations to predictor transition training. The complete system improves over LeWM on four simulated tasks. Holding the trained model fixed, joint scoring improves most on Cube and Reacher, while the Two-Room gain occurs during training in that experiment. Separate frozen-backbone controls support gains over a capacity- and supervision-matched endpoint scheme on Cube, Reacher, and Two-Room, with an inconclusive Push-T difference. Distinguishing useful intermediate-path information from richer endpoint reasoning remains an experimental question.

AI USE STATEMENT

Generative AI tools assisted result analysis, writing and translation, methodological and experimental discussions, code implementation, and figure preparation. AI-assisted analyses were subsequently checked by the authors. The authors led the research motivation, methodology, and scientific decisions, and take responsibility for the experiments, reported results, and final manuscript.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning (ICML)*, pp. 783–792, 2019.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, pp. 330–359, 2020.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with JEPA world models. *arXiv preprint arXiv:2601.00844*, 2026. URL https://arxiv.org/abs/2601.00844. Presented at the World Modeling Workshop 2026.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Avik Jain, Lawrence Chan, Daniel S. Brown, and Anca D. Dragan. Optimal cost design for model predictive control. In *Learning for Dynamics and Control (L4DC)*, pp. 1205–1217, 2021.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Yann LeCun. A path towards autonomous machine intelligence. *Open Review*, 2022.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 43905–43941. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3e7cf447f21cd11c846463affefce665-Paper-Conference.pdf.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning (ICML)*, pp. 4732–4741, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=cHZn5Gdh8e.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Reproducibility Details

A.1 Model, Training, and Mining

Architecture. The visual encoder is a ViT with 14×14 image patches, hidden dimension 192, 12 Transformer blocks, 3 attention heads, and MLP dimension 768. The autoregressive predictor contains 6 Transformer blocks, 16 attention heads, head dimension 64, MLP dimension 2048, and dropout 0.1. The latent trajectory cost (LTC) takes

$$[z_t,\ z_{t+1} - z_t,\ z_g - z_t,\ t/(T-1)]$$

as input and applies three width-512 hidden layers, each with LayerNorm, GELU, and dropout 0.1, followed by a positive scalar output and temporal averaging. For the retained configurations with a 10-dimensional model action, LeWM has 18,042,672 parameters, while Traj-LeWM-syn and Traj-LeWM each have 18,867,505 parameters; the LTC contributes 824,833 additional parameters. Cube's 25-dimensional model action adds 2,880 action-embedding parameters to each total.

Optimization. We use AdamW with learning rate 5×10^{-5}, weight decay 10^{-3}, batch size 128, bf16 mixed-precision training, gradient clipping at 1.0, and a linear-warmup cosine-annealing scheduler. The trajectory-supervision configuration uses $\lambda_{\text{path}} = \lambda_{\text{mined}} = 0.05$, logistic-loss temperature $\beta = 0.2$, SIGReg weight 0.09, 17 knots, and 1,024 projections. The synthetic goal-mismatch and jitter terms have weights 1.0 and 0.5, respectively; the jitter scale is 0.05 times the scalar latent standard deviation computed over the current batch and time dimensions.

Failure mining. After each training epoch, the current model evaluates 200 start–goal queries using endpoint-only CEM. If a rollout does not reach the environment termination condition within 50 steps, it provides a negative trajectory; the corresponding dataset segment with the same start and goal provides the positive trajectory. Detached latent pairs are stored in a FIFO buffer with capacity 2,048.

A.2 Planning

The jointly trained model evaluation uses MPC with CEM. Each planning round samples 300 action sequences, retains 30 elites, and updates the Gaussian mean and standard deviation from those elites. The initial CEM distribution has zero mean and unit variance in standardized action coordinates. Subsequent means and standard deviations are the empirical elite statistics, with warm starts enabled across replanning rounds.

For joint planning, the retained implementation uses

$$C_{\text{joint}} = C_{\text{endpoint}} + w_{\text{LTC}} C_{\text{LTC}},$$
$$w_{\text{LTC}} = \lambda \frac{\text{IQR}(C_{\text{endpoint}})}{\max\{\text{IQR}(C_{\text{LTC}}), 10^{-12}\}}.$$

The IQRs are estimated from endpoint-only calibration candidates. The final joint-scoring configuration adopted for the paper is $\lambda = (0.5, 0.8, 0.6, 0.5)$ for Push-T, Cube, Reacher, and Two-Room, respectively.

A.3 Details of Matched Scorer Controls

Backbones and data. The experiment in Table 3 freezes a separately trained LeWM backbone for each task and training seed (3072, 4072, 5072, 6072, and 7072). Each backbone uses the architecture in Appendix A.1 and is trained for ten epochs before head training. Episodes are split 80/10/10 into training, validation, and test partitions with split seed 90225; all model seeds share this split. Normalization statistics use the training partition. The two scoring heads share the same frozen encoder and predictor within each seed. These controls use a separate split and calibration procedure from the jointly trained system in Table 2.

Control variants. We evaluate endpoint distance C_0, two endpoint heads, two LTC heads, and a parameter-free smoothness score. For each head architecture, one variant uses synthetic preferences and the other adds the shared failure pool. Synthetic-only training omits the mined loss while

retaining the synthetic data and update budget. The smoothness score is the mean squared second difference of successive predicted latent states, averaged over time and latent dimensions. Every added score uses the same validation-based scale and weight-selection procedure described below.

Scoring functions and capacity. The endpoint head takes $[z_0, z_T - z_0, z_g - z_0, 1]$ as input to a three-hidden-layer, width-512 MLP with LayerNorm, GELU, dropout 0.1, and a positive scalar output. This input exposes the complete start, terminal, and goal vectors but no intermediate states. The constant channel matches LTC's input dimension, giving both heads 824,833 parameters. LTC retains its trajectory aggregation. Changing interior states with endpoints held fixed leaves the endpoint head's input unchanged. Its jitter preference is therefore an exact tie with zero gradient; independent dropout draws are not used to create a spurious difference between identical endpoint inputs.

Shared preferences and optimization. For each backbone, all four heads use the same pool of 20,000 training windows. The shared-failure variants also use failures collected from 2,000 training queries by the frozen endpoint-only planner. Synthetic windows contain six states at offsets $0, 5, 10, 15, 20, 25$. Expert positives span 25 simulator steps and mined failures span 50 steps; both are sampled to six states. Equal sampled length does not make the physical durations equal. Failure collection uses 64 CEM candidates, five iterations, a five-block horizon, and a 50-step execution budget. Each head receives 10,000 AdamW updates with learning rate 5×10^{-5}, weight decay 10^{-3}, 500 warmup steps, cosine decay, and gradient clipping at 1.0. Each update uses 128 synthetic windows; the shared-failure variants additionally use 32 mined pairs. The preference temperature is 0.2; synthetic and mined loss weights are both 0.05, with relative jitter weight 0.5 within the synthetic term. Head training uses bf16 mixed precision, and planning uses float32.

Validation and planning. Each seed uses 64 validation queries and the same 128 test queries, drawn from disjoint episode partitions. For a scoring function S, the planner minimizes $C_0 + \alpha s_S S$. We estimate s_S on validation candidates as the ratio of the median within-query standard deviation of C_0 to that of S, with denominator clamped below at 10^{-12}. Sixteen candidates per query are drawn equally from the first and last iterations of endpoint-only CEM. The scale is fixed per backbone and scorer; it is not recomputed during test-time CEM. The task-level weight is selected from $\{0, 0.1, 1, 10\}$ by validation success, with zero disabling the added score. Two-Room's fifth training seed was evaluated after the first four, retaining their selected weights and calibrating its scale on validation queries. Test planning uses 300 candidates, ten CEM iterations, 30 elites, and five action blocks of five simulator steps each. The joint score determines elite selection at every iteration. Five actions are executed before replanning, within a total budget of 50 simulator steps.

Success criteria and uncertainty. The task success conditions are distance below 16 pixels for Two-Room; joint agent/block position error below 20 pixels and wrapped block-angle error below $\pi/9$ for Push-T; absolute error below 0.05 radians in each Reacher joint; and cube position error at most 0.04 m for Cube. Push-T thus measures state matching rather than polygon coverage. For the shared-failure comparison in Table 3, let $d_{s,q}$ be LTC success minus endpoint-scheme success for training seed s and shared query q. The reported difference is $100 \, \mathrm{mean}_{s,q} \, d_{s,q}$. Paired 95% intervals use 10,000 crossed bootstrap replicates: independently resample training-seed indices and shared-query indices with replacement, retaining the two methods' pairing and using each sampled query index across all sampled seeds. The interval endpoints are the 2.5th and 97.5th percentiles. These are per-task intervals, without a simultaneous four-task adjustment. The 640 executions per scheme include repeated queries across model seeds; they are not 640 distinct start–goal queries.

Complete results. Table 6 reports all six schemes under this protocol. The two shared-failure rows reproduce Table 3. The auxiliary comparisons in this appendix report success counts, rates, and descriptive differences. Paired confidence intervals for the primary shared-failure comparison are reported in Table 3.

Planning score	Head supervision	Push-T	Cube	Reacher	Two-Room
C_0	None	561 (87.66)	434 (67.81)	388 (60.63)	598 (93.44)
C_0+Endpoint	Synthetic	561 (87.66)	428 (66.88)	385 (60.16)	598 (93.44)
C_0+LTC	Synthetic	565 (88.28)	434 (67.81)	408 (63.75)	622 (97.19)
C_0+Endpoint	Synthetic + failures	561 (87.66)	425 (66.41)	382 (59.69)	602 (94.06)
C_0+LTC	Synthetic + failures	573 (89.53)	465 (72.66)	422 (65.94)	628 (98.13)
C_0+Smoothness	None	561 (87.66)	435 (67.97)	386 (60.31)	598 (93.44)

Table 6: Complete frozen-backbone scorer controls. Each cell gives successes out of 640 executions, followed by success rate (%). All schemes use the same five backbones and 128 shared test queries per task, with identical planning budgets. Failures are shared between the two head architectures. The shaded row is the LTC scheme in Table 3.

Success-rate difference (pp)	Push-T	Cube	Reacher	Two-Room
LTC − Endpoint, synthetic	$+0.63$	$+0.94$	$+3.59$	$+3.75$
LTC − Endpoint, synthetic + failures	$+1.88$	$+6.25$	$+6.25$	$+4.06$
Adding failures, endpoint head	0.00	-0.47	-0.47	$+0.63$
Adding failures, LTC head	$+1.25$	$+4.84$	$+2.19$	$+0.94$
Change in LTC–endpoint gap	$+1.25$	$+5.31$	$+2.66$	$+0.31$
Smoothness scheme − C_0	0.00	$+0.16$	-0.31	0.00

Table 7: Descriptive contrasts derived from Table 6. Differences are calculated from success counts before rounding. "Adding failures" compares synthetic-plus-failure with synthetic-only training for the same head architecture. The change in the LTC–endpoint gap subtracts the synthetic-only gap from the shared-failure gap. significance of the auxiliary contrasts.

Scorer capacity and failure supervision. Under synthetic-only supervision, LTC exceeds the endpoint head by 0.63, 0.94, 3.59, and 3.75 percentage points on Push-T, Cube, Reacher, and Two-Room, respectively. Adding shared failures increases LTC success by 1.25, 4.84, 2.19, and 0.94 points. The corresponding endpoint-head changes are 0.00, -0.47, -0.47, and $+0.63$ points. Thus, access to the same failure examples does not give the endpoint scheme the same observed benefit. The LTC–endpoint gap widens on every task in the point estimates, with the largest change on Cube (Table 7). Because each scheme selects its planning weight on validation data, these contrasts describe the trained and calibrated schemes. They do not isolate a head-training effect at a common nonzero weight.

Smoothness and interpretation. The smoothness scheme matches C_0 on Push-T and Two-Room, changes Cube by $+0.16$ points, and changes Reacher by -0.31 points. In contrast, LTC with shared failures exceeds C_0 by 1.88, 4.84, 5.31, and 4.69 points, respectively. The tested smoothness scheme therefore does not reproduce LTC's observed improvement. The capacity- and supervision-matched comparison in Table 3 supports an advantage beyond additional labels or parameter count in this frozen-backbone setting. The auxiliary contrasts describe how that advantage varies with supervision and scoring function. The role of dynamically valid intermediate paths remains unresolved by these controls and the perturbation tests in Appendix B.2.

B Additional Results and Diagnostics

B.1 Verified Multi-Seed Evaluation

Table 8 reports the Traj-LeWM results for evaluation seeds 42–44. The success-count column lists the number of successes out of 50 queries for each seed. Success rates are the mean and sample standard deviation across the three evaluation seeds.

Task	Successes (42/43/44)	Success rate (mean ± SD)
Push-T	49/50/49	98.7 ± 1.2
Cube	44/46/42	88.0 ± 4.0
Reacher	46/47/46	92.7 ± 1.2
Two-Room	47/45/49	94.0 ± 4.0

Table 8: Epoch-10 Traj-LeWM results on 50 start–goal queries per task and evaluation seed. Success rates are mean ± sample standard deviation across evaluation seeds 42–44, in percent.

B.2 Controlled Path-Ranking Diagnostics

Representation and prediction-error diagnostics. Both models receive the same candidate action sequences, whose success labels come from environment execution. We compare predicted endpoint distances in their respective representation spaces with LTC removed from the score. For the prediction-error analysis, Traj-LeWM predicts each candidate's endpoint, and we encode the observation reached by executing the candidate. Within each start–goal query, we compare endpoint-only and joint scoring on successful–failed candidate pairs with large prediction–execution discrepancies. The scoring comparison measures how often the successful candidate receives the lower cost.

Metric definitions. Success rate. Let $s_q = 1$ if query q terminates successfully and $s_q = 0$ otherwise. For N queries, the success rate is the percentage of successful queries:

$$\frac{100}{N} \sum_{q=1}^{N} s_q.$$

Endpoint ROC–AUC. Some diagnostic queries contain multiple successful and failed candidate trajectories. For each such query, candidates are scored by $-C_{\text{endpoint}}$, so a lower endpoint cost gives a higher score. The query-level ROC–AUC measures how often a successful candidate is ranked above a failed candidate; a tie contributes one half. A query containing only successful candidates or only failed candidates is excluded because its ROC–AUC is undefined. We report the arithmetic mean over the remaining queries, giving every query equal weight.

LTC pair-ordering accuracy. The endpoint-matched analysis forms pairs containing one successful and one failed executed trajectory with similar endpoint costs. A pair receives a score of one if LTC assigns lower cost to the successful trajectory, zero if it assigns lower cost to the failed trajectory, and one half for a tie. The reported percentage is 100 times the mean score over all retained pairs.

Endpoint-matched executed trajectories. For each start–goal query, successful executed trajectories are paired one-to-one with unused failed trajectories having the nearest endpoint energy, subject to an endpoint-cost gap of at most 0.25 times the within-query interquartile range. On Reacher, 1,344 candidates yield 395 matched success–failure pairs, of which 207 satisfy this caliper. Table 9 reports the percentage of pairs for which LTC assigns lower cost to the successful trajectory. We omit confidence intervals because the corrected aggregate count is insufficient to reconstruct the previous query-level bootstrap interval.

Task	Queries	Pairs	Correct ordering (%)
Push-T	21	94	85.8
Cube	15	30	90.0
Reacher	22	207	94.4
Two-Room	18	106	83.0

Table 9: LTC ordering accuracy, in percent, for endpoint-matched successful and failed executed trajectories.

Endpoint-preserving perturbations. We fix each trajectory's initial and final latent states and perturb its intermediate states, keeping endpoint costs identical. Across 200 pairs per task, LTC assigns higher cost to the perturbed trajectory in 100% of pairs on Push-T, Cube, and Reacher, and

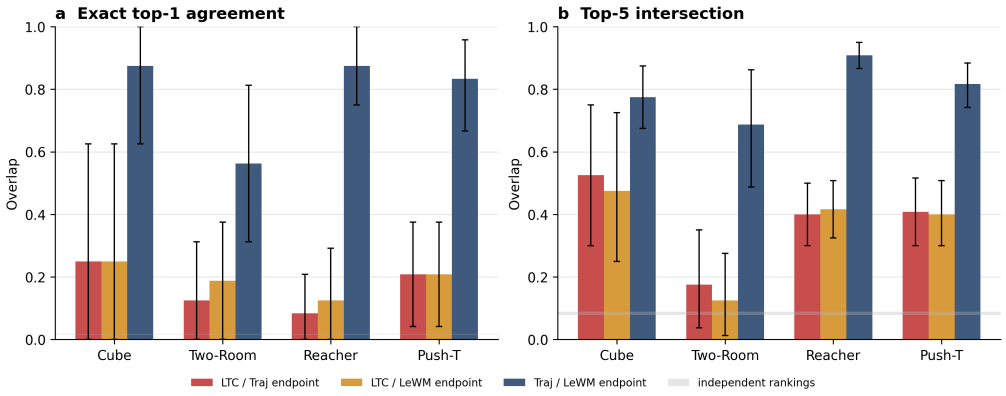

Figure 5: Agreement between candidate selectors on shared candidate pools. Panel (a) reports exact top-1 agreement, and panel (b) reports the fraction of shared candidates in the two top-5 sets. Bars are query means, error bars are query-level bootstrap 95% confidence intervals, and the gray band denotes the expectation for two independent rankings.

96% on Two-Room. Figure 2(d) shows Reacher. Scalar endpoint-cost matching does not match endpoint vectors, and the jitter test uses a training-style perturbation that may yield dynamically invalid latent states. These diagnostics do not isolate valid-path reasoning from richer endpoint scoring or corruption detection.

B.3 Candidate-Selector Agreement

We test whether LTC merely reproduces endpoint ranking by applying three selectors to the same candidate action sequences: LeWM endpoint distance, Traj-LeWM endpoint distance, and LTC alone. For selectors a and b, exact top-1 agreement indicates whether they choose the same minimum-cost candidate, while top-5 intersection is $|S_a^{(5)} \cap S_b^{(5)}|/5$. Metrics are averaged equally across queries, and 95% confidence intervals use 20,000 bootstrap samples of complete queries. Queries for this analysis are drawn from evaluation seeds 42–44 across all tasks.

The Traj-LeWM and LeWM endpoint selectors have exact top-1 agreement of 87.5%, 56.3%, 87.5%, and 83.3% on Cube, Two-Room, Reacher, and Push-T, respectively; their corresponding top-5 intersections are 77.5%, 68.8%, 90.8%, and 81.7%. In contrast, LTC-versus-endpoint top-1 agreement ranges from 8.3% to 25.0%, and top-5 intersection ranges from 12.5% to 52.5%. These measurements establish that LTC and endpoint distance induce different candidate rankings. Agreement alone does not establish that the difference improves execution or arises from useful intermediate-path information. The endpoint-matched analysis relates rankings to execution outcomes while leaving the full endpoint vectors unconstrained.

B.4 Open-Loop Latent Rollout Consistency

This diagnostic asks whether LeWM or Traj-LeWM more accurately predicts future latent states when both models receive the same recent observations and future actions. For each task, we select one window from each of 1,500 distinct expert episodes. Both models receive the first three observations and the identical recorded action sequence, then autoregressively predict the next ten latent states. At each prediction step, the corresponding observed future frame is encoded by the model being evaluated to obtain its reference latent state.

Because latent dimensions have different numerical scales, we normalize the squared error in each dimension by its variance, estimated from a separate set of 500 episodes, and take the root mean across dimensions. For every episode and prediction step, we then compute the paired difference

$$\Delta e = e_{\text{Traj-LeWM}} - e_{\text{LeWM}}.$$

Thus, a negative value means that Traj-LeWM has lower open-loop latent prediction error for the same input window and action sequence.

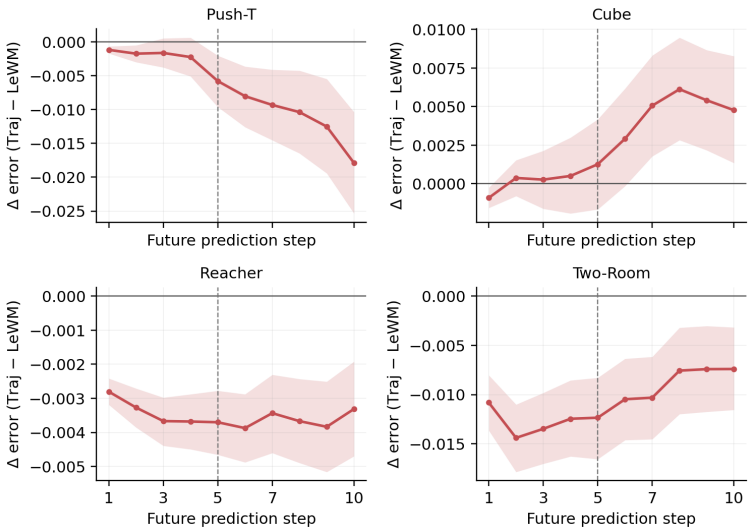

Figure 6: Paired change in variance-normalized open-loop latent rollout error. Negative values favor Traj-LeWM; shading denotes paired-bootstrap 95% confidence intervals. The dashed line marks the five-step planning horizon.

To summarize the planning-relevant range, we average Δe over the first five prediction steps for each episode and then average across episodes. The result is -0.00251 on Push-T (95% CI $[-0.00440, -0.00064]$), $+0.00030$ on Cube ($[-0.00122, 0.00182]$), -0.00343 on Reacher ($[-0.00394, -0.00291]$), and -0.01269 on Two-Room ($[-0.01535, -0.01013]$). The result is favorable on three tasks and indistinguishable from zero on Cube.

B.5 Qualitative Attention Comparison

The attention comparison provides a qualitative view of this representation effect. Along the same Two-Room trajectory, LeWM's final-layer CLS-to-patch attention largely follows the agent, whereas Traj-LeWM additionally responds to the central partition and doorway boundary (Fig. 3). This example visualizes a difference in the spatial evidence emphasized by the two encoders. It complements the endpoint-only measurements, but attention weights alone do not establish a causal contribution to planning success.

C Real-Robot Evaluation Details

Section 4.4 presents the task images and outcomes in Fig. 4 and table 5. This appendix specifies the recorded data and paired evaluation protocol. A completed rollout above the goal-distance threshold is counted as a target miss.

C.1 Data and Paired Evaluation

Data. Experiments were conducted using a Franka FR3 robot at a sampling frequency of 30 Hz. Each frame stores an eight-dimensional state and an eight-dimensional action, comprising seven arm joints and the gripper, together with three synchronized 640×480 RGB streams.

Paired protocol. The supplied evaluation report contains 20 paired episode–start samples. LeWM and Traj-LeWM receive the same initial observation and target joint state in each pair, and each model is executed once. The target is 45 recorded 30-Hz steps after the start, equivalent to 1.5 s, 15 10-Hz endpoints, or three 0.5-s macro steps. For sample i and model m, the report records the smallest seven-joint distance reached during the rollout,

$$d_{i,m}^{\min} = \min_t \left\| q_{i,m}(t) - q_i^{\text{goal}} \right\|_2 .$$

A rollout is successful only if it is not manually terminated for collision and $d_{i,m}^{\min} \leq 0.11$ rad. The threshold is the rounded Euclidean norm of the retained per-joint tolerances $[0.020, 0.050, 0.035, 0.050, 0.025, 0.055, 0.040]$ rad, whose norm is 0.108972 rad. The real-robot sequence in Figure 8 shows five synchronized time points from one archived episode through two external RGB cameras and the D515 camera. These frames document the retained observation streams only and are not associated with a LeWM or Traj-LeWM evaluation outcome.

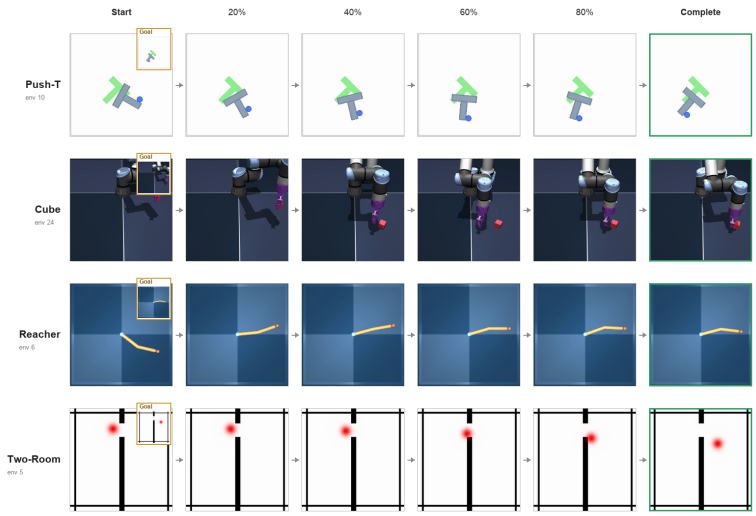

Figure 7: Representative retained Traj-LeWM rollouts for Push-T, Cube, Reacher, and Two-Room, with each row progressing from the start observation to the final frame.

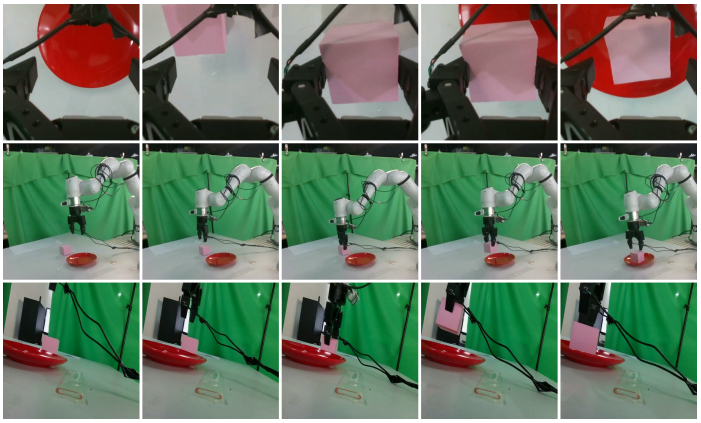

Figure 8: One Franka FR3 demonstration captured through two external cameras and the D515 camera at five synchronized time points. These sequences document completed executions and recorded observations.