LEWM++: REPAIRING THE PLANNING INTERFACE OF LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models enable visual control without reconstructing pixels, yet remain unreliable over long horizons. We identify three planning mismatches that worsen with goal offset: *target mismatch* between a distant goal and a short reliable rollout; *search mismatch* between sparse effective action sequences and uninformed sampling; and *cost mismatch* between goal-reaching semantics and restrictive terminal-only cost. Addressing these failures requires a planning interface that effectively translates latent representations into actions. We introduce **LeWorldModel++ (LeWM++)**, a planning-interface framework built on a frozen latent world model. LeWM++ comprises three complementary components guided by a central principle: preserving global intent through the final goal while using a local target to drive short-horizon planning. LatentPathFlow generates a temporally aligned latent path and uses its endpoint as the local target; a final-goal-conditioned Action Chunk Prior guides CEM search with temporally coherent action-chunk proposals; and Min-over-Horizon scores each rollout by its closest approach to the local target. On the LeWM Control Suite, LeWM++ achieves the highest average success at all four evaluated goal offsets; as goal distance increases, its lead over LeWM grows from 12.0 to 32.3 pp. Across eight visual OGBench datasets, LeWM++ achieves 50.62% average success, exceeding the model-based planning method Latent Diffusion Planning (LDP) by 34.2 pp. On two real-robot tasks, LeWM++ outperforms goal-conditioned Diffusion Policy by 18.9 pp, demonstrating its control effectiveness under the hardware protocol. These results show that effective latent planning depends not only on predictive dynamics, but also on aligning the planner's target, proposal, and cost.

1 INTRODUCTION

Latent world models support visual control by encoding observations and image goals and predicting action-conditioned transitions in a compact space (Hafner et al., 2019; Rybkin et al., 2021; Zhou et al., 2025). Reconstruction-free models such as LeWM (Maes et al., 2026) allow goal-directed planning directly in that space. Yet a distant goal can lie beyond the model's reliable rollout horizon; extending the horizon accumulates prediction error and expands the action-sequence search space (Caselli et al., 2026). How a planner uses a fixed representation and dynamics model is therefore central to long-horizon control.

We focus on three points where latent predictions are translated into actions. A *target mismatch* occurs when a short rollout is optimized directly toward a distant final goal. A *search mismatch* arises when the Cross-Entropy Method (CEM) starts from an unstructured Gaussian despite goal-directed action structure in offline trajectories. A *cost mismatch* arises when terminal-only scoring penalizes a rollout that reaches its target within the planning window and then moves away. Temporal abstraction (Hafner et al., 2022; Zhang et al., 2026; Masip et al., 2026) and learned action guidance (Hansen et al., 2024; Wang et al., 2026a) address parts of this problem; SAGE (Cheng et al., 2026) combines learned subgoals and action proposals on a frozen world model. These observations motivate designing the planner's target, proposal, and rollout cost together.

To address these mismatches, we build **LeWM++** on a frozen LeWorldModel (LeWM) (Maes et al., 2026), following a central principle: the final goal preserves global intent, while a local target guides short-horizon planning. LatentPathFlow predicts a latent path under offline transition supervision

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.

and uses its endpoint as that target; a final-goal-conditioned Action Chunk Prior learns action-chunk proposals through advantage-weighted regression to guide CEM; and Min-over-Horizon scores the closest predicted approach to the local target across action-chunk boundaries. The controller executes the first optimized chunk and replans from the next observation. At $H = 25$ on the LeWM Control Suite, LeWM++ reaches 97.0% average success across four tasks; as goal offset increases, its lead over LeWM grows from 12.0 to 32.3 pp. Across eight visual OGBench datasets, it achieves 50.62% average success, 34.2 pp above LDP. The same interface also improves a differently structured latent world model on deformable-object tasks. LeWM++ also demonstrates effective real-robot control, outperforming goal-conditioned Diffusion Policy by 18.9 pp across two tasks.

Our contributions are **(1)** a unified analysis of target, search, and cost mismatches in latent planning; **(2)** a three-part interface that turns offline-supervised latent paths, action chunk prior, and closest-approach scoring into a short-horizon planner for frozen world models; and **(3)** validation across 16 benchmark tasks, including a 32.3 pp lead over LeWM at $H = 100$, 50.62% average success on visual OGBench, and an 18.9 pp gain over goal-conditioned Diffusion Policy on real robots, with ablations isolating each interface's contribution.

2 Preliminaries

Problem Setting. We study offline goal-conditioned sequential decision-making from image observations, formalized as a goal-conditioned Markov decision process $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{G}, \mu, r, p, \gamma)$. Here, $o_t \in \mathcal{O}$ and $a_t \in \mathcal{A}$ denote the image observation and continuous action at time t, respectively; $g \in \mathcal{G} \subseteq \mathcal{O}$ is a goal image; μ is the initial-observation distribution; $p(o_{t+1} \mid o_t, a_t)$ denotes the environment dynamics; $r(o_t, g)$ is the goal-conditioned reward; and $\gamma \in [0, 1)$ is the discount factor. Training uses only an offline dataset $\mathcal{D}$ whose trajectories have the form $\tau = (o_0, a_0, \ldots, o_{T-1}, a_{T-1}, o_T)$. Training goals are relabeled from observations in the offline dataset, and we use the reward $r(o_t, g) = 0$ when o_t reaches g and $r(o_t, g) = -1$ otherwise.

Latent World Model. Within this framework, we use LeWM (Maes et al., 2026) as the base world model for visual dynamics. LeWM consists of an image encoder E_ξ, which maps an observation o_t to $z_t = E_\xi(o_t)$, and an action-conditioned predictor F_ω, which estimates the next latent state as $\hat{z}_{t+1} = F_\omega(z_t, a_t)$. The encoder and predictor are jointly optimized on $\mathcal{D}$:

$$\mathcal{L}_{\text{LeWM}} = \mathcal{L}_{\text{pred}} + \lambda_{\text{sig}} \, \text{SIGReg}(Z), \qquad \mathcal{L}_{\text{pred}} = \|\hat{z}_{t+1} - z_{t+1}\|_2^2 . \tag{1}$$

Here, $z_{t+1} = E_\xi(o_{t+1})$ is the encoded target for future prediction, Z denotes a batch of latent embeddings, and λ_{sig} controls the weight of SIGReg (Balestriero & LeCun, 2025) relative to the prediction loss. SIGReg regularizes the latent distribution, preventing representation collapse and inducing a well-structured latent geometry.

Latent-Space Planning. Given the latent state $z_t = E_\xi(o_t)$ and goal representation $z_g = E_\xi(g)$, we consider a horizon-k candidate action sequence $a_{t:t+k-1} = (a_t, \ldots, a_{t+k-1})$. Starting from z_t, the frozen predictor F_ω recursively rolls out the corresponding future latent trajectory $\hat{z}_{t+1:t+k}$. Standard latent planning then uses a trajectory optimizer such as CEM to find an action sequence whose terminal prediction is closest to the goal. We define the terminal objective as

$$J_{\text{terminal}}\left(a_{t:t+k-1}\right) \coloneqq \|\hat{z}_{t+k} - z_g\|_2^2 . \tag{2}$$

3 Method

As shown in Fig. 1, LeWM++ combines three planning interfaces. LatentPathFlow predicts a latent path supervised by offline transitions and uses its endpoint as the local planning target (Sec. 3.1). The Action Chunk Prior provides goal-conditioned action-chunk proposals to guide CEM under a limited search budget (Sec. 3.2). Finally, Min-over-Horizon scores each rollout by its closest approach to that target; the controller executes one chunk and replans (Sec. 3.3).

3.1 LatentPathFlow: Temporally Aligned Latent Path Generation

Standard latent planning assumes that the goal lies within a reliable model rollout. For distant goals, a short rollout may leave all predicted terminal states far from the goal, making terminal distance a weak local signal; extending the horizon instead increases model error and search complexity. We

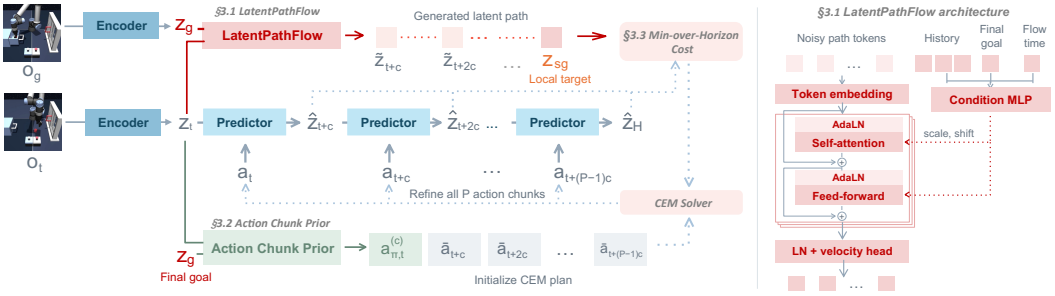


Figure 1: **Overall framework of LeWM++.** *Left:* LatentPathFlow predicts a latent path whose endpoint becomes the local target. The Action Chunk Prior supplies first-chunk proposals to CEM, which refines all action chunks using Min-over-Horizon rollout scores; the controller executes only the first chunk before replanning. *Right:* LatentPathFlow architecture.

address this target mismatch with **LatentPathFlow**. Conditioned on recent observation and the final goal, it predicts a latent path supervised by short offline transitions and uses its endpoint as the target for the planning cycle. Replanning toward successive endpoints drives progress toward distant goal.

Latent Path Generation. Given the most recent ℓ latent states $z_{t-\ell+1:t}$ and the goal latent $z_g = E_\xi(g)$, LatentPathFlow models $p_\eta\big(\mathbf{z}^*_{t\rightsquigarrow t+k} \mid z_{t-\ell+1:t}, z_g\big)$, where k is the local-subgoal horizon and $\mathbf{z}^*_{t\rightsquigarrow t+k}$ denotes an ordered target latent path spanning the next k environment steps. Its temporal discretization is specified together with chunk-level planning in Sec. 3.3. Unlike a model that predicts only an isolated endpoint z_{t+k}, LatentPathFlow generates the entire path leading to that endpoint. An endpoint-only regressor may produce a plausible representation without transition structure supported by the data; path-level modeling additionally constrains the endpoint through intermediate state transitions observed in the offline trajectories.

Conditional Flow Matching. Even under the same current observation and final goal, an environment may admit multiple feasible local paths. Regressing a single path with mean-squared error can average across modes and erase their motion structure. We instead use conditional flow matching (Lipman et al., 2023) to learn a distribution over complete latent paths. For Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ with the same shape as the target path and flow time $u \sim \mathcal{U}(0, 1)$, define the linear probability path $x_u = (1-u)\epsilon + u\mathbf{z}^*_{t\rightsquigarrow t+k}$. We train the conditional vector field with

$$\mathcal{L}_{\text{LatentPath}} = \mathbb{E}\left[\left\| v_\eta(x_u, u \mid z_{t-\ell+1:t}, z_g) - \big(\mathbf{z}^*_{t\rightsquigarrow t+k} - \epsilon\big)\right\|_F^2\right]. \tag{3}$$

This objective jointly models dependencies across future offsets while preserving multiple modes of the conditional path distribution.

Path-Token Transformer. We parameterize v_η with a Path-token Transformer. Each noisy future latent becomes a path token with a temporal embedding; the history, final goal, and flow time form a global condition injected into the Transformer blocks through adaptive layer normalization. Self-attention models dependencies among future offsets, and a velocity head outputs the vector field for the complete path. At inference, we integrate the learned field from Gaussian noise x_0 to obtain a conditional latent path $\tilde{\mathbf{z}}_{t\rightsquigarrow t+k}$. Planning uses only the generated endpoint $z_t^{\text{sg}} = \tilde{z}_{t+k}$ as the local target and does not require the model rollout to track every generated path token.

3.2 Action Chunk Prior for Goal-Directed Search

Under a limited sampling budget, Gaussian-initialized CEM may struggle to find goal-directed action sequences. So we learn a goal-conditioned action prior from offline datasets to guide its search.

Why Action Chunks. Single actions are often ambiguous: different goals may share the same initial control, and offline noise can obscure its intent. A contiguous sequence better reveals how control unfolds toward a goal. We therefore construct c-action chunks $a_t^{(c)} := (a_t, \ldots, a_{t+c-1})$ and learn $\pi_\phi(a_t^{(c)} \mid z_t, z_g)$. These local proposals guide CEM while leaving the full sequence available for model-based refinement.

Learning a Chunked Critic and Policy. Each offline sample contains a contiguous within-trajectory chunk $a_t^{(c)}$ and a relabeled goal g, whose frozen encoding is denoted by z_g. We extend the IQL objectives (Kostrikov et al., 2022; Li et al., 2026a) to learn a goal-conditioned state value

$V_\psi(z_t, z_g)$ and a chunk critic $Q_\theta(z_t, a_t^{(c)}, z_g)$. The value objective is

$$\mathcal{L}_V(\psi) = \mathbb{E}_{z_t, a_t^{(c)}, z_g \sim \mathcal{D}} \left[L_2^\tau \Big(Q_\theta(z_t, a_t^{(c)}, z_g) - V_\psi(z_t, z_g) \Big) \right], \tag{4}$$

where L_2^τ is the expectile regression loss. Let $R_t^{(c)}(g)$ denote the terminal-masked discounted return accumulated within the chunk, and let $m_t^{(c)}(g)$ be 0 if the goal is reached within the chunk and 1 otherwise. The chunk critic uses the c-step target

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{\mathcal{D}} \left[\Big(R_t^{(c)}(g) + \gamma^c m_t^{(c)}(g) V_\psi(z_{t+c}, z_g) - Q_\theta(z_t, a_t^{(c)}, z_g) \Big)^2 \right]. \tag{5}$$

Unlike a one-step critic, Q_θ conditions on the full action chunk and bootstraps from the state c steps.

We fit the Gaussian chunk policy by advantage-weighted regression (Peng et al., 2019),

$$\begin{aligned} A_t &= Q_\theta(z_t, a_t^{(c)}, z_g) - V_\psi(z_t, z_g), \\ \mathcal{L}_{\text{ActionPrior}} &= -\mathbb{E}_{\mathcal{D}} \left[\exp(\beta A_t) \log \pi_\phi \Big(a_t^{(c)} \mid z_t, z_g \Big) \right]. \end{aligned} \tag{6}$$

This objective stays within the offline action support while emphasizing chunks with higher goal-conditioned advantage. We condition the prior on the original goal z_g and use the LatentPathFlow endpoint only to score local rollouts. This separation gives action proposals a globally consistent direction and local scoring a trajectory-supervised target, without making both interfaces depend on the same subgoal estimate. App. A.4 provides further details on Action Prior training.

3.3 Closed-Loop Action-Prior Planning with Min-over-Horizon

At each planning cycle, LeWM++ combines the local target, action-prior-guided CEM search, and Min-over-Horizon (MoH) scoring, then executes one chunk and replans.

Temporally Aligned Planning. Let c be the chunk length and k the local horizon, with k divisible by c. Each candidate sequence $a_{t:t+k-1}$ contains $P = k/c$ chunks $a_{t+jc}^{(c)}$ for $j = 0, \dots, P-1$. Starting from $\hat{z}_t = z_t$, LeWM rolls out one chunk-level transition, denoted by $F_\omega^{(c)}$:

$$\hat{z}_{t+(j+1)c} = F_\omega^{(c)} \Big(\hat{z}_{t+jc}, a_{t+jc}^{(c)} \Big), \qquad j = 0, \dots, P-1. \tag{7}$$

Under this discretization, the target path introduced in Sec. 3.1 is $\mathbf{z}^*_{t \rightsquigarrow t+k} = (z_{t+c}, z_{t+2c}, \dots, z_{t+Pc})$. LatentPathFlow uses its endpoint $z_t^{\text{sg}} = \tilde{z}_{t+k}$ as the local target. Because $k = Pc$, it has the same temporal offset as the last predicted state $\hat{z}_{t+Pc}$ of the P-block rollout. The prior, model rollout, and planning therefore share the same c-step unit.

Action-Prior Guidance. Let $a_{\pi,t}^{(c)} = \mu_\phi(z_t, z_g)$ denote the mean action chunk proposed by the learned prior. At each planning cycle, it initializes the first block of the initial CEM mean:

$$\boldsymbol{\mu}_t^{(0)} = \Big(a_{\pi,t}^{(c)}, \bar{a}_{t+c}^{(c)}, \dots, \bar{a}_{t+(P-1)c}^{(c)} \Big), \qquad \boldsymbol{\sigma}_t^{(0)} = \sigma_0 \mathbf{1}, \tag{8}$$

Here, the superscript (0) denotes the distribution before the first CEM update. The remaining blocks use the corresponding blocks from the shifted previous plan and are zero-padded when unavailable.

In addition to initializing the first block of the CEM mean, the prior can supply first-chunk samples at every refinement iteration. Of the N complete candidate sequences, K_π draw their first chunk from $\pi_\phi(\cdot \mid z_t, z_g)$ and all later chunks from the current CEM Gaussian; the remaining $N - K_\pi$ sequences are sampled entirely from that Gaussian. All candidates are rolled out with the same world model and scored by the same Min-over-Horizon objective. Thus, $K_\pi = 0$ gives mean-only guidance, whereas $K_\pi > 0$ reinjects policy-sampled first chunks throughout CEM refinement. This hybrid proposal biases part of the search toward the offline action support while retaining Gaussian candidates and iterative CEM updates for local correction, thereby substantially reducing out-of-distribution action search in the offline setting. Unlike IQL-TD-MPC, which plans only over policy samples and removes iterative MPPI refinement (Chitnis et al., 2023), LeWM++ retains both policy-supported candidates and model-based local search.

Min-over-Horizon Closed-Loop Planning. A terminal objective evaluates only $\hat{z}_{t+k}$ and can penalize a sequence that reaches the local target at an earlier chunk boundary before moving away. We

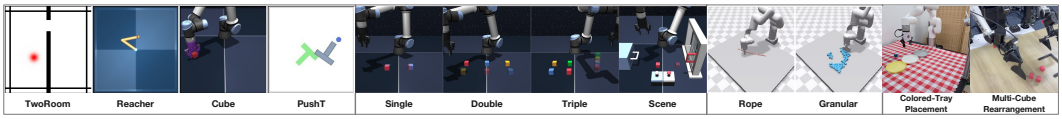


Figure 2: Representative observations from (a) LeWM Control Suite; (b) visual OGBench; (c) DINO-WM deformable manipulation; and (d) real-robot tasks.

instead score each candidate by its closest predicted approach:

$$J_{\mathrm{MoH}}\left(a_{t:t+k-1}\right) = \min_{j\in\{1,\ldots,P\}} \left\|\hat{z}_{t+jc} - z_t^{\mathrm{sg}}\right\|_2^2 . \tag{9}$$

Taking the minimum over $j = 1, \ldots, P$ permits arrival at any future chunk boundary without assigning a low cost before an action is taken. CEM updates its distribution using the action sequences with the lowest rollout costs; after L iterations, the controller executes the first optimized chunk and replans rather than following the remaining sequence. MoH identifies the closest predicted approach within the planning window, while one-chunk execution avoids committing to the full sequence.

4 Experiments

We first examine how performance changes with goal offset on the LeWM Control Suite (Sec. 4.2), then evaluate long-horizon image-goal control on eight visual OGBench datasets, including compositional tasks (Sec. 4.3). We next test generality with another frozen latent world model (Sec. 4.4) and assess control on two real-robot tasks (Sec. 4.5). Finally, ablations isolate the contributions of the local target, action prior, and rollout scoring (Sec. 4.6). We provide additional diagnostics, sensitivity tests, runtime measurements, and complete results in Apps. D and E.

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate three complementary simulated suites. The LeWM Control Suite contains four tasks (Maes et al., 2026) with goal offsets $H \in \{25, 50, 75, 100\}$. Visual OGBench comprises eight datasets covering cube manipulation and scene rearrangement under play and noisy data, testing long-horizon and compositional image-goal control beyond the local planning horizon (Park et al., 2025). Following DINO-WM (Zhou et al., 2025), we also evaluate deformable-object manipulation with an XArm (Rope and Granular), using Chamfer distance, where lower is better. Sec. 4.5 evaluates two real-robot tasks. Representative observations are shown in Fig. 2, and complete environment descriptions appear in App. C.

Evaluation Protocol. Unless stated otherwise, we train with three independent random seeds and report the mean and standard deviation across the resulting runs. On the LeWM Control Suite, all experiments use the same normalized interaction budget: the maximum episode length is $B = 2H$, and we evaluate 50 episodes per task. On OGBench, we evaluate each of the five predefined tasks in every dataset over 50 episodes and then average their scores. For deformable-object manipulation, we follow the DINO-WM protocol (Zhou et al., 2025) and evaluate 10 instances per task.

LeWM++ uses action chunks of length $c = 5$, a local latent path spanning $k = 10$ environment steps, and a planning horizon of $P = 2$ action chunks; it executes only the first chunk before replanning. On the LeWM Control Suite and OGBench, CEM uses 300 candidates, 5 iterations, and 30 elites, while LatentPathFlow uses 16 integration steps. The Control Suite uses only the mean of the Action Prior to initialize search ($K_\pi = 0$). On OGBench, each CEM iteration samples the first action chunk of $K_\pi = 200$ among $N = 300$ candidates from the stochastic Action Prior. In contrast, canonical LeWM initializes search from an unstructured Gaussian and requires 30 CEM iterations to avoid a substantial performance drop. Training details are provided in App. A, and task and evaluation settings for each suite are provided in App. C.

4.2 Planning Effectiveness and Long-Horizon Scaling

Baselines. We first compare LeWM++ with latent world-model planners TD-MPC2 (Hansen et al., 2024), DINO-WM (Zhou et al., 2025), and LeWM (Maes et al., 2026); subgoal-guided

Table 1: Success rate (%) on the four LeWM Control Suite tasks at $H = 25$ protocol.

Category	Method	TwoRoom	Reacher	PushT	Cube	Average
Latent Planning References	TD-MPC2	—	—	12.0	4.0	—
	DINO-WM	100.0±0.0	79.0±5.1	74.0±4.5	86.0±4.7	84.8
	LeWM	97.3±1.2	90.0±2.0	80.0±6.0	72.7±7.6	85.0
Subgoal Guidance	SAGE	100.0±0.0	96.7±2.5	85.3±3.1	92.0±3.5	93.5
Policy Guidance	GC-IDM	100.0±0.0	99.7±0.3	84.2±2.8	98.7±0.6	95.7
	INTACT	97.7±1.1	96.8±0.9	93.1±0.9	98.2±0.7	96.4
	PRISM	100.0±0.0	95.3±1.2	92.7±5.0	62.7±8.5	87.7
Planning Objective	RC-aux	98.0±1.4	87.2±6.4	90.8±3.3	76.0±7.5	88.0
Joint Planning Interfaces	LeWM++	100.0±0.0	96.7±2.5	96.0±1.6	95.3±0.9	97.0
	LeWM++ w/o Action Prior	100.0±0.0	88.7±3.4	92.7±0.9	86.7±1.9	92.0
	LeWM++ w/o MoH	100.0±0.0	98.7±1.9	85.3±3.4	92.0±3.3	94.0

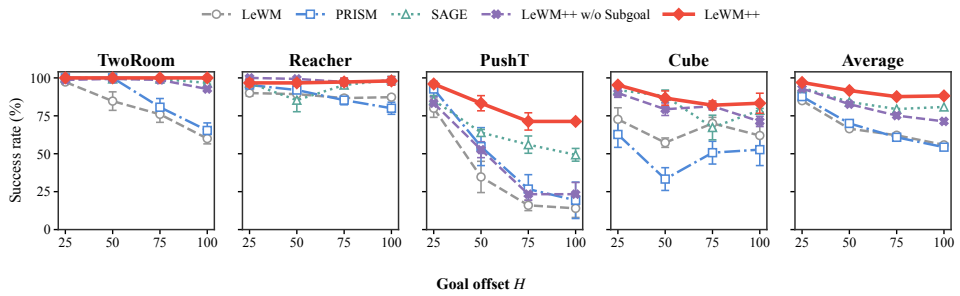


Figure 3: Success rate versus goal offset on the LeWM Control Suite. Task panels show mean ± standard deviation over three runs; Average shows the four-task macro-average.

SAGE (Cheng et al., 2026); action-guided GC-IDM (Nguyen et al., 2026), INTACT (Sun et al., 2026), and PRISM (Wang et al., 2026a); and RC-aux (Li et al., 2026b) at Tab. 1.

Results. At Tab. 1, LeWM++ achieves the highest macro-average success rate of 97.0%, 12.0 percentage points above LeWM. Removing the Action Chunk Prior lowers it to 92.0%, and replacing MoH with terminal scoring lowers it to 94.0%. These ablations show the contributions of action proposals and rollout scoring; Sec. 4.6 analyzes their design further. Fig. 3 shows that LeWM++ has the highest average at every goal offset, losing only 8.85 percentage points from $H = 25$ to $H = 100$. Its lead over LeWM grows from 12.00 to 32.32 points, while its lead over the variant without LatentPathFlow grows from 4.00 to 16.82 points. At $H = 100$, PushT success is 71.3% versus 23.3% without LatentPathFlow, highlighting the value of local targets as the final goal recedes. At comparable per-plan latency, LeWM with 5 CEM iterations reaches 77.7% success at 25.45 ms, whereas LeWM++ reaches 97.0% at 26.42 ms. Full latency results appear in Tabs. 10 and 14.

4.3 Long-Horizon and Compositional Control on Visual OGBench

On eight visual OGBench datasets (Park et al., 2025), we evaluate long-horizon and compositional image-goal control against model-free methods GCBC (Ghosh et al., 2021), GCIQL (Kostrikov et al., 2022; Park et al., 2025), CRL (Eysenbach et al., 2022), and HIQL (Park et al., 2025), and model-based methods Latent Diffusion Planning (LDP) (Xie et al., 2025) and LeWM (Maes et al., 2026). LDP encodes images with a VAE trained for pixel reconstruction, plans trajectories in the resulting latent space, and recovers actions through inverse dynamics. We adapt LDP's conditioning inputs for image-goal tasks; details are provided in App. C.2.

As shown in Tab. 2, LeWM++ reaches a 50.62% average success, the highest among the compared model-based methods, and is competitive with the model-free hierarchical method HIQL. LeWM uses the same frozen representation backbone but achieves only 1.1%, underscoring the role of planning-interface design. Compared with direct execution of the Action Prior, LeWM++ gains another 5.01 percentage points, showing that explicit planning provides further benefits. The triple-cube datasets remain challenging, with success below 20%.

4.4 Generality across Latent World Models

To assess whether the planning interface also works with a different latent world-model architecture, we evaluate LeWM++ with frozen DINO-WM (Zhou et al., 2025) named as DINO-WM++ on Rope

Table 2: Success rate (%) on eight visual OGBench datasets.

Dataset	Model-free				Model-based		Ours	
	GCBC	GCIQL	CRL	HIQL	LDP	LeWM	Direct Action Prior	LeWM++
visual-cube-single-play-v0	5±1	30±5	31±15	82.0±5.7	36.1±2.8	1±1	65.3±4.7	66.3±2.1
visual-cube-double-play-v0	1±1	1±1	2±1	42.8±3.0	3.1±0.5	0±0	28.8±3.1	45.8±3.2
visual-cube-triple-play-v0	15±2	15±1	17±2	20.0±0.2	1.7±0.5	2±1	18.7±0.2	18.4±0.4
visual-scene-play-v0	12±2	12±2	11±2	42.4±6.1	19.9±0.5	0±0	47.6±1.7	54.9±1.6
visual-cube-single-noisy-v0	14±3	48±3	39±30	98.8±0.7	41.3±1.7	0±0	91.9±0.2	93.3±1.4
visual-cube-double-noisy-v0	5±1	22±2	6±3	58.4±4.2	3.5±1.2	3±1	49.8±3.1	58.0±2.0
visual-cube-triple-noisy-v0	16±1	12±1	16±1	20.8±1.2	1.9±1.5	0±0	19.2±0.2	19.3±1.3
visual-scene-noisy-v0	13±2	12±4	15±2	44.4±3.7	23.9±1.0	3±1	43.6±1.4	48.8±3.2
Average	10.1	19.0	17.1	51.2	16.4	1.1	45.6	50.6

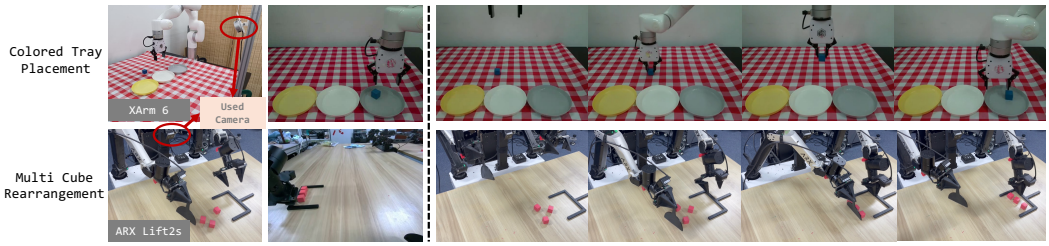


Figure 4: Real-robot tasks and example rollouts. Top: Colored-Tray Placement on an XArm 6. Bottom: Multi-Cube Rearrangement on an ARX Lift2s.

and Granular manipulation (protocol in App. C.3). Tab. 3 compares DINO-WM++ with DINO-WM, DreamerV3 (Hafner et al., 2023), TD-MPC2 (Hansen et al., 2024), and two component controls.

DINO-WM++ achieves the lowest Chamfer distance on both tasks, reducing DINO-WM from 0.80 to 0.31 on Rope and from 0.29 to 0.17 on Granular. Direct Action Prior execution reaches 1.27 and 0.50, while MoH alone improves DINO-WM to 0.42 and 0.27, respectively. These comparisons support the interface's applicability to a latent world model with a different architecture.

Table 3: Chamfer distance on DINO-WM Rope and Granular manipulation (lower is better).

Method	Rope ↓	Granular ↓
DreamerV3	2.49	1.05
TD-MPC2	2.52	1.21
DINO-WM	0.80±0.11	0.29±0.05
Action Prior (Direct Execution)	1.27±0.34	0.50±0.05
DINO-WM + MoH	0.42±0.23	0.27±0.09
DINO-WM++ (Ours)	**0.31±0.08**	**0.17±0.04**

4.5 Real-World Evaluation

We test two image-goal tasks on XArm and ARX Lift2s (Fig. 4): Colored-Tray Placement selects one of three containers for a cube, while Multi-Cube Rearrangement specifies target positions for one to three cubes. Each task uses 120 offline training trajectories and 45 evaluation trials; rearrangement includes 15 trials per cube count. We compare LeWM++ with goal-conditioned Diffusion Policy (GC-DP) (Chi et al., 2023) and LeWM, which shares the frozen representation and dynamics but initializes CEM from an unstructured Gaussian. Protocol details appear in App. C.4.

Tab. 4 shows that LeWM++ succeeds in 65/90 trials (72.2%), versus 48/90 (53.3%) for GC-DP, with gains on both tasks. The margin widens in Multi-Cube Rearrangement: with two cubes, success is 10/15 versus 4/15; with three, 8/15 versus 3/15. LeWM is N/A because its unstructured search repeatedly triggered motor protection or approached the workspace boundary, so we halted its evaluation under the protocol. Guided by the Action Chunk Prior and bounded planning perturbations, LeWM++ completed all trials under the shared clipping, workspace, and emergency-stop constraints. GC-DP failures mostly involved selecting the wrong target; LeWM++ made fewer such errors by scoring predicted rollouts against the specified goal. These results highlight the complementary roles of data-supported proposals and goal-sensitive planning.

4.6 Dissecting the Planning Interface

Tab. 1 and Fig. 3 have established the contributions of the local target, action prior, and rollout scoring. We now examine design choices within the target predictor and action prior.

Table 4: **Real-robot results.** Each entry reports successful trials / total trials. LeWM is marked N/A because its evaluation was halted after repeated safety-limit activations. Best results are bolded.

	Colored-Tray	Multi-Cube Rearrangement			Overall
Method		1 cube	2 cubes	3 cubes	
GC-DP	32/45 (71.1%)	9/15 (60.0%)	4/15 (26.7%)	3/15 (20.0%)	48/90 (53.3%)
LeWM	N/A	N/A	N/A	N/A	N/A
LeWM++	**38/45 (84.4%)**	**9/15 (60.0%)**	**10/15 (66.7%)**	**8/15 (53.3%)**	**65/90 (72.2%)**

Table 5: H25 protocol of LeWM success (%) under different target interfaces. Task columns show mean ± std. The three predictors are parameter-matched within the LeWM++ planner.

Target interface	TwoRoom	Reacher	PushT	Cube	Avg.
w/o Subgoal Path	98.67±1.89	**100.00±0.00**	83.33±4.11	90.00±2.83	93.00
MLP	**100.00±0.00**	98.67±0.94	88.67±4.11	70.67±4.11	89.50
EndpointFlow	**100.00±0.00**	88.00±4.32	91.33±3.40	74.00±3.27	88.33
LatentPathFlow	**100.00±0.00**	96.67±2.49	**96.00±1.63**	**95.33±0.94**	**97.00**

Target Interface: How to Generate a Latent Path? Within the same LeWM++ planner, we compare three parameter-matched predictors: an MLP that regresses the local endpoint, EndpointFlow that generates that endpoint using flow matching, and LatentPathFlow that generates the intermediate path and endpoint. Tab. 5 shows that LatentPathFlow has the highest average success rate (97.00%) and leads on PushT and Cube, supporting path-level prediction over endpoint-only alternatives. The gap is especially large on Cube: LatentPathFlow reaches 95.33%, versus 74.00% for EndpointFlow and 70.67% for the MLP.

This parameter-matched comparison further supports intermediate-path prediction as a way to produce useful local planning targets. Additional action-consistency and closed-loop progress diagnostics appear in App. D.2.

Qualitative Closed-Loop Alignment. Fig. 5 aligns generated subgoals, LeWM++ terminal imaginations, and realized observations across $H = 50$ rollouts. Agreement between subgoals and imaginations shows that planning follows the generated targets; agreement between imaginations and observations shows that predictions track execution. Together, these views suggest that the intermediate targets are actionable. Auxiliary decoders visualize frozen latents only and do not affect planning or evaluation.

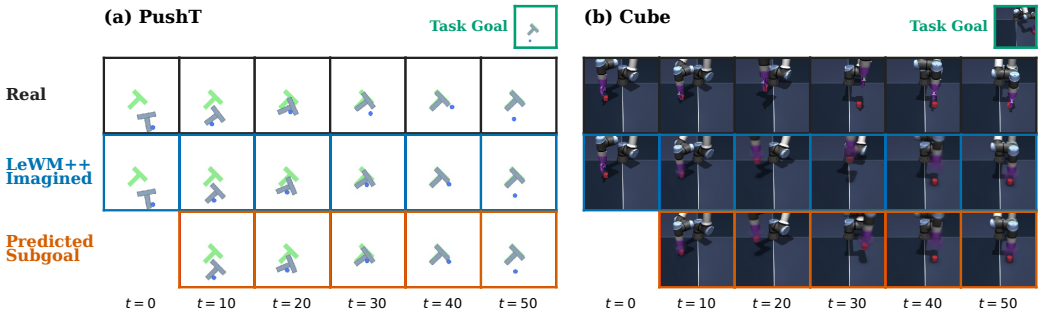


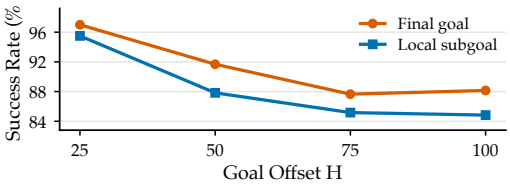
Figure 5: PushT and Cube closed-loop alignment ($H = 50$): observations, LeWM++ terminal imaginations, and LatentPathFlow subgoals at 10-step intervals.

Proposal Interface: How Should the Action Prior Guide Search? We isolate three decisions: how the prior enters CEM, whether it predicts individual actions or chunks, and whether it is conditioned on the final goal z_g or the LatentPathFlow subgoal z^{sg}.
The default uses the policy mode only to initialize the first action block of the CEM mean; model-based search then optimizes the full action sequence. As shown in Fig. 6(a), it reaches a 97.00% average at $H = 25$, above zero initialization (91.83%), first-round injection of 64 policy samples (96.33%), and a policy-mode anchor retained at every iteration (96.17%).

Replacing the five-step chunk with a single-action output lowers success to 94.00%, suggesting that chunk proposals are more effective in this setting. In Fig. 6(b), final-goal conditioning yields a higher

Design axis	Variant	$H = 25$
Default	**Mode init., chunk** ($c = 5$)	**97.00**
CEM integration	Zero initialization	91.83
	Population injection (64)	96.33
	Mode init. + anchor	96.17
Temporal output	Single action ($c = 1$)	94.00

(a) CEM integration and temporal output.



(b) Prior conditioning across goal offsets.

Figure 6: Action-prior ablations (four-task average success, %): (a) CEM integration and action chunks at $H = 25$; (b) final-goal versus local-subgoal conditioning across goal offsets.

macro-average at every tested offset, supporting a division of labor: the Action Prior maintains global task direction, while LatentPathFlow supplies the local target for rollout scoring.

Additional experiments on local-horizon sensitivity, test-time planning settings, and runtime are presented in App. E.

5 Related Work

Long-Horizon Planning with Latent World Models. Long-horizon visual planning often reduces effective planning depth through intermediate goals or temporal abstraction. Hierarchical Visual Foresight (Nair & Finn, 2020), KeyIn (Pertsch et al., 2020b), and GCP (Pertsch et al., 2020a) decompose distant goals into images, keyframes, or latent waypoints; Director (Hafner et al., 2022), HWM (Zhang et al., 2026), VLWM (Du et al., 2026), and FF-JEPA (Masip et al., 2026) use behavioral hierarchies, multi-scale dynamics, variable temporal abstraction, or latent-subgoal prediction to shorten planning depth. SAGE (Cheng et al., 2026) is the closest prior work: with a frozen world model, it predicts duration-conditioned endpoint subgoals and generates full action options for terminal-state CEM refinement. However, LeWM++ instead trains a flow over temporally aligned latent paths, using only the sampled endpoint as the local target. Its advantage-weighted action-chunk prior conditions on the final goal, while the generated subgoal enters rollout scoring through Min-over-Horizon. Thus, beyond the shared subgoal-and-proposal template, the methods differ in target modeling, proposal training and conditioning, rollout scoring.

Learned Action Guidance and Planning Objectives. Offline trajectories can be used to learn policies, value functions, or latent plans (Ghosh et al., 2021; Kostrikov et al., 2022; Lynch et al., 2020; Yuan et al., 2022; 2025). In model-based control, TD-MPC2 (Hansen et al., 2024) uses policy priors and terminal values to guide latent MPC; PRISM (Wang et al., 2026a) incorporates a learned action prior into MPPI sampling; and PiJEPA (Chahe & Zhou, 2026) initializes JEPA-based planning with an instruction-conditioned policy. GC-IDM (Nguyen et al., 2026) and INTACT (Sun et al., 2026) reduce test-time search by predicting actions directly. RC-aux (Li et al., 2026b) uses reachability supervision to improve latent representations and planning signals. LeWM++ uses value estimates to learn a final goal-conditioned action-chunk prior from offline data, then uses that prior to guide CEM search.

6 Conclusion

We introduce LeWM++, a planning-interface framework built on a frozen latent world model and guided by a unifying principle: use the final goal to preserve global intent and a local target to drive short-horizon planning. LatentPathFlow predicts a latent path supervised by offline transitions and uses its endpoint as the local target. The Action Chunk Prior then guides planning with goal-conditioned action proposals, while Min-over-Horizon scores predicted rollouts by their closest approach to that target. Across 16 benchmark tasks and dataset settings, LeWM++ scales favorably with horizon on the LeWM Control Suite and leads the compared model-based planners on eight visual OGBench datasets while remaining competitive with model-free HIQL. These results highlight planning-interface design alongside predictive dynamics. Limitations remain: evaluation is still dominated by simulation, a fixed local horizon is not uniformly optimal across all goal offsets, and very long-horizon compositional tasks remain challenging. Future work could use epistemic uncertainty to adaptively choose the local horizon and extend the framework to open-vocabulary, language-conditioned control in real physical environments (Wang et al., 2026b; Yuan et al., 2026).

AI USE STATEMENT

We used generative AI tools to edit the manuscript for clarity and to check the presentation and interpretation of experimental results. We did not use them to design the method or experiments, implement the method, generate data, or produce experimental measurements. The authors take responsibility for the final text, claims, and artifacts.

ETHICS STATEMENT

This study does not raise ethics-related concerns. It uses public simulation benchmarks and conducts real-robot object manipulation in controlled environments. Real-robot execution enforces workspace and action-magnitude limits, action clipping, and an emergency-stop mechanism; any trial that triggers a safety boundary is stopped or counted as a failure under the predefined protocol.

REPRODUCIBILITY STATEMENT

The method and main evaluation are described in Secs. 3 and 4. Apps. A to C provide training settings, full planning pseudocode, and task and evaluation protocols. Apps. D and E report per-task results, ablations, sensitivity studies, and runtime measurements. We report mean and standard deviation across three runs where applicable. We commit to open-sourcing the code.

REFERENCES

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Niccolò Caselli, Salvatore Lo Sardo, Francesco Massafra, Ippokratis Pantelidis, Samuele Punzo, and Sathya Kamesh Bhethanabhotla. Mind the gap: Promises and pitfalls of hierarchical planning in leworldmodel. *arXiv preprint arXiv:2607.12547*, 2026.

Amirhosein Chahe and Lifeng Zhou. Policy-guided world model planning for language-conditioned visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 834–841, June 2026.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning, 2026.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Rohan Chitnis, Yingchen Xu, Bobak Hashemi, Lucas Lehnert, Urun Dogan, Zheqing Zhu, and Olivier Delalleau. IQL-TD-MPC: Implicit q-learning for hierarchical model predictive control. *arXiv preprint arXiv:2306.00867*, 2023. doi: 10.48550/arXiv.2306.00867. URL https://arxiv.org/abs/2306.00867.

Tianqi Du, Qi Zhang, Yifei Wang, and Yisen Wang. Beyond the next step: Variable-length latent world models for long-horizon planning, 2026.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

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*, 2024.

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

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026a.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: RC-aux for latent world models, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2020.

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.

Sergi Masip, Jonathan Swinnen, Yutong Hu, Renaud Detry, and Tinne Tuytelaars. FF-JEPA: Long-horizon planning in world models with latent planners, 2026.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1909.05829.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models, 2026.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL http://arxiv.org/abs/1910.00177.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. In *Advances in Neural Information Processing Systems*, volume 33, 2020a.

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Konstantinos Derpanis, Kostas Daniilidis, Joseph Lim, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 969–979. PMLR, 2020b.

Oleh Rybkin, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine. Model-based reinforcement learning via latent-space collocation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9190–9201. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/rybkin21b.html.

Junhan Sun, Hao Zhao, and Guofeng Zhang. INTACT: Isomorphic intent-to-action learning for search-free world models, 2026.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided Imagination Sampling in world Models, 2026a.

Ziyi Wang, Xumin Yu, Yongming Rao, Yonggen Ling, Yunheng Li, Oran Wang, Mingqi Gao, Yuchen Zhou, Yves Liang, Zuyan Liu, et al. Hy-embodied-vlm-1.0: Efficient physical-world agents. *arXiv preprint arXiv:2607.12894*, 2026b.

Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning, 2025.

Yifu Yuan, Jianye Hao, Fei Ni, Yao Mu, Yan Zheng, Yujing Hu, Jinyi Liu, Yingfeng Chen, and Changjie Fan. Euclid: Towards efficient unsupervised reinforcement learning with multi-choice dynamics model. *arXiv preprint arXiv:2210.00498*, 2022.

Yifu Yuan, Hongyao Tang, Cong Wang, Yan Zheng, and Jianye Hao. Ed2: environment dynamics decomposition world models for continuous control. *Visual Intelligence*, 3(1):23, 2025.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, Hongyao Tang, and Jianye Hao. Embodied-r1: Reinforced embodied reasoning for general robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 8984–9009, 2026.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

A Training and Planning Details

We provide the training details of the LeWM backbone and the two learned planning interfaces, followed by the planning optimization procedure.

A.1 Shared Setup

Let H denote the final-goal offset in environment steps, c the action-chunk length, and k the local planning horizon. The planner optimizes $P = k/c$ chunk transitions. Each Control Suite environment or OGBench task has separate frozen LeWM parameters (E_ξ, F_ω) and separate interface modules.

A.2 LeWM Training

We reproduce LeWM in JAX and train a separate model for each Control Suite environment and visual OGBench dataset. We adopt the IMPALA-small CNN visual encoder used in OGBench (Park et al., 2025) and jointly optimize it with the action-conditioned autoregressive predictor using latent prediction MSE plus SIGReg with weight 0.09. The LeWM Control Suite uses 224×224 inputs, whereas visual OGBench uses 64×64 inputs. All downstream experiments use the epoch-10 checkpoints; the remaining settings are listed in Tab. 6.

Table 6: LeWM training hyperparameters used for the frozen backbones.

Hyperparameter	Value	Hyperparameter	Value
Visual encoder	IMPALA-small	Latent dimension	192
Input resolution	224 (Suite) / 64 (OGBench)	History frames	3
Frame skip	5	Predictor depth	6
Projection MLPs / normalization	$192 \to 2048 \to 192$ / BatchNorm	Predictor / embedding dropout	0.1 / 0
Prediction loss	MSE	SIGReg weight	0.09
SIGReg knots / projections	17 / 1,024	Training epochs	10
Batch size	128	Gradient-norm clip	1
Optimizer	AdamW	Peak learning rate	5×10^{-5}

A.3 LatentPathFlow Training

Goal and path sampling. For offline trajectories $\mathcal{D} = \{\tau_n\}_{n=1}^N$, where T_n is the final observation index of trajectory n, we construct training pairs by sampling a valid anchor and a later goal from the same trajectory. The future goal is sampled uniformly from the valid suffix after the anchor:

$$(n,t) \sim \mathrm{Unif}(\{(n,t) : 1 \le n \le N,\ 0 \le t < T_n\}), \qquad t_g \mid (n,t) \sim \mathrm{Unif}\{t+1, \dots, T_n\}. \quad (10)$$

The frozen encoder produces $z_s^n = E_\xi(o_s^n)$. With $P = k/c$ path tokens, the goal latent and supervised path are

$$z_g = E_\xi(o_{t_g}^n), \qquad \mathbf{z}^*_{t \rightsquigarrow t+k} = \left(z^n_{\min(t+c,t_g)}, z^n_{\min(t+2c,t_g)}, \dots, z^n_{\min(t+Pc,t_g)}\right). \quad (11)$$

If the goal precedes a nominal waypoint, that waypoint and all later tokens are clipped to t_g.

Architecture. LatentPathFlow parameterizes the conditional vector field v_η with a path-token Transformer. Each noisy 192-dimensional future latent is projected to a 512-dimensional token and augmented with its temporal-position embedding. The $\ell = 3$ latent-history states, final-goal latent z_g, and flow time form a global condition injected into all four Transformer blocks through adaptive layer normalization. Each block uses eight-head self-attention and a 2,048-dimensional feed-forward layer; a velocity head maps the output tokens back to the latent dimension.

Flow-matching objective and optimization. For Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ with the same shape as $\mathbf{z}^*_{t \rightsquigarrow t+k}$ and flow time $u \sim \mathcal{U}(0,1)$, we construct the linear interpolation $x_u = (1-u)\epsilon + u\mathbf{z}^*_{t \rightsquigarrow t+k}$ and minimize

$$\mathcal{L}_{\mathrm{LPF}}(\eta) = \mathbb{E}_{\mathcal{D},\epsilon,u}\left[\left\| v_\eta(x_u, u \mid z_{t-\ell+1:t}, z_g) - \left(\mathbf{z}^*_{t \rightsquigarrow t+k} - \epsilon\right)\right\|_F^2\right]. \quad (12)$$

Along each sampled linear path, the network is supervised by the constant target velocity $\mathbf{z}^*_{t\rightsquigarrow t+k} - \epsilon$; the predicted vector field still depends on x_u, u, and the conditioning variables. This trains the complete latent path rather than only its endpoint. The LeWM encoder remains frozen throughout training. We maintain an exponential moving average of η for validation and inference; at inference, 16 Euler steps integrate the EMA vector field from Gaussian noise. Tab. 7 summarizes the architecture and optimization settings.

Table 7: LatentPathFlow architecture and training hyperparameters.

Hyperparameter	Value	Hyperparameter	Value
Training steps	200,000	Batch size	1,024
Peak learning rate	10^{-4}	Final learning rate	10^{-5}
Warmup steps	5,000	Weight decay	10^{-4}
Gradient-norm clip	1	EMA decay	0.9999
Optimizer	AdamW	Schedule	warmup + cosine
Model width	512	Transformer depth	4
Attention heads	8	Feed-forward width	2,048

A.4 Action Chunk Prior Training

Samples and goal relabeling. We sample anchors uniformly from trajectory positions whose next c actions form a complete chunk. Each sample contains the frozen LeWM latents (z_t, z_{t+c}), the flattened action chunk $a_t^{(c)}$, and a relabeled goal g. Following standard goal relabeling (Ghosh et al., 2021), goals are drawn from observations in the offline dataset, with future observations from the same trajectory providing goal-directed supervision.

Twin critics and masked chunk backup. Following expectile-based offline value learning (Kostrikov et al., 2022) and fixed-length action chunking (Li et al., 2026a), we train one goal-conditioned value function V_ψ and two chunk critics $Q_{\theta_1}, Q_{\theta_2}$ on the frozen latents. In the expectile objective of Eq. (4), the regression target is the conservative target-critic estimate

$$Q_{\bar{\theta},\min}(z_t, a_t^{(c)}, z_g) = \min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(z_t, a_t^{(c)}, z_g), \tag{13}$$

where the target parameters are updated by $\bar{\theta} \leftarrow (1-\rho)\bar{\theta} + \rho\theta$.

For a relabeled goal g, let $R_t^{(c)}(g)$ denote the discounted reward accumulated over the chunk, truncated once the goal is reached. The continuation mask $m_t^{(c)}(g)$ is zero after goal attainment and one otherwise. Both critics use the shared target

$$y_t = R_t^{(c)}(g) + \gamma^c m_t^{(c)}(g) V_\psi(z_{t+c}, z_g), \qquad \mathcal{L}_Q = \mathbb{E}_{\mathcal{D}}\left[\sum_{i=1}^{2}\left(Q_{\theta_i}(z_t, a_t^{(c)}, z_g) - y_t\right)^2\right]. \tag{14}$$

Thus, reaching g within the chunk terminates the return and disables bootstrapping; otherwise, the target bootstraps from $V_\psi(z_{t+c}, z_g)$.

Table 8: Action Chunk Prior hyperparameters. Shared settings are reported once; benchmark-specific settings are labeled explicitly.

Hyperparameter	Value
Chunk length c	5
Number of critics	2
MLP hidden width / depth	512 / 3
Fixed actor std. σ_π	1
Goal rewards $r_{\text{goal}}/r_{\text{step}}$	$0/-1$
Discount γ	0.99
Expectile τ	0.9
Target update rate ρ	0.005
AWR inverse temperature β	3.0
Maximum AWR weight $w_{\max}$	100
Optimizer	Adam
Learning rate	3×10^{-4}
Training steps	Visual OGBench: 500k LeWM Control Suite: 100k
Batch size	Visual OGBench: 512 LeWM Control Suite: 256

Advantage-weighted actor. The actor is a diagonal-Gaussian policy whose multilayer perceptron predicts the mean of the complete action chunk; its diagonal standard deviation is fixed to σ_π. For a relabeled goal g, we form the conservative advantage and clipped regression weight (Peng et al., 2019),

$$A_t = \min_{i\in\{1,2\}} Q_{\theta_i}(z_t, a_t^{(c)}, z_g) - V_\psi(z_t, z_g), \qquad w_t = \min\{\exp(\beta A_t), w_{\max}\}, \tag{15}$$

and minimize $\mathcal{L}_\pi = -\mathbb{E}_{\mathcal{D}}[w_t \log \pi_\phi(a_t^{(c)} \mid z_t, z_g)]$. Tab. 8 lists the architecture and all numerical training settings.

B Complete LeWM++ Planning Algorithm

Alg. 1 gives the unified test-time procedure. At each planning cycle, the deterministic policy mode initializes the first block of the initial CEM mean, while the remaining blocks are warm-started by shifting the unused suffix of the previous plan and zero-padding the tail. At each refinement iteration, K_π candidates additionally draw their first action chunk from the stochastic Action Prior; their remaining blocks, together with all blocks of the other $N - K_\pi$ candidates, are sampled from the current CEM Gaussian. Setting $K_\pi = 0$ recovers mean-only guidance. All candidates are rolled out and scored using the same world model and Min-over-Horizon objective. The current state is excluded from MoH, so a zero-action candidate cannot receive a low cost before any predicted transition is made. Benchmark-specific settings are given in App. C.

Algorithm 1 LeWM++ Closed-Loop Latent-Space Planning

Require: Goal image g; frozen E_ξ, F_ω; LatentPathFlow p_η; Action Prior π_ϕ with mode μ_ϕ; chunk length c; local horizon k; CEM population N, prior-guided count K_π, elites M, iterations L

$z_g \leftarrow E_\xi(g)$; $P \leftarrow k/c$; previous suffix $\leftarrow \varnothing$
for $t = 0, c, 2c, \ldots$ until success or budget exhaustion **do**
  Encode recent history $z_{t-\ell+1:t} \leftarrow E_\xi(o_{t-\ell+1:t})$
  Generate $\tilde{\mathbf{z}}_t \sim p_\eta(\cdot \mid z_{t-\ell+1:t}, z_g)$; set $z_t^{\mathrm{sg}} \leftarrow \tilde{z}_{t+k}$
  $a_{\pi,t}^{(c)} \leftarrow \mu_\phi(z_t, z_g)$
  Shift the previous suffix and zero-pad it to form $\boldsymbol{\mu}_t^{(0)}$
  Replace the first block of $\boldsymbol{\mu}_t^{(0)}$ with $a_{\pi,t}^{(c)}$; set $\boldsymbol{\sigma}_t^{(0)} \leftarrow \mathbf{1}$
  for $r = 0, \ldots, L-1$ **do**
    for $i = 1, \ldots, N$ **do**
      Sample $\mathbf{A}_i^{(r)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(r)}, \mathrm{diag}((\boldsymbol{\sigma}_t^{(r)})^2))$
      if $i \le K_\pi$ **then**
        Sample $a_{i,t}^{(c)} \sim \pi_\phi(\cdot \mid z_t, z_g)$ and replace the first block of $\mathbf{A}_i^{(r)}$
      end if
      Set $\hat{z}_{i,t} \leftarrow z_t$
      for $j = 0, \ldots, P-1$ **do**
        $\hat{z}_{i,t+(j+1)c} \leftarrow F_\omega^{(c)}(\hat{z}_{i,t+jc}, a_{i,t+jc}^{(c)})$
      end for
      $J_i \leftarrow \min_{j \in \{1,\ldots,P\}} \|\hat{z}_{i,t+jc} - z_t^{\mathrm{sg}}\|_2^2$
    end for
    Select the M lowest-cost sequences and refit $\boldsymbol{\mu}_t^{(r+1)}, \boldsymbol{\sigma}_t^{(r+1)}$
  end for
  Let $\mathbf{A}_t^\star$ denote the resulting plan after L refinements
  Execute the first action chunk of $\mathbf{A}_t^\star$
  Store its unused suffix for warm starting; observe o_{t+c}
end for

C Evaluation Protocols

C.1 LeWM Control Suite

Tasks. The suite contains four image-goal control tasks (Maes et al., 2026). **TwoRoom** is continuous 2D navigation: a point agent must pass through the single opening in the separating wall and reach the location shown in the goal image. **Reacher** controls a two-joint planar arm whose configuration must align with the target configuration. **PushT** requires a planar pusher to translate and rotate a T-shaped block to match the goal image. **Cube** requires a 3D robot end effector to pick up a cube and place it at the target location.

Goal-offset and evaluation protocol. We evaluate each task at goal offsets $H \in \{25, 50, 75, 100\}$, denoted H25, H50, H75, and H100. An episode starts from a state selected from an offline trajectory, and its goal image is the observation H environment steps later in the same trajectory. The maximum episode length is $B = 2H$, giving budgets of 50, 100, 150, and 200 environment steps. Each setting is evaluated over 50 episodes per task for each of three seeds. We report task success rates and their unweighted four-task macro-average.

Planning settings. LeWM++ uses three observation-history frames, action chunks of $c = 5$ environment steps, a local target horizon of $k = 10$, and $P = 2$ latent transitions. Each planning call draws one LatentPathFlow sample with 16 Euler steps. CEM uses 300 candidates, 30 elites, five refinement iterations, and initial standard deviation $\sigma_0 = 1$. The Control Suite uses mean-only action-prior guidance with $K_\pi = 0$: the prior mode initializes the first block of the initial CEM mean, and no stochastic prior candidates are injected during refinement. The controller executes the first chunk of the final optimized mean before replanning. All candidates are scored in the frozen LeWM latent space without image reconstruction.

C.2 VISUAL OGBENCH

Tasks and datasets. Visual OGBench evaluates pixel-based image-goal control on single-, double-, and triple-cube manipulation and scene rearrangement (Park et al., 2025). Pairing each environment with `play` and `noisy` offline data yields eight datasets. Cube goals involve moving, stacking, swapping, or permuting blocks. Scene rearrangement combines a cube, window, drawer, and two lock buttons, creating prerequisite-dependent plans such as unlocking and opening the drawer before placing the cube. The `play` datasets use non-Markovian expert policies with temporally correlated noise, whereas `noisy` datasets use Markovian experts with larger independent Gaussian noise and broader state coverage.

Evaluation protocol. Each dataset contains five official state–goal tasks, and resetting a task samples the benchmark's small perturbations around its prescribed initial and goal states. For each of three seeds, we run 50 episodes per task under the registered episode horizon: 200 environment steps for single-cube, 500 for double-cube, 1,000 for triple-cube, and 750 for scene rearrangement. An episode is successful if the official goal predicate is satisfied before this horizon. We first average the 50 binary outcomes for each task and then average the five task success rates to obtain one dataset score. Thus, each dataset contains $5 \times 50 = 250$ episodes per seed (750 across three seeds), and the complete eight-dataset evaluation contains 2,000 episodes per seed (6,000 in total). Dataset entries report the mean and standard deviation across the three seeds; the final score is the unweighted macro-average of the eight dataset means.

Planning settings. Visual OGBench uses the same three-frame history, $c = 5$, $k = 10$, $P = 2$, and one 16-step Euler LatentPathFlow sample as the Control Suite. CEM uses $N = 300$ candidates, 30 elites, and five refinement iterations. At each iteration, the first action chunk of $K_\pi = 200$ candidates is sampled from the stochastic final-goal-conditioned Action Chunk Prior, while their remaining chunks are sampled from the current CEM Gaussian. The other $N - K_\pi = 100$ candidates are sampled entirely from the Gaussian. This stronger action-prior guidance keeps a substantial portion of the search near the offline action support, thereby limiting excessive out-of-distribution action search and the associated exploitation of world-model errors, while retaining Gaussian candidates for model-based local correction. After the fifth iteration, the controller executes the first chunk of the best-scoring candidate in the final population and replans from the next observation.

Goal-conditioned LDP adaptation. We include Latent Diffusion Planning (LDP) (Xie et al., 2025) as a complementary model-based latent-planning baseline and build our adaptation on its official implementation.[1] Whereas LeWM++ optimizes actions through an action-conditioned joint-embedding predictor, LDP directly generates a future latent trajectory and recovers actions between adjacent generated latents with an inverse-dynamics model. Its latent space is learned by a convolutional VAE with pixel-reconstruction and KL objectives, so LDP represents a reconstruction-trained alternative to LeWM's reconstruction-free latent planning. Because the two methods also differ in their encoder objectives, trajectory models, and action-generation mechanisms, this comparison evaluates complete planning pipelines rather than isolating the effect of reconstruction. The released LDP model targets imitation learning and conditions trajectory generation on only the current observation. For image-goal OGBench, we additionally condition its trajectory diffusion model on $[z_t, z_g]$ while retaining the original goal-free inverse-dynamics interface. We train a separate VAE and LDP model for each dataset. Training windows remain within one trajectory segment, use its terminal latent as the goal, and terminal-pad at segment boundaries. At evaluation time, LDP predicts eight latent transitions, executes the actions recovered from the first four, and then replans from the new image.

[1] https://github.com/amberxie88/latent_diffusion_planning

Table 9: Hyperparameters of the goal-conditioned LDP adaptation on visual OGBench.

Hyperparameter	Value	Hyperparameter	Value
Input resolution	64×64 RGB	VAE training steps	300,000
VAE batch size	128	VAE KL weight	10^{-5}
VAE EMA decay	0.99	LDP training steps	500,000
LDP batch size	128	Predicted latent horizon	8
Executed action horizon	4	DDPM denoising steps	100
Peak / final learning rate	$10^{-4}/10^{-6}$	Warmup steps	1,000
Planner U-Net widths	256, 512, 1,024	Train/validation split	95/5 episodes
Evaluation episodes	50 per task	Official tasks per dataset	5

C.3 DINO-WM Suite

Our implementation is based on the authors' official DINO-WM codebase,[2] and the evaluation follows the Rope and Granular environments of Zhou et al. (2025). **Rope** uses an XArm in NVIDIA Flex to move a soft rope from an arbitrary initial configuration to a target shape and pose. **Granular** uses the same robot and simulator to arrange approximately one hundred particles into a target shape. These environments replace rigid-body dynamics with deformable-object interaction and use a different frozen latent world model. Each task is evaluated on 10 held-out initial–goal instances. Performance is the Chamfer distance between the achieved and target object configurations, reported as the mean and standard deviation across instances; lower is better.

For DINO-WM++, we keep the released DINO encoder and dynamics model frozen and instantiate the complete LeWM++ planning interface in their latent space. We encode the offline trajectories with the DINO encoder and train LatentPathFlow to generate a trajectory-aligned latent path conditioned on the recent latent history and final DINO goal embedding; its endpoint supplies the local rollout target. DINO-WM++ uses mean-only action-prior guidance with $K_\pi = 0$: the mode of the goal-conditioned Action Chunk Prior initializes the first block of the initial CEM mean, and no stochastic prior candidates are injected during refinement. Min-over-Horizon evaluates the closest predicted approach to the generated local target. CEM then refines the complete action sequence through the frozen DINO dynamics before the controller executes the first action chunk and replans.

C.4 Real-World Image-Goal Tasks

Hardware and observation interface. We use an ARX Lift2s mobile dual-arm robot and an XArm6 manipulator. The ARX Lift2s uses its built-in egocentric RGB camera and is controlled in joint action space, while the XArm6 uses a fixed third-person camera as shown in the task figures and is controlled via normalized Cartesian end-effector increments; workspace and per-step safety limits are applied before execution. The visual input consists of the latest three RGB frames and one goal image. In **Colored-Tray Placement**, one cube is initialized outside three visually distinct receptacles. The goal image specifies the target receptacle, and the robot must transport and release the cube. The 45 trials balance the three target receptacles and cover diverse cube start positions and layouts. In **Multi-Cube Rearrangement**, the goal image specifies a target configuration for pushing one, two, or three cubes into goal regions. The 45 trials contain 15 conditions at each cube count and cover different initial layouts and goal arrangements. A dropped object, contact outside the safety envelope, or exhaustion of the 300-step budget is counted as failure.

Planning settings. GC-DP predicts a 10-step action sequence with 16 DDIM denoising steps, executes the first five actions, and replans from the newest image. LeWM++ uses the same three-frame observation history, action chunks of length $c = 5$ environment steps, and a local target horizon of $k = 10$. At each CEM refinement iteration, 256 of the 280 candidates draw their first action chunk from stochastic samples of the final-goal-conditioned Action Chunk Prior; this constraint reduces excessive on-hardware exploration caused by model-based planning. LeWM++ executes five actions and replans, matching the receding-horizon cadence of GC-DP. All action sequences pass through the same clipping, gripper thresholding, and emergency-stop layer.

[2] https://github.com/gaoyuezhou/dino_wm

D Complete Results and Additional Experiments

D.1 LeWM Control Suite Horizon Scaling

Tab. 10 reports the complete results, the five-iteration LeWM compute diagnostic, and matched component ablations at all four goal offsets. The five-iteration LeWM macro-average is 77.7 ± 4.5, 56.8 ± 1.5, 52.3 ± 1.5, and 43.8 ± 2.8 for $H = 25, 50, 75, 100$, respectively.

Table 10: Complete success rates (%) and component ablations across goal offsets. LeWM (iter.=5) is the reduced-iteration compute diagnostic; rows beginning with "w/o" are LeWM++ ablations. Task values are mean ± standard deviation over three seeds; the Average column reports the macro-average mean.

H	Method	TwoRoom	Reacher	PushT	Cube	Average
25	LeWM	97.30±1.20	90.00±2.00	80.00±6.00	72.70±7.60	85.00
	LeWM (iter.=5)	90.7±3.1	70.7±4.2	78.0±6.0	71.3±10.3	77.7
	PRISM	100.00±0.00	95.33±1.15	92.70±5.00	62.70±8.50	87.68
	SAGE	100.00±0.00	96.70±2.50	85.30±3.10	92.00±3.50	93.50
	w/o LatentPathFlow	98.67±1.89	100.00±0.00	83.33±4.11	90.00±2.83	93.00
	w/o Action Prior	100.00±0.00	88.70±3.40	92.70±0.90	86.70±1.90	92.00
	w/o MoH	100.00±0.00	98.70±1.90	85.30±3.40	92.00±3.30	94.00
	LeWM++	100.00±0.00	96.70±2.50	96.00±1.60	95.30±0.90	97.00
50	LeWM	84.70±6.10	89.30±1.20	34.70±10.30	57.30±3.10	66.50
	LeWM (iter.=5)	62.7±16.2	86.0±7.2	23.3±11.4	55.3±6.4	56.8
	PRISM	100.00±0.00	92.00±2.00	54.70±12.50	33.30±7.50	70.00
	SAGE	99.30±1.20	85.30±7.60	64.00±2.00	88.00±4.00	84.15
	w/o LatentPathFlow	99.33±0.94	99.33±0.94	52.67±5.25	79.33±4.11	82.67
	w/o Action Prior	100.00±0.00	94.00±3.46	76.00±6.93	61.33±1.15	82.83
	w/o MoH	100.00±0.00	97.33±3.06	68.00±4.00	68.67±6.11	83.50
	LeWM++	100.00±0.00	96.70±1.90	83.30±5.00	86.70±4.70	91.68
75	LeWM	76.00±5.30	86.70±1.20	16.00±3.50	70.00±4.00	62.18
	LeWM (iter.=5)	44.0±8.7	88.7±5.8	6.7±1.2	70.0±2.0	52.3
	PRISM	80.67±5.73	85.33±3.06	26.70±9.50	50.70±7.50	60.85
	SAGE	99.30±1.20	95.30±2.20	56.00±5.70	67.30±8.10	79.48
	w/o LatentPathFlow	98.67±1.89	97.33±2.49	23.33±4.11	81.33±2.49	75.17
	w/o Action Prior	100.00±0.00	98.00±2.00	64.00±8.72	62.67±1.15	81.17
	w/o MoH	100.00±0.00	96.67±4.16	53.33±4.16	70.00±3.46	80.00
	LeWM++	100.00±0.00	97.30±2.50	71.30±5.70	82.00±2.80	87.65
100	LeWM	60.00±3.50	87.30±1.20	14.00±6.00	62.00±7.20	55.83
	LeWM (iter.=5)	17.3±1.2	90.0±2.0	5.3±1.2	62.7±10.1	43.8
	PRISM	65.33±4.99	80.00±4.00	19.30±12.00	52.70±10.50	54.33
	SAGE	96.70±0.90	98.70±2.30	49.30±4.20	78.70±1.20	80.85
	w/o LatentPathFlow	92.67±0.94	98.00±2.83	23.33±7.72	71.33±3.40	71.33
	w/o Action Prior	100.00±0.00	97.33±3.06	66.00±4.00	66.67±5.77	82.50
	w/o MoH	100.00±0.00	98.00±3.46	55.33±4.62	75.33±5.77	82.17
	LeWM++	100.00±0.00	98.00±2.80	71.30±0.90	83.30±6.60	88.15

D.2 Action-Consistency and Closed-Loop-Progress Diagnostics

The diagnostics in Tab. 11 measure the action consistency of planned rollouts and the real controller's closed-loop progress toward generated targets. At each replanning event, CEM produces a candidate sequence whose LeWM rollout approaches the target. An independently trained inverse-dynamics model maps each imagined transition back to an action block for comparison with the candidate actions. The progress metric instead uses observations from real execution.

For each task, we encode its offline trajectories with the frozen LeWM encoder and train a separate inverse-dynamics model I_ψ on real five-step transitions $(z_t, z_{t+c}, a_{t:t+c-1})$, with $c = 5$. We split episodes into 90% training and 10% validation sets and optimize I_ψ for 200k steps using conditional flow matching on normalized action chunks. After training, we freeze I_ψ and apply it to candidate rollouts from all target-generator variants.

At a replanning event, let z_t be the current latent, z_t^{sg} the generated subgoal, $\hat{z}_{t+jc}$ the imagined latent after block j, and I_ψ the frozen inverse-dynamics diagnostic described above. With normalized actions, we define

$$\mathrm{CycleMSE}_t = \frac{1}{P}\sum_{j=1}^{P} \mathrm{MSE}\big(I_\psi(\hat{z}_{t+(j-1)c}, \hat{z}_{t+jc}), a_{t+(j-1)c:t+jc-1}\big)\,, \tag{16}$$

where $\hat{z}_t = z_t$. A lower value indicates stronger agreement between the candidate action chunks and the inverse-dynamics predictions.

For closed-loop progress, let z_{t+u} be the latent encoding of the real observation after u environment steps. We compute

$$r_t = \frac{\min_{1\le u\le k}\|z_{t+u} - z_t^{\mathrm{sg}}\|_2^2}{\max(\|z_t - z_t^{\mathrm{sg}}\|_2^2, \epsilon)}, \qquad \mathrm{Progress@0.25} = \mathbb{E}_t[\mathbf{1}\{r_t \le 0.25\}]\,. \tag{17}$$

Progress@0.25 is the percentage of replanning events for which the real trajectory reduces its squared latent distance to the initially proposed target by at least 75% within k steps. Normalization makes the threshold relative to the initial target distance. Here $k = 10$ and $c = 5$, so the measurement includes the next replanning cycle after step 5.

Tab. 11 reports the resulting comparison. LatentPathFlow has the lowest CycleMSE and the highest Progress@0.25 among the three learned target generators, indicating stronger consistency with the learned action model and greater closed-loop progress toward the generated target.

Table 11: Action-consistency and closed-loop-progress diagnostics for learned target generators. Lower CycleMSE and higher Progress@0.25 are better.

Generator	CycleMSE ↓	Progress@0.25 ↑
MLP	0.2955	37.44
EndpointFlow	0.2382	68.48
LatentPathFlow	**0.1857**	**73.86**

D.3 OGBench Ablation Study

To understand the contribution of each planning interface on OGBench, we conduct matched ablations by removing LatentPathFlow (w/o Subgoal) or replacing Min-over-Horizon with terminal-only scoring (w/o MoH). Tab. 12 reports the per-dataset results.

Across the eight datasets, the full LeWM++ planner averages 50.62%, compared with 47.28% without LatentPathFlow and 47.20% with terminal-only scoring. Thus, the local target and MoH contribute 3.34 and 3.42 pp to the macro-average, respectively. Direct Action Prior execution averages 45.6% (Tab. 2), further indicating an aggregate benefit from planning over learned action proposals.

Table 12: Success rate (%) on eight visual OGBench datasets for LeWM++ ablations. Each entry reports mean ± standard deviation across three independent training seeds.

Dataset	w/o Subgoal	w/o MoH
`visual-cube-single-play-v0`	63.60 ± 4.23	65.40 ± 5.20
`visual-cube-double-play-v0`	29.60 ± 1.06	28.40 ± 1.20
`visual-cube-triple-play-v0`	17.33 ± 0.83	17.20 ± 0.40
`visual-scene-play-v0`	52.27 ± 1.80	53.20 ± 2.43
`visual-cube-single-noisy-v0`	92.93 ± 0.92	92.27 ± 1.01
`visual-cube-double-noisy-v0`	57.07 ± 1.15	51.60 ± 2.08
`visual-cube-triple-noisy-v0`	18.80 ± 0.69	19.23 ± 0.61
`visual-scene-noisy-v0`	46.67 ± 2.20	50.27 ± 0.46
Average	47.28	47.20

E Robustness and Efficiency

Local-Horizon Sensitivity. We vary the local subgoal horizon k to test sensitivity to its temporal scale. Across $k \in \{10, 15, 25\}$, the macro-average ranges by 3.34 points at $H = 50$ and 1.83 points

at $H = 75$. At $H = 25$, success decreases from 97.00% at $k = 10$ to 92.00% at $k = 15$ and 91.83% at $k = 25$. When $k = 25 = H$, the generated target spans the full goal offset rather than an intermediate horizon. It is still produced by LatentPathFlow, so this setting is distinct from the w/o-subgoal ablation. Across the tested settings, $k = 10$ is the most consistent default.

Table 13: Macro-average success (%) versus local subgoal horizon k.

Local horizon	$H = 25$	$H = 50$	$H = 75$
$k = 10$	**97.00**	**91.67**	87.67
$k = 15$	92.00	88.67	86.50
$k = 25$	91.83	88.33	**88.33**

Runtime. In the LeWM Control Suite at $H = 25$, LeWM++ takes 26.42 ms per planning call versus 128.04 ms for canonical LeWM with 30 CEM iterations, a 4.85× speedup (Tab. 14). Reducing LeWM to five CEM iterations brings its latency to 25.45 ms but yields only 77.7% macro-average success, compared with 97.0% for LeWM++ at 26.42 ms. Its per-task results at every goal offset appear in Tab. 10. Within LeWM++, 16-step LatentPathFlow sampling and one Action Prior pass take 5.69 ms and 2.52 ms, respectively; shorter local-target rollouts offset this overhead.

Table 14: Steady-state planning latency and four-task macro-average success at $H = 25$. Speedups are relative to canonical LeWM. LeWM (iter.=5) matches LeWM++'s CEM compute; component latencies are included in LeWM++. Latency ± indicates variation across four tasks.

Method	Configuration	Latency (ms/plan)	Speedup	Success (%)
LeWM	CEM iter. = 30	128.04±0.37	1.00×	85.0
LeWM (iter.=5)	CEM iter. = 5	25.45±0.68	5.03×	77.7
LeWM++	CEM iter. = 5	26.42±2.11	**4.85×**	**97.0**
LatentPathFlow	Euler, 16 NFEs	5.69±0.07	–	–
Action Prior	One actor pass	2.52±0.40	–	–

Test-Time Hyperparameter Sensitivity. We vary CEM iterations and LatentPathFlow integration steps independently around defaults of five and 16. Success is non-monotonic across goal offsets (Fig. 7); exact four-task macro-averages appear in Tab. 15.

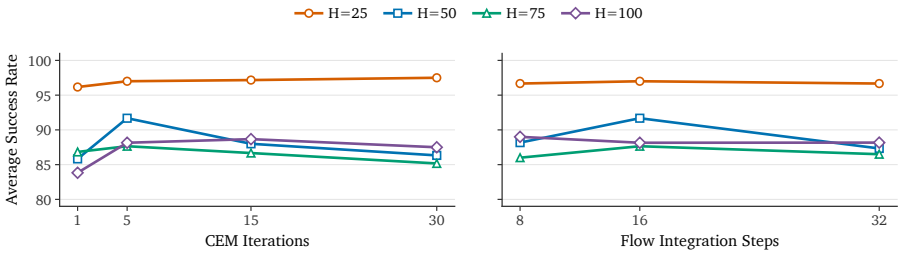


Figure 7: Sensitivity to two test-time planning hyperparameters. Left: CEM iterations with 16 Flow integration steps. Right: LatentPathFlow integration steps with five CEM iterations. Each curve is the four-task macro-average at one goal horizon.

Table 15: Macro-average success (%) in the CEM-iteration and Flow-integration-step sensitivity sweeps.

Hyperparameter	Setting	$H = 25$	$H = 50$	$H = 75$	$H = 100$
CEM iterations	1	96.17	85.83	86.83	83.83
	5	97.00	91.68	87.65	88.15
	15	97.17	88.00	86.67	88.67
	30	97.50	86.33	85.17	87.50
Flow steps	8	96.67	88.17	86.00	89.00
	16	97.00	91.68	87.65	88.15
	32	96.67	87.33	86.50	88.17