Traj-LeWM: Path-Aware World-Model Planning via Latent Trajectory Cost

Xiaodi Huang[1,2,*] **Ziyi Ding**[3,5,*] **Yuchen Liu**[4] **Yuan Zhang**[7]
Jingtian Wan[6,5] **Xiao-Ping Zhang**[3] **Jiayu Chen**[4,5,†] **Zhang Zhang**[1,†] **Tao Huang**[2,†]

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, suggesting that endpoint information alone cannot reliably capture candidate quality. This motivates using complete-trajectory information in both representation learning and candidate ranking.

We attribute this behavior to two missing uses of trajectory information. First, LeWM's next-step prediction loss supervises local transitions between adjacent states, while SIGReg only regularizes the distribution of latent representations; neither requires the shared encoder to preserve information about a complete trajectory relative to its task goal. Consequently, although autoregressive rollout produces a sequence of predicted states, the representation space has not been directly trained to assess the goal-conditioned quality of a complete candidate trajectory. Second, LeWM ranks candidates only by the distance between their predicted latent endpoints and the goal representation. A predicted endpoint close to the goal does not guarantee that the same action sequence will perform well when executed in the environment, and candidates with similar predicted endpoints cannot be distinguished using their different intermediate evolutions.

To address these limitations, we propose Traj-LeWM. It retains LeWM's original prediction objective and endpoint score while introducing a goal-conditioned latent trajectory cost (LTC) over complete latent trajectories. The overall framework is illustrated in Fig. 1. LTC maps the goal-relative evolution of a complete latent trajectory to a learned scalar cost. During training, LTC is learned from pairwise preferences that favor goal-matched expert trajectories over synthetic negative trajectories and closed-loop execution failures; synthetic preferences also shape the shared encoder. During planning, LTC is combined with endpoint distance so that candidate ranking uses both endpoint and intermediate-path information. Across four simulated tasks, Traj-LeWM consistently outperforms LeWM, while evaluation on a physical Franka FR3 further demonstrates its real-robot feasibility. Controlled experiments separately show that trajectory supervision improves the shared representation and that LTC uses intermediate-path information to improve candidate ranking.

In summary, our contributions are threefold:

[*]Equal contribution. [†]Corresponding authors.
[1]Institute of Automation, Chinese Academy of Sciences;
[2]Shanghai Jiao Tong University; [3]Tsinghua Shenzhen International Graduate School; [4]The University of Hong Kong;
[5]INFIFORCE; [6]University of Science and Technology of China;
[7]Peking University.

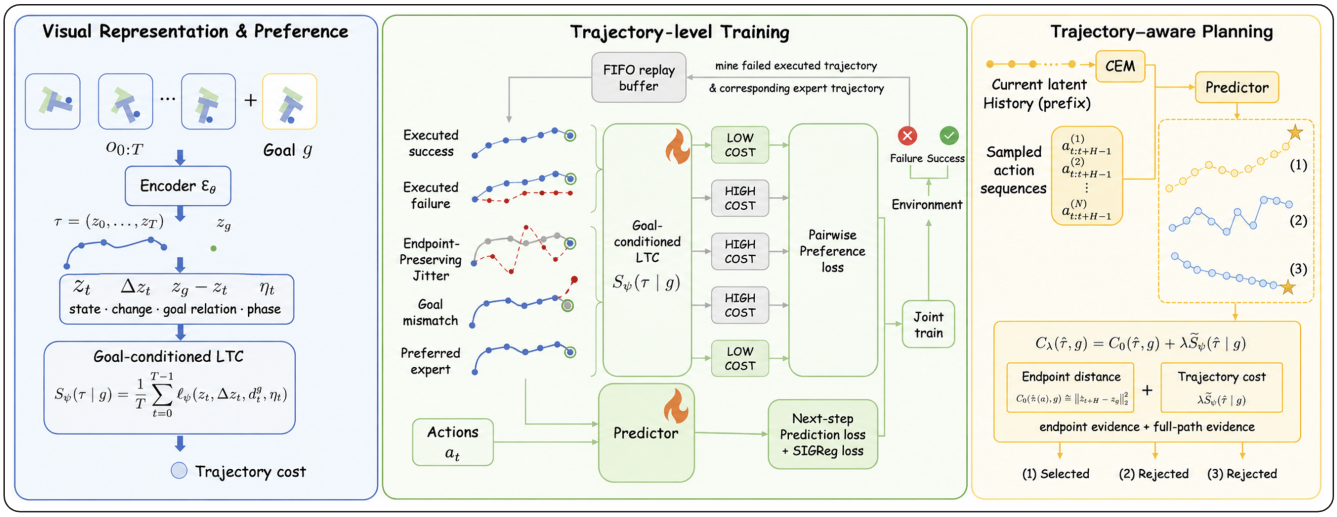

Figure 1: Overview of Traj-LeWM. Stage 1 encodes the observation trajectory and goal into latent representations used by LTC. Stage 2 trains LTC with synthetic and mined closed-loop trajectory preferences while retaining LeWM's next-step prediction objective. Stage 3 ranks CEM candidates using the joint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

- 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 endpoint-only, endpoint-matched, and fixed-endpoint analyses, together with ablations, further verify the effects of trajectory-level representation shaping and intermediate-path-aware candidate ranking.

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, Su, and Wang 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, Salakhutdinov, and Levine 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 replanning 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, Jain, and Abbeel 2020), Diffuser adapts it to classifier-guided trajectory planning (Janner et al. 2022), and energy-based models define scalar landscapes over structured inputs (Du and 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, Nair, and Levine 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 and 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, Goo, and Niekum 2020). Energy-based models likewise represent relative plausibility through a learned scalar landscape (LeCun et al. 2006; Du and 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.

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.

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 $\tau = (z_0, \ldots, z_T)$ 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.

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

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

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 $\tau^+ = (z_0^+, \ldots, z_T^+)$, 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':

$$x_{\text{gm}}^- = (\tau^+, g'). \quad (7)$$

The second negative adds noise to the intermediate states. 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

$$z_t^- = z_t^+ + \epsilon_t, \qquad 0 \leq t \leq T. \quad (8)$$

Let $\tau_{\text{jit}}^- = (z_0^-, \ldots, z_T^-)$ 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

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

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 $\tau_i^- = (\mathcal{E}_\theta(o_{i,0}^{\text{exec}}), \ldots, \mathcal{E}_\theta(o_{i,T_i^-}^{\text{exec}}))$ and pair it with an expert trajectory τ_i^+ having the same initial condition and goal:

$$(x_i^+, x_i^-) = \big((\tau_i^+, g_i), (\tau_i^-, g_i)\big). \quad (10)$$

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

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

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.

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.

Joint Training and Gradient Flow

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

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

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.

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 the closed-loop planning performance of Traj-LeWM across four simulated environments. We then organize controlled experiments around the two gaps identified in the motivation: Gap 1 tests how trajectory-preference supervision affects the shared representation, while Gap 2 tests whether LTC uses intermediate paths and whether this information improves candidate ranking. We further report a real-robot evaluation. Finally, ablations separate the effects of trajectory supervision during training and explicit LTC-based ranking during planning.

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 collects a small number of closed-loop failures solely to construct preference pairs for LTC; these trajectories are not added to the predictor's transition-training data. Accordingly, the comparison holds the predictor's transition data and learning objective fixed and evaluates the effect of introducing the proposed trajectory-preference module into the LeWM framework.

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. Bold indicates the best result for each task.

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, Nair, and Levine 2022). Complete details, including the number of evaluation trajectories, goal construction, interaction budget, data, training, and planning settings, are provided in the supplementary material.

Main results. With joint endpoint-plus-LTC scoring on every task, Traj-LeWM achieves mean success rates of 99%, 88%, 93%, and 94% on Push-T, Cube, Reacher, and Two-Room, respectively, averaged over three evaluation seeds. Compared with LeWM, these results correspond to gains of 3, 14, 7, and 7 percentage points. Thus, Traj-LeWM improves closed-loop planning across object manipulation, continuous control, and navigation tasks.

Analysis. Traj-LeWM outperforms LeWM on all four tasks and achieves the best results in the table on Push-T, Cube, and Reacher. It uses an end-to-end trained lightweight encoder, and LTC adds fewer than 1M parameters, leaving the overall model in the same parameter regime as LeWM. These results show that goal-conditioned trajectory preferences and path-sensitive evaluation improve planning within a lightweight world model.

Controlled Tests of the Two Gaps

We organize the following controlled experiments around the two gaps identified in the motivation. For Gap 1, we exclude LTC from planning and compare endpoint-only rankings between LeWM and Traj-LeWM, isolating the effect of trajectory-preference supervision on the shared representation. For Gap 2, we first test whether LTC improves candidate ranking when endpoint predictions are inaccurate, and then control endpoint information to verify that LTC uses intermediate paths. Execution outcomes are obtained by running the corresponding candidate action sequences in the environment.

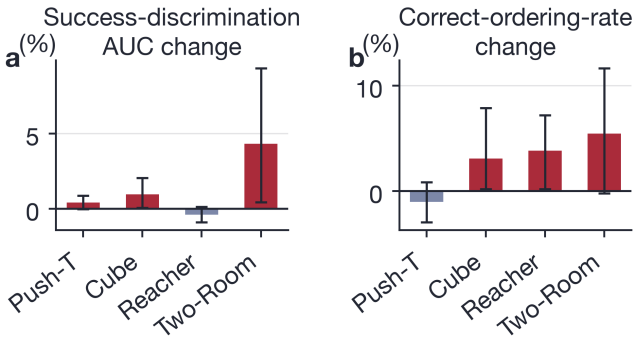

Figure 2: Controlled ranking tests. **(a)** Endpoint-only AUC change from LeWM to Traj-LeWM. **(b)** Correct-ordering change after adding LTC for candidates with large prediction–execution discrepancies.

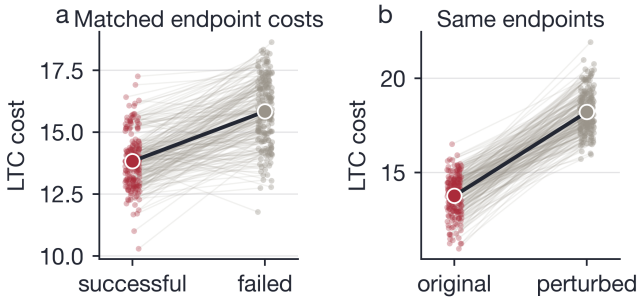

Figure 3: Endpoint-controlled tests on Reacher. **(a)** LTC correctly ranks 94.4% of 207 endpoint-matched success–failure pairs. **(b)** Intermediate-state perturbations increase LTC cost in all 200 pairs with fixed endpoints.

Testing Gap 1: representation shaping. To test Gap 1, we conduct a controlled experiment using the same candidate action sequences for both models. We first execute each sequence in the environment to obtain its ground-truth success or failure label. We then remove LTC from the planning score and use LeWM and Traj-LeWM to score the candidates solely by predicted endpoint distance in their respective representation spaces. By comparing these endpoint scores with the execution labels, we compute the success-discrimination AUC for each model. Because the candidate sequences and scoring form are held fixed, the AUC difference reflects the effect of trajectory-preference supervision on the shared representation.

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. 2a). These results show that trajectory-preference supervision can shape the shared encoder and improve the ability of endpoint scores to distinguish successful from failed executions.

Testing Gap 2: path use and candidate ranking. We evaluate LTC from two perspectives. First, we test whether LTC improves candidate ranking when endpoint predictions are inaccurate. We sample a set of candidate action sequences and use Traj-LeWM to roll out each sequence in latent space. For each candidate, we measure the latent-space discrepancy between its predicted endpoint and the endpoint obtained by executing the same action sequence in the environment and encoding the resulting observation. Within each start–goal instance, we construct success–failure candidate pairs and focus on pairs with large prediction–execution discrepancies. We then compare endpoint-only scoring with joint endpoint-plus-LTC scoring and measure how often the successful candidate is ranked ahead of the failed candidate. Adding LTC improves the correct-ordering rate by 3.1, 3.8, and 5.5 percentage points on Cube, Reacher, and Two-Room, respectively, while the change on Push-T is -1.0 percentage points (Fig. 2b). These results show that when endpoint predictions are inaccurate, LTC improves endpoint-based ranking on three of the four tasks.

We next control for endpoint information to directly test whether LTC uses intermediate paths. Within each start–goal instance, we match successful and failed executions one-to-one according to their Traj-LeWM endpoint costs. We retain only the hardest pairs, whose endpoint-cost difference is at most 0.25 times the within-instance interquartile range. On Reacher, we evaluate $1{,}344$ candidates, obtain 395 matched success–failure pairs, and retain the 207 hardest pairs under this criterion. When start–goal instances are weighted equally, LTC ranks the successful trajectory ahead of the failed trajectory with correct-ordering rates of 85.8%, 90.0%, 94.4%, and 83.0% on Push-T, Cube, Reacher, and Two-Room, respectively; Fig. 3a shows Reacher as an example. We then fix the initial and final states of each trajectory and perturb only its intermediate latent states, giving the original and perturbed trajectories identical endpoint costs. Across 200 comparisons per task, LTC assigns a higher cost to the perturbed trajectory in 100% of the comparisons on Push-T, Cube, and Reacher, and in 96% on Two-Room; Fig. 3b shows Reacher as an example. Together, these endpoint-controlled tests demonstrate that LTC uses intermediate-path information unavailable to endpoint-only scoring.

Real-Robot Evaluation

We evaluate goal-conditioned visual planning on a physical Franka FR3 using real manipulation trajectories recorded at 30 Hz from three synchronized RGB views. The evaluation contains 20 short-horizon start–goal tasks, each evaluated by both methods. Each task asks the robot to move from a recorded initial observation toward the visual goal observed 1.5 seconds later in the same trajectory; the corresponding seven-joint configuration is used to measure goal-reaching accuracy. LeWM and Traj-LeWM receive the same start and goal in every task. A rollout is considered successful if it avoids collision and its minimum seven-joint L_2 distance to the goal configuration does not exceed $0.11\,\mathrm{rad}$. A completed rollout above this threshold is counted as a target miss.

As shown in Table 3, Traj-LeWM succeeds in 14 of the 20 trials, compared with 10 for LeWM. Target misses decrease from six with LeWM to four with Traj-LeWM, while collision-induced terminations decrease from four to two.

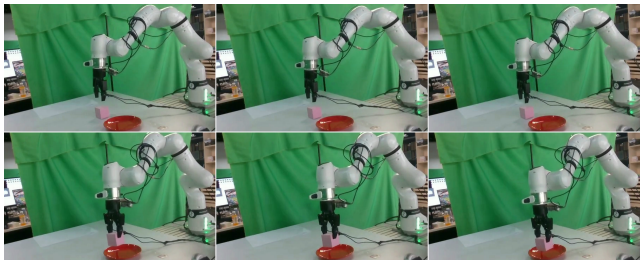

Figure 4: Real-robot setup and two example start–goal tasks. Each row shows the initial observation, an intermediate state, and the visual goal 1.5 seconds later.

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 3: Real-robot outcomes for LeWM and Traj-LeWM on the same 20 start–goal tasks using a Franka FR3.

Given the limited number of paired tasks, we interpret these outcomes as preliminary evidence of real-robot feasibility rather than a conclusive performance advantage.

Ablation Studies

Table 4 disentangles the method along two dimensions. The first three rows use the same endpoint score and vary the training supervision, isolating how trajectory preferences affect the shared latent model. The last three rows use the same trained Traj-LeWM model and vary the planning score, isolating the contribution of explicit LTC-based ranking.

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 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. 5). This redistribution is consistent with the quantitative endpoint-only results: trajectory supervision changes the spatial evidence emphasized by the shared encoder even when LTC is absent from the planning score.

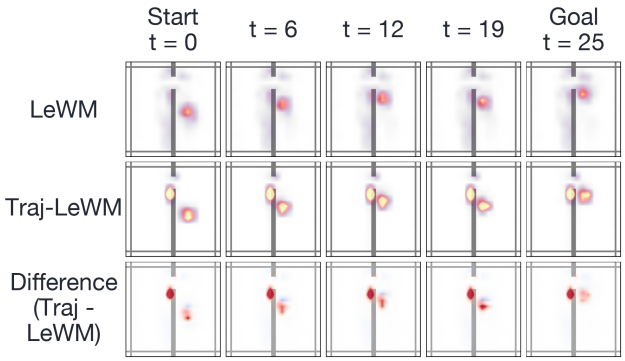

Figure 5: Final-layer CLS-to-patch attention along a Two-Room trajectory. Red indicates higher attention in Traj-LeWM/LeWM.

Effect of planning-time LTC scoring. With the trained Traj-LeWM model fixed, LTC-only ranking achieves an average success rate of 80.3% and reaches 94% on Reacher. Thus, the learned path score can independently provide a useful planning signal. Combining LTC with the endpoint term raises the average from 89.0% under endpoint-only scoring to 93.5%, including gains from 80% to 88% on Cube and from 84% to 93% on Reacher. Across tasks, the ablation shows that the two components contribute differently. Trajectory supervision mainly improves Cube and Two-Room under endpoint scoring, whereas LTC-only scoring is strongest on Reacher. This task-dependent behavior further suggests that intermediate-path evidence is most valuable when endpoint distance alone provides an incomplete ranking signal. The joint score achieves the highest average success rate, indicating that endpoint distance and LTC are complementary rather than interchangeable ranking signals.

5 Conclusion

This work addresses two limitations of goal-conditioned world models: local prediction objectives lack trajectory-level constraints, and endpoint distance ignores intermediate paths. We propose Traj-LeWM, which learns a goal-conditioned Latent Trajectory Cost (LTC) from trajectory-level preferences and uses trajectory information for both representation learning through the shared encoder and candidate ranking during planning. With joint endpoint-plus-LTC scoring, Traj-LeWM improves success rates over LeWM by 14, 7, 7, and 3 percentage points on Cube, Two-Room, Reacher, and Push-T, respectively. Controlled tests show that trajectory-preference supervision improves the ability of endpoint scores to discriminate execution outcomes on three tasks. They also show that LTC distinguishes trajectories when endpoint information is similar or identical and improves the ranking of candidates with inaccurate endpoint predictions on Cube, Reacher, and Two-Room. Ablations further verify the complementary roles of representation shaping and explicit path-sensitive ranking.

References

Assran, M.; Bardes, A.; Fan, D.; Garrido, Q.; Howes, R.; Komeili, M.; Muckley, M.; Rizvi, A.; Roberts, C.; Sinha, K.; Zholus, A.; Arnaud, S.; Gejji, A.; Martin, A.; Hogan, F. R.; Dugas, D.; Bojanowski, P.; Khalidov, V.; Labatut, P.; Massa, F.; Szafraniec, M.; Krishnakumar, K.; Li, Y.; Ma, X.; Chandar, S.; Meier, F.; LeCun, Y.; Rabbat, M.; and Ballas, N. 2025. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*.

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

Bradley, R. A.; and Terry, M. E. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4): 324–345.

Brown, D.; Goo, W.; Nagarajan, P.; and Niekum, S. 2019. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations. In *International Conference on Machine Learning (ICML)*, 783–792.

Brown, D. S.; Goo, W.; and Niekum, S. 2020. Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations. In *Conference on Robot Learning (CoRL)*, 330–359.

Chi, C.; Xu, Z.; Feng, S.; Cousineau, E.; Du, Y.; Burchfiel, B.; Tedrake, R.; and Song, S. 2025. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *The International Journal of Robotics Research*, 44(10–11): 1684–1704.

Christiano, P. F.; Leike, J.; Brown, T. B.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Destrade, M.; Bounou, O.; Le Lidec, Q.; Ponce, J.; and LeCun, Y. 2026. Value-Guided Action Planning with JEPA World Models. *arXiv preprint arXiv:2601.00844*. Presented at the World Modeling Workshop 2026.

Du, Y.; and Mordatch, I. 2019. Implicit Generation and Modeling with Energy-Based Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Eysenbach, B.; Salakhutdinov, R.; and Levine, S. 2019. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

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

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

Hansen, N. A.; Su, H.; and Wang, X. 2022. Temporal Difference Learning for Model Predictive Control. In *International Conference on Machine Learning (ICML)*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jain, A.; Chan, L.; Brown, D. S.; and Dragan, A. D. 2021. Optimal Cost Design for Model Predictive Control. In *Learning for Dynamics and Control (L4DC)*, 1205–1217.

Janner, M.; Du, Y.; Tenenbaum, J. B.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning (ICML)*.

Kaelbling, L. P. 1993. Learning to Achieve Goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations (ICLR)*.

LeCun, Y. 2022. A Path Towards Autonomous Machine Intelligence. *Open Review*.

LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; and Huang, F. J. 2006. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press.

Ma, Y. J.; Sodhani, S.; Jayaraman, D.; Bastani, O.; Kumar, V.; and Zhang, A. 2023. VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training. In *International Conference on Learning Representations (ICLR)*.

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

Park, S.; Frans, K.; Eysenbach, B.; and Levine, S. 2025. OGBench: Benchmarking Offline Goal-Conditioned RL. In *International Conference on Learning Representations (ICLR)*.

Sobal, U.; Zhang, W.; Cho, K.; Balestriero, R.; Rudner, T. G. J.; and LeCun, Y. 2025. Learning from Reward-Free Offline Data: A Case for Planning with Latent Dynamics Models. In *Advances in Neural Information Processing Systems*, volume 38, 43905–43941. Curran Associates, Inc.

Srinivas, A.; Jabri, A.; Abbeel, P.; Levine, S.; and Finn, C. 2018. Universal Planning Networks: Learning Generalizable Representations for Visuomotor Control. In *International Conference on Machine Learning (ICML)*, 4732–4741.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; Lillicrap, T.; and Riedmiller, M. 2018. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*.

Terver, B.; Yang, T.-Y.; Ponce, J.; Bardes, A.; and LeCun, Y. 2026. What Drives Success in Physical Planning with Joint-Embedding Predictive World Models? *Transactions on Machine Learning Research*.

Zhou, G.; Pan, H.; LeCun, Y.; and Pinto, L. 2024. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. *arXiv preprint arXiv:2411.04983*.