# How Long, Not How Close: A Learned Temporal Metric for Planning in Latent World Models

**Anonymous authors**
Paper under double-blind review

## Abstract

Latent world models plan by rolling a frozen predictor forward under candidate action sequences and ranking the candidates by the latent distance between their imagined end state and the goal. However, this ranking breaks down when the goal lies several plans away, because the latent distance measures how closely an end state resembles the goal rather than how far it remains from reaching it. To address this, we propose ***TEMPO***, a temporal-distance planning objective that leaves the world model untouched, learns only from the demonstrations already used to train it, and adds negligible cost to the planner's search. *TEMPO* learns a small map of the frozen latent in which the distance between two states of an episode reflects the number of environment steps between them, and blends this distance into the planner's cost. It requires no rewards, policies or success labels and, being a cost rather than a model, applies to frozen world models with one latent vector per state that plan by a latent distance. We evaluate *TEMPO* on eleven simulated environments (e.g., maze navigation, tabletop pushing, robotic arm control and three-dimensional manipulation) with the LeWM and PLDM planners. With a 0.13M-parameter map that adds at most 0.3% to a plan's arithmetic, *TEMPO* improves both planners at every goal distance, including the one-plan setting of their published evaluations, raises LeWM from 36% to 99% on TwoRoom three plans from the goal, and remains competitive on a broad range of 2D and 3D navigation, reaching and manipulation tasks.

## 1 Introduction

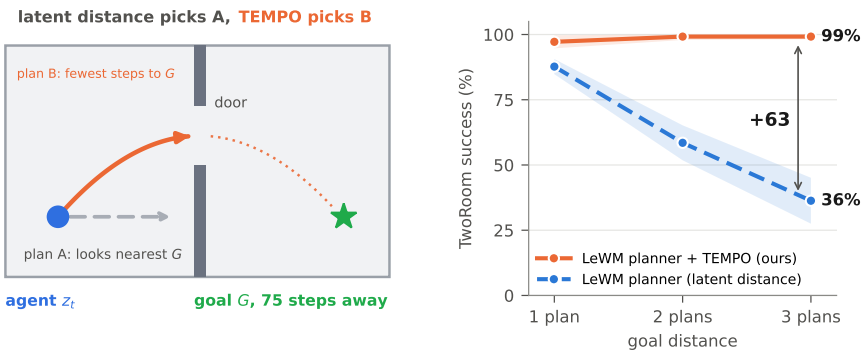


Figure 1: **How long, not how close.** Left: three plans from the goal, the latent distance prefers the plan whose end state looks most like the goal frame, toward the wall, whereas a distance that counts steps prefers the plan through the door. Right: TwoRoom success as the goal moves from one to three plans away.

A long-standing goal of learning-based control is an agent that plans directly from pixels, without hand-engineered state or a task-specific policy. World models offer a path toward this goal. By pre-



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.

dicting the consequences of its actions, an agent can search over them in imagination and act on the best sequence it finds. Joint-embedding predictive architectures (JEPA) have become the practical recipe for learning such models from video. An encoder maps each frame to a compact latent, a predictor rolls the latent forward under actions, and both are trained to predict the embedding of the next frame rather than to reconstruct pixels. Planning then amounts to model-predictive control in the latent space, which rolls the predictor out under candidate action sequences, keeps the one whose imagined end state is closest to the encoded goal, executes it, and replans from the next observation. Recent planners of this kind, such as PLDM and LeWM (Sobal et al., 2026; Maes et al., 2026b), keep the model frozen after training, replan in closed loop and learn no low-level policy. On the benchmarks these models are trained for, a single plan suffices to reach the goal, and LeWM's planner succeeds on $94\%$ of Push-T windows whose goal lies 25 steps ahead, $88\%$ on TwoRoom and $95\%$ on Reacher.

Yet the goals that matter in practice, such as reaching the far side of a room through a door or pushing an object to a distant target pose, lie well beyond a single plan, and the original evaluations of these planners stop one plan from the goal. When the goal lies several plans away, no single action sequence the planner can score reaches it. Three plans away, the same planner collapses to $22\%$ on Push-T and $36\%$ on TwoRoom, and every latent world model we test degrades in the same way (Figure 1). Such failures are usually attributed to compounding rollout error and a search space that grows with the horizon (Zhang et al., 2026), both of which afflict planners that imagine the entire horizon at once. The planners we study instead imagine a single plan, execute it and replan, reaching a distant goal through a sequence of short plans, and what breaks this sequence is the cost itself, since every short plan ends far from a goal 75 steps out and ranking end states by their resemblance to the goal frame reveals nothing about which plan leads there. The predictor is trained to be locally accurate, whereas the planner reads latent distance as how much of the task remains, and the two agree within one plan of the goal but diverge several plans away. Existing remedies each sacrifice part of what makes the frozen planner attractive. Retraining or fine-tuning the world model with a temporal or reachability objective (Li et al., 2026; Bai & Xiong, 2026) changes the model, which is not possible when it is frozen, and hierarchical planners (Zhang et al., 2026; Cheng et al., 2026; Caselli et al., 2026; Hafner et al., 2022) insert a second decision process between the goal and the planner and search over subgoals or macro-actions.

We introduce ***TEMPO*** (***TEM****poral-distance* ***P****lanning* ***O****bjective*), a learned temporal distance for the planner's cost that leaves the world model untouched, draws only on the demonstrations already used to train it, and adds negligible cost to the planner's search. A small map $F_\phi$ embeds the frozen latent so that the distance between two states of an expert episode reflects the number of environment steps between them. It is trained by regression on pairs of expert frames, requires nothing beyond their frame indices, the time step at which each frame was recorded, and continues to learn online from the episodes the planner itself collects. Blended with the latent distance rather than substituted for it, $F_\phi$ improves planning at every goal distance, and on the standard tasks most when the goal is far, raising LeWM from 36 to $99\%$ on TwoRoom and from 90 to $98\%$ on Reacher three plans from the goal and from 22 to $28\%$ on Push-T. The two distances are complementary. The latent distance captures the fine positioning that manipulation requires, while $F_\phi$ measures *how far from the goal in time* a state lies, and combining the two yields the strongest planner overall.

We evaluate *TEMPO* on eleven environments spanning maze navigation, tabletop pushing, robotic arm control and three-dimensional manipulation, on top of the LeWM and PLDM planners and on identical windows from one to three plans away from the goal. Our experiments reveal the following findings:

- *TEMPO* enables a frozen short-horizon planner to reach goals several plans away. Three plans from the goal it improves the LeWM planner by 16 points on average across our standard tasks, and LeWM with *TEMPO* outperforms the PLDM planner on every standard task (Section 4.3).
- *TEMPO* improves both planners at every goal distance, including the one-plan setting of their original evaluations (Section 4.3).
- *TEMPO* can be trained across environment variations within a task family (e.g., different wall layouts for navigation or different block shapes for pushing) and, without retraining, achieves higher success than the planner under layout shifts (unseen wall layouts and door

positions) and shape shifts (unseen block shapes), tripling LeWM's success with a moved door three plans from the goal (Section 4.4).

- *TEMPO* is computationally lightweight. It requires the same number of predictor rollouts as the original planner, introduces no second model or level of search, and increases the cost of each plan by at most $0.3\%$ without task-specific tuning (Appendix B).

## 2 Related work

**JEPA world models.** World models learn the environment's dynamics from data so that an agent can reason about future states in imagination (Ha & Schmidhuber, 2018). Generative world models simulate future observations in pixel space (Hafner et al., 2019b;a; 2025; Micheli et al., 2022; Alonso et al., 2024). Joint-embedding predictive architectures (JEPA) instead predict in a compact latent space (LeCun et al., 2022), first as self-supervised representation learners (I-JEPA, V-JEPA (Assran et al., 2023; Bardes et al., 2024)) and, trained on reward-free offline trajectories, as action-conditioned world models that plan zero-shot for goals given as images. PLDM and LeWM encode each frame as a single latent vector. PLDM learns its encoder and predictor jointly from pixels with a VICReg-style regulariser (Sobal et al., 2026; Bardes et al., 2021), and LeWM trains a ViT encoder end to end with LeJEPA's single Gaussian regulariser and plans with CEM over its 192-dimensional class token (Maes et al., 2026b; Balestriero & LeCun, 2025). DINO-WM instead predicts a grid of frozen DINOv2 patch features per frame and so cannot collapse (Zhou et al., 2024; Oquab et al., 2023), but its pretrained encoder and many tokens per frame make it larger and much slower to train and plan than models with a single latent vector (Maes et al., 2026b), and its cost also reads the agent's proprioceptive state, so we focus here on single-latent world models.

**Planning with latent dynamics.** Dreamer and TD-MPC learn a policy or value in imagination and need rewards (Hafner et al., 2019a; Hansen et al., 2022; 2024), while offline goal-conditioned RL learns a policy or value directly (Ghosh et al., 2019; Kostrikov et al., 2021; Park et al., 2023). Reward-free JEPA planners instead query the frozen dynamics by sampling-based trajectory optimisation (Rubinstein & Kroese, 2004; Pinneri et al., 2021), executing the candidate whose imagined end state is closest to the encoded goal (Zhou et al., 2024; Sobal et al., 2026; Maes et al., 2026b). Their published evaluations (Maes et al., 2026b; Zhou et al., 2024; Park et al., 2025) stop one plan from the goal, whereas we evaluate several plans out, with every arm on identical windows, and add layout, shape and visual shifts at test time (Wang et al., 2026).

**Long-horizon planning and latent distances.** Learned world models are known to struggle over long horizons, as rollout errors compound and the search space grows with the horizon (Zhang et al., 2026). Hierarchical planners address this with coarse models or subgoal generators (Zhang et al., 2026; Cheng et al., 2026; Caselli et al., 2026; Pertsch et al., 2020; Nasiriany et al., 2019; Hafner et al., 2022; Park et al., 2023), at the price of a second model or level of search, while a closed-loop planner that replans after each short plan avoids both issues yet still fails several plans out, because its distance ranks resemblance to the goal rather than distance from it. Another line of work therefore makes latent distances track progress. RC-aux (Li et al., 2026) adds a reachability-correction objective to LeWM, TD-JEPA (Bai & Xiong, 2026) retrains the encoder so that its latent distance reflects temporal progress, Hilbert representations (Park et al., 2024) learn a space whose distance counts steps, and AdaJEPA (Wang et al., 2026) adapts the world model inside MPC, but all of these change the representation or the predictor and so require a model that can be retrained. We instead propose to leave the world model frozen and learn a metric embedding of the frozen latent whose distance counts steps, updated from the planner's own episodes and blended into the planner's cost. Because it changes only the cost, this distance is complementary to hierarchical methods and can serve as the distance of their low-level planner.

## 3 Method: *TEMPO*

In this section we introduce *TEMPO*, a temporal-distance planning objective for frozen world models with one latent vector per state that plan by a latent distance to the goal, such as LeWM and PLDM. We first state the problem setting, namely the offline reward-free data, the latent world

model and the planner that every method in this paper shares. We then describe the learned temporal distance $F_\phi$ and its training objective, and finally how it enters the planner's cost.

### 3.1 PROBLEM SETTING

**Offline reward-free data.** We consider goal-conditioned control from pixels. The only training data is an offline set of trajectories

$$\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}, \qquad \tau = (o_{1:T},\, a_{1:T-1}), \tag{1}$$

where $o_t$ is an image and $a_t$ the action taken between $o_t$ and $o_{t+1}$, with no rewards, task labels or success signals, and the behaviour policies that produced the trajectories need not be optimal. A task is specified at test time by a goal image $o_G$.

**Latent world model.** The world models we study are joint-embedding predictive architectures (Figure 2), in which an encoder maps an observation to a latent, $z_t = E(o_t)$, and an action-conditioned predictor advances it, $\hat{z}_{t+1} = \mathrm{WM}(z_t, a_t)$. Both are trained on $\mathcal{D}$ to predict in embedding space rather than to reconstruct pixels, with the one-step latent prediction error $\|\mathrm{WM}(z_t, a_t) - E(o_{t+1})\|^2$ plus, when the encoder is trained, a regulariser that keeps the embedding from collapsing. LeWM and PLDM differ only in that regulariser, and both are single-latent world models that compress a frame into one 192-d latent vector, with actions in blocks of 5 environment steps. Neither has a decoder, a reward model or a policy, and both are frozen once trained.

**Planning by latent distance.** With $G = E(o_G)$, model-predictive control rolls the predictor forward under a candidate action sequence $a = a_{t:t+H-1}$, $\hat{z}_{t+k} = \mathrm{WM}(\hat{z}_{t+k-1}, a_{t+k-1})$ with $\hat{z}_t = z_t$, and scores the imagined trajectory by a terminal, a running and an action term,

$$J(a) = w_T\, \|\hat{z}_{t+H} - G\|^2 + w_r \sum_{k=1}^{H-1} \|\hat{z}_{t+k} - G\|^2 + w_a \sum_{k} \|a_{t+k}\|^2, \qquad a^\star = \arg\min_a J(a). \tag{2}$$

The published planners are the terminal-only instance, $w_T = 1$, $w_r = w_a = 0$,

$$a^\star = \arg\min_a \|\hat{z}_{t+H}(z_t, a) - G\|^2, \tag{3}$$

solved by the cross-entropy method (300 sequences, 30 iterations, top 30, and $H = 25$ steps as 5 blocks of 5) and executed in full before the state is re-encoded and Eq. (3) re-solved. LeWM and PLDM score the learned latent alone. This is the flat planner of every table, and the object *TEMPO* changes.

### 3.2 LEARNING THE TEMPORAL DISTANCE

$F_\phi$ is a two-layer MLP (hidden 128, ReLU) from the 192-d frozen latent $z$ to a 64-d embedding, trained so that distance in the embedding counts environment steps. On pairs of frames of the same episode $(z_t, z_{t+k})$,

$$\begin{aligned} \mathcal{L}_F = \text{smooth-}L_1\Big(\|F_\phi(z_t) - F_\phi(z_{t+k})\|,\ \tfrac{k}{\kappa}\Big)_{k \le 75} \\ + \tfrac{1}{2}\ \max\big(0,\ 3 - \|F_\phi(z_t) - F_\phi(z_{t+d})\|\big)_{d>75}, \qquad \kappa = 25, \end{aligned} \tag{4}$$

so that a 25-step displacement has unit length and states more than three plans apart are pushed at least three units apart, so the metric is calibrated up to three plans and only orders states beyond that. The loss uses only the time steps of the recorded trajectories, with no reward, environment state, action or success label. $F_\phi$ is first trained offline on the expert windows (Adam $10^{-3}$, batch 256, gradient clipping 10, 3,000 steps on 20,000 windows) and then keeps learning online. For eight rounds of 50 training windows the planner collects its own episodes in 50 parallel environments and $F_\phi$ takes one gradient step on every stored same-episode pair after each decision batch. The expert windows are the ones the world model is trained on and the online episodes are the planner's own, so *TEMPO* needs no data or labels beyond what the world model already uses.

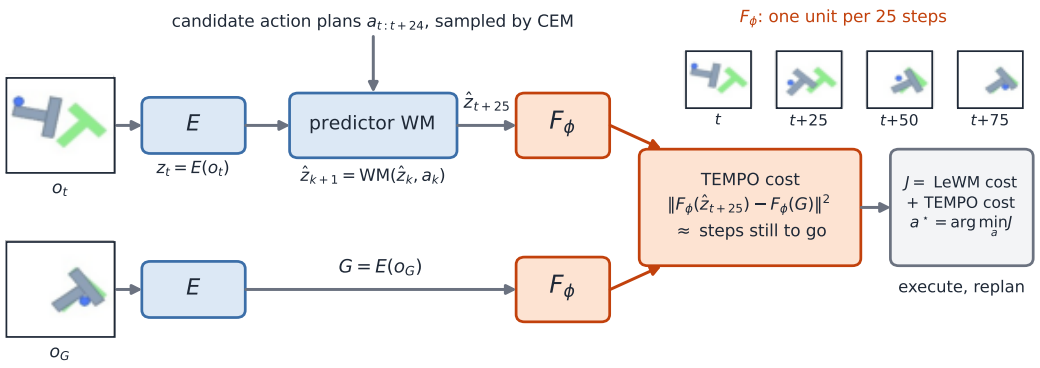


Figure 2: ***TEMPO* in the planner.** LeWM's frozen encoder $E$ and predictor WM (blue) are unchanged, and nothing is decoded. The planner rolls WM out under each CEM candidate plan and executes the one that minimises $J$. *TEMPO* adds the map $F_\phi$ (orange) and scores each candidate plan by $\|F_\phi(\hat{z}_{t+25}) - F_\phi(G)\|^2$, the steps its end state still is from the goal, added to LeWM's cost. Top right: $F_\phi$ places the frames of one episode one unit apart per 25 steps.

### 3.3 Planning with *TEMPO*

The planner of Eq. (3) keeps its world model, CEM, horizon and cadence and minimises

$$J(a) = (1-w)\,\|\hat{z}_{t+H}(z_t, a) - G\|^2 \;+\; w\,\bar{d}_H^{\,2}\,\|F_\phi(\hat{z}_{t+H}(z_t, a)) - F_\phi(G)\|^2, \qquad w = 0.5, \quad (5)$$

where $\bar{d}_H$ is the median 25-step latent displacement of the training data, so that the two terms are in latent units and the blend weights them equally. The raw latent keeps the fine position that the temporal coordinate does not resolve, and the temporal term orders plans by how far their end state still is from the goal in time. We use this blend rather than $F_\phi$ alone in every experiment because it is better on average for both LeWM and PLDM at every goal distance on the five standard environments (Appendix Table 5). Every $H = 25$ environment steps, with $z_t = E(o_t)$ and $G = E(o_G)$ from the frozen encoder,

$$\hat{z}_{t+k} = \mathrm{WM}(\hat{z}_{t+k-1}, a_{t+k-1}), \qquad \hat{z}_t = z_t, \qquad k = 1, \ldots, H, \quad (6)$$

$$a^\star_{t:t+H} = \arg\min_a J(a_{t:t+H}) \quad \text{by CEM with the cost of Eq. (5); execute all } H \text{ actions and re-encode,} \quad (7)$$

$$\phi \leftarrow \phi - \eta\,\nabla_\phi \mathcal{L}_F \quad \text{(online rounds: one step per decision batch on the stored pairs).} \quad (8)$$

The flat planner is Eqs. (6) and (7) with $w = 0$, and *TEMPO* adds the second term of Eq. (5) and the update of Eq. (8). $F_\phi$ is evaluated in batch on the CEM's own predicted end states, one small MLP forward pass per candidate against the $H$ predictor steps that produced them, so it adds well under $1\%$ of the compute of a plan (Appendix A). Control is closed-loop at the plan level, as for every method. Evaluation freezes everything, with 5 fixed groups of 50 windows per world-model seed and three seeds, with defaults in Appendix A. Why a distance that counts steps orders candidate plans where raw latent similarity does not is explained in Appendix I, which also shows that $F_\phi$ is goal-agnostic, generalises to held-out episodes and does not collapse to the frame index.

## 4 Experiments

Our experiments aim to answer the following questions: 1) Does *TEMPO* improve a frozen flat planner on goals several plans away? 2) Does it improve both frozen world models, LeWM and PLDM (Section 4.3)? 3) Does it transfer, without retraining, to environment configurations unseen in training (Section 4.4)? 4) Which part of *TEMPO*'s cost drives the gain, and why does the latent-distance cost fail far from the goal (Section 4.5)?

### 4.1 Environments and tasks

We use eleven environments spanning navigation, reaching and manipulation in two and three dimensions (Maes et al., 2026b; Zhou et al., 2024; Park et al., 2025; Fu et al., 2020), all run on stable-worldmodel (Maes et al., 2026a). They range from 2D point navigation (TwoRoom, Wall, WallRandom, PointMaze medium and large) through two-joint arm reaching (Reacher) and contact-rich tabletop pushing (Push-T, PushObj) to 3D manipulation with a robot arm (Cube, Scene) and a legged robot (Ant-U-Maze). Every trial starts from a state of a recorded expert episode, and the agent must reach a goal given only as an image, the expert's frame 25, 50 or 75 steps later, one to three plans away. The agent sees nothing but $224 \times 224$ RGB frames, and success follows each environment's own criterion. Details on the environments and datasets can be found in Appendix C.

### 4.2 Baselines and evaluation protocol

**Baselines.** We evaluate *TEMPO* on two JEPA world-model planners. Both learn from the offline trajectories alone, with no rewards or task labels, and plan as published through the same CEM on the same windows, and *TEMPO* is then added to each one's cost with everything else unchanged, so that each pair isolates the effect of the cost.

a) **LeWM** (Maes et al., 2026b): a recent JEPA world model that trains a ViT encoder and a latent predictor end to end from pixels with a prediction loss and a single anti-collapse regulariser (SIGReg), and plans by minimising the latent distance to the encoded goal. The frozen LeWM model is the instance on which every claim of this paper is made.

b) **PLDM** (Sobal et al., 2026): a world model that learns its encoder and predictor jointly with a VICReg-style multi-term objective (prediction, variance, covariance and temporal smoothness terms) and plans with the same latent-distance cost. We implement it with PLDM's objective on LeWM's ViT encoder and a 192-d latent, as in LeWM's baselines.

LeWM and PLDM are the current end-to-end JEPA latent-dynamics models learned from pixels, each with one compact latent per frame and 18M parameters, planning by the latent distance *TEMPO* corrects. TwoRoom, Reacher, Push-T and Cube are LeWM's benchmark tasks with its released datasets, and one plan from the goal is LeWM's own protocol (goal 25 steps ahead, budget 50), and on these tasks LeWM is the primary comparator. We do not include offline goal-conditioned reinforcement-learning and imitation methods such as GCBC, IQL and IVL (Park et al., 2023; 2025), which learn a policy or value function from the data, whereas *TEMPO* is not a reinforcement-learning method.

**Training.** Every world model is trained once on the released offline expert set for its environment, 1,789 to 20,000 reward-free trajectories (Table 4), with the 10-epoch recipe of Maes et al. (2026b), three seeds per environment, and is then frozen, with hyperparameters fixed across environments (Appendices A and D). $F_\phi$ learns in two stages, first offline, on windows cut from the same expert trajectories, and then online, from the episodes the planner itself generates during eight online rounds (Section 3.2). Its architecture, loss and schedule are identical everywhere.

**Planning.** Every planner optimises a plan of $H = 5$ action blocks (25 environment steps) with the cross-entropy method (CEM) (Maes et al., 2026b; Zhou et al., 2024), with LeWM's settings (300 candidates, 30 iterations and 30 elites, Appendix B). The whole plan is executed, the state re-encoded and the plan re-solved until the episode ends. We measure goal distance in plans, so a goal $k$ plans away is the frame $25k$ environment steps later in the same recorded trajectory, so beyond one plan no single plan reaches it and the planner must chain at least $k$ of them, and the planner is the same at every goal distance, only the goal changes. At that range the latent distance acts like a straight-line cost that can stall against obstacles, whereas $F_\phi$, learned from trajectories, counts steps along feasible paths, so each short plan makes progress along the true route. The only difference between a planner and its *TEMPO* variant is the cost the CEM minimises, Eq. (3) for the planner and Eq. (5) for *TEMPO*.

**Evaluation.** An evaluation window is a slice of a recorded expert episode, in which the environment is reset to frame $s$ and the planner is shown the expert's frame at $s +$ offset as the goal. Offset 25 is the published setting, one plan away, while offsets 50 and 75 ask for states two and three plans

Table 1: **Success rates (%) on five pixel-based goal-conditioned control tasks, one to three plans from the goal.** Bold marks the better of each pair. $\Delta$ is the gain of *TEMPO* over the same planner, averaged over environments. Every world model plans from pixels alone.

| Planner | TwoRoom | Reacher | Push-T | Cube | Scene | Avg. | $\Delta$ |
|---|---|---|---|---|---|---|---|
| **LeWM** | | | | | | | |
| one plan | 87.7±0.8 | 94.7±0.7 | 93.6±1.0 | 74.8±2.4 | 69.6±1.1 | 84.1 | – |
| **+ TEMPO (ours)** | **97.2**±0.6 | **95.5**±0.7 | **95.1**±0.6 | **77.1**±1.7 | **71.1**±1.1 | **87.2** | +3.1 |
| two plans | 58.5±1.7 | 95.7±0.6 | 49.7±1.3 | 57.9±2.4 | 56.4±1.2 | 63.7 | – |
| **+ TEMPO (ours)** | **99.2**±0.3 | **96.8**±0.5 | **55.7**±1.2 | **58.7**±2.0 | **57.9**±1.3 | **73.7** | +10.0 |
| three plans | 36.3±2.3 | 89.7±0.4 | 22.0±0.6 | 68.7±1.6 | 48.7±1.3 | 53.1 | – |
| **+ TEMPO (ours)** | **99.2**±0.3 | **98.4**±0.5 | **28.1**±2.0 | **70.4**±1.2 | **50.5**±1.3 | **69.3** | +16.2 |
| **PLDM** | | | | | | | |
| one plan | 96.8±0.4 | 93.6±1.0 | 76.1±1.8 | 60.3±2.3 | 56.1±1.8 | 76.6 | – |
| **+ TEMPO (ours)** | **98.9**±0.3 | **94.5**±1.0 | **90.1**±1.2 | **68.7**±1.4 | **58.4**±1.3 | **82.1** | +5.5 |
| two plans | 82.0±2.0 | 93.6±0.6 | 31.9±1.5 | 48.5±1.7 | 43.6±0.9 | 59.9 | – |
| **+ TEMPO (ours)** | **97.2**±0.5 | **98.0**±0.4 | **39.5**±1.8 | **48.7**±1.8 | **45.9**±1.2 | **65.8** | +5.9 |
| three plans | 70.5±1.4 | 88.0±0.5 | 11.6±1.3 | 65.6±1.7 | 38.7±1.7 | 54.9 | – |
| **+ TEMPO (ours)** | **95.7**±1.1 | **97.7**±0.5 | **17.1**±1.7 | **66.7**±2.0 | **41.1**±1.5 | **63.7** | +8.8 |

away, beyond anything the predictor is rolled out to. Success is mean $\pm$ std error over 15 evaluations (5 groups $\times$ 3 seeds) of 250 identical windows per seed. In tests outside the training distribution, $F_\phi$ is trained in the training environment and kept fixed during evaluation, so any gain under the shift comes from the temporal distance.

### 4.3 TEMPO: LEARNED TEMPORAL DISTANCE FOR PLANNING WITH FROZEN WORLD MODELS

**The flat planners collapse with distance.** Tables 1 and 2 report planning success for every planner one, two and three plans from the goal, on identical windows, where each "+ *TEMPO*" row keeps the planner above it and changes only the cost it minimises. Both flat planners succeed one plan from the goal but degrade as it moves away, most sharply on TwoRoom and Push-T. The collapse is therefore a property of the objective these planners share, a latent distance to a far goal, rather than of any one world model.

***TEMPO* closes the gap.** *TEMPO* consistently improves both planners at every horizon, and on the standard tasks its gain is largest three plans from the goal, where the flat planners collapse ($\Delta$ column of Table 1). Remarkably, three plans from the goal it takes LeWM on TwoRoom and Reacher to near-perfect success, and on the more complex tasks of Table 2 it more than doubles PointMaze medium one plan out. On the remaining tasks, *TEMPO* remains competitive. Additionally, the same pattern holds for PLDM, and LeWM with *TEMPO* consistently outperforms the published PLDM planner on every core environment at every goal distance. These gains come at negligible cost, since both variants roll the same candidates through the predictor, and *TEMPO* adds one small MLP evaluation per candidate, at most $0.3\%$ of a plan's arithmetic (Appendix B).

**Cross-room TwoRoom.** On PLDM's own Two-Rooms evaluation (Sobal et al., 2026), the goal always lies in the other room, so every trial must find the door. Each group uses a fixed set of 50 start and goal pairs in different rooms, and the one-, two- and three-plan rows use a fixed budget of 50, 100 and 150 steps. Using the same world models and the same $F_\phi$ without refitting, *TEMPO* roughly quadruples LeWM's success at every budget (Table 2).

### 4.4 TOWARDS GENERALISATION TO NOVEL ENVIRONMENT CONFIGURATIONS

Beyond new goals, we also evaluate *TEMPO* on environments that differ from training, using three families with test configurations unseen in training, namely WallRandom (Zhou et al., 2024), with unseen wall and door layouts (a layout shift) and a cross-wall test that asks the planner to cross an unseen wall, PushObj (Zhou et al., 2024), with unseen block shapes (a shape shift), and TwoRoom with the door moved to a position never seen in training. The world model, the planner and $F_\phi$ see only the training configurations (Figure 3). Notably, *TEMPO* substantially improves the LeWM

Table 2: **Success rates (%) on more complex goal-conditioned control tasks, one to three plans from the goal.** Reporting, bold and Δ as in Table 1. †Wall episodes end at 50 steps, so its two-plan cell uses goal 40 and it has no three-plan cell.

| Planner | Wall† | PointMaze M | PointMaze L | Ant-U | Avg. | Δ | |
|---|---|---|---|---|---|---|---|
| **LeWM** | | | | | | | |
| one plan | 67.6±1.4 | 12.0±1.0 | 32.1±1.4 | 28.0±2.0 | 19.7±1.0 | 31.9 | – |
| **+ TEMPO (ours)** | **75.7**±2.2 | **46.5**±1.2 | **75.7**±1.7 | **54.8**±1.6 | **21.9**±1.1 | **54.9** | +23.0 |
| two plans | 57.7±2.4 | 15.5±1.0 | 17.6±1.0 | 2.9±0.6 | 25.5±1.5 | 23.8 | – |
| **+ TEMPO (ours)** | **65.9**±2.2 | **69.6**±1.8 | **41.1**±2.0 | **21.3**±1.2 | **29.6**±1.0 | **45.5** | +21.7 |
| three plans | – | 17.1±1.1 | 10.8±0.8 | 0.3±0.2 | 28.1±1.6 | 14.1 | – |
| **+ TEMPO (ours)** | – | **74.5**±1.7 | **31.7**±1.0 | **6.5**±1.0 | **34.0**±2.1 | **36.7** | +22.6 |
| **PLDM** | | | | | | | |
| one plan | 71.2±1.9 | 23.9±1.5 | 22.8±2.1 | 32.3±1.5 | 17.1±1.5 | 33.5 | – |
| **+ TEMPO (ours)** | **82.3**±1.7 | **42.5**±1.5 | **43.1**±1.8 | **48.5**±2.9 | **18.5**±1.7 | **47.0** | +13.5 |
| two plans | 63.5±2.4 | 35.2±1.7 | 16.8±1.0 | 6.5±0.7 | 24.3±1.1 | 29.3 | – |
| **+ TEMPO (ours)** | **72.5**±1.4 | **62.0**±1.6 | **26.9**±2.1 | **15.1**±1.8 | **29.3**±1.7 | **41.2** | +11.9 |
| three plans | – | 39.1±1.8 | 8.7±0.8 | 3.1±0.8 | 26.8±1.5 | 19.4 | – |
| **+ TEMPO (ours)** | – | **65.1**±1.7 | **16.7**±0.8 | **7.1**±1.0 | **29.3**±1.7 | **29.5** | +10.1 |

Table 3: **Generalisation to unseen configurations: a layout shift (WallRandom), a shape shift (PushObj) and a moved door (TwoRoom).** World models and TEMPO are trained on the training layouts, block shapes or door position; in-distribution results are in Tables 1 and 2. Cross-wall: the test of Zhou et al. (2024), with start and goal on opposite sides of an unseen wall. $^{\ell}$: LeWM trained with single-layout batches. Bold marks the better of each pair.

| | WallRandom | | PushObj | | TwoRoom | |
|---|---|---|---|---|---|---|
| Planner | unseen 1 plan | unseen cross-wall | unseen 1 plan | unseen 3 plans | door moved 1 plan | door moved 3 plans |
| LeWM$^{\ell}$ planner | 67.5±1.7 | 15.1±1.9 | 53.6±2.2 | 18.8±1.4 | 65.3±1.9 | 9.9±1.0 |
| **+ TEMPO (ours)** | **76.9**±1.4 | **22.9**±2.5 | **54.8**±2.0 | **20.5**±1.4 | **74.5**±1.6 | **30.1**±0.8 |
| PLDM planner | 75.6±2.0 | 6.0±1.0 | 44.1±1.6 | 11.1±0.8 | 73.9±1.7 | 18.9±1.0 |
| **+ TEMPO (ours)** | **77.9**±2.0 | **6.5**±1.1 | **48.8**±1.7 | **15.3**±1.0 | **76.3**±1.5 | **30.4**±1.0 |

planner on unseen wall layouts and on the cross-wall test. It also improves PLDM on PushObj's unseen shapes and remains competitive for LeWM, and on the moved door it substantially improves both planners three plans out, where every planner's success is lowest (Table 3). Beyond configurations, *TEMPO* also keeps its advantage under test-time visual and dynamics shifts, improving the planner under blur and colour jitter on TwoRoom and Push-T and with a larger agent or a heavier arm, and remaining competitive under the remaining corruptions and camera changes (Appendix G, Tables 6, 7 and 8). We further extend the standard and more complex tasks to a mobile manipulator in RoboCasa kitchens (Nasiriany et al., 2024), covering the core skills of pick-and-place and navigation (Appendix H).

### 4.5 Ablations

Appendix Table 5 separates the two terms of the cost. $F_\phi$ alone remains competitive with the blend on TwoRoom and Reacher but falls behind on Push-T for both planners and on Cube and Scene for LeWM, because the latent term carries the fine positioning that manipulation needs and $F_\phi$ the ordering of far states, so combining the two terms works best. Finally, Appendix J shows why the flat planner fails at range. Three plans out, the latent distance ranks where a plan ends but not whether it succeeds, while *TEMPO*'s cost ranks the same plans closer to their eventual success (Tables 12 and 13).

## 5 Conclusion

We introduce *TEMPO*, a simple yet effective temporal-distance planning objective that corrects the planning cost of frozen world models with one latent vector per state without retraining them, task-

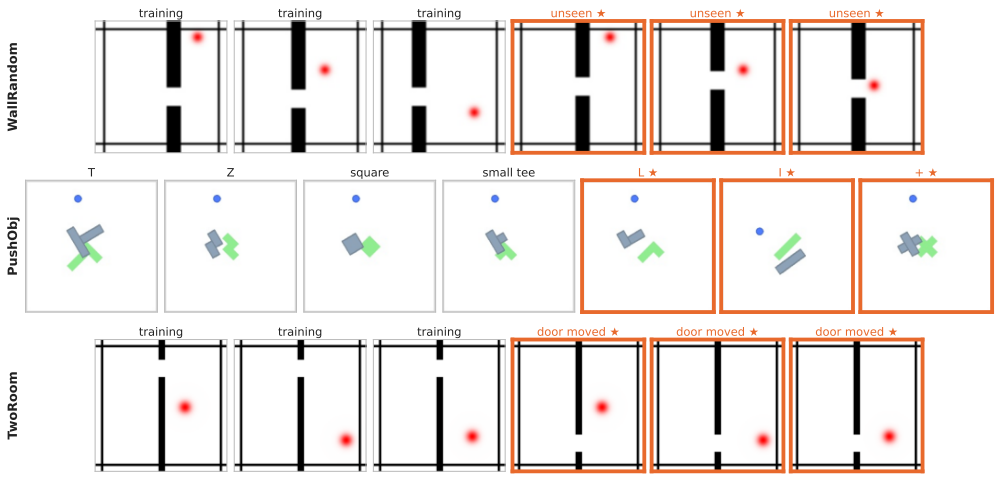


Figure 3: **Training and test setups for the three generalisation families of Table 3.** Unseen configurations are outlined in orange ($\star$). WallRandom holds out wall and door positions outside every training range, PushObj holds out the L, I and + shapes, and TwoRoom moves the door to the opposite end of the wall.

specific tuning or reward. A distance that counts environment steps, learned from the time steps of recorded trajectories and blended with the latent distance, turns the long-horizon collapse of flat planners into near-complete success, taking LeWM from 36 to 99% on TwoRoom and from 90 to 98% on Reacher three plans from the goal and from 17 to 75% on the harder cross-room task. The same cost improves both the LeWM and the PLDM planner at every goal distance, including one plan away, for at most 0.3% of a plan's arithmetic, and generalises without retraining to layout shifts (unseen wall layouts and door positions), shape shifts (unseen block shapes), visual shifts (blur and colour jitter) and dynamics shifts (a larger agent or a heavier arm).

**Future work.** Like the world models it builds on, *TEMPO* benefits from a diverse offline dataset and follows the general trade-off that broader coverage improves generalisation. It adds no requirement of its own, as $F_\phi$ is trained on the world model's own training set, needs no reward signals or labels beyond the time steps of its trajectories, and works on random-walk data (Wall, WallRandom) as well as on demonstrations. Because it keeps learning from the planner's own episodes, continuing to update it with new experience could enable progressive improvement without extensive interaction up front. Already, an $F_\phi$ trained only on training configurations substantially improves the planner on unseen wall layouts and remains competitive on unseen block shapes. Another direction is to extend *TEMPO* to world models that plan over the full spatial grid of patch tokens (Zhou et al., 2024). Like the world models it corrects, *TEMPO* is trained per environment with one shared recipe, and learning a single temporal distance across environments on top of a multi-task world model is a further extension. Finally, all our experiments are in simulation, and since *TEMPO* leaves the world model untouched and learns only from the time steps of recorded trajectories, validating it on a real robot is a natural next step.

## REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

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

Jiaxin Bai and Jiaxuan Xiong. Temporal-distance jepa: Plan-aware representation learning for latent world model predictive control. *arXiv preprint arXiv:2607.25337*, 2026.

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

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

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

Letian Cheng, Qi Zhang, and Yisen Wang. Sage: Subgoal-conditioned action generation for latent world model planning. *arXiv preprint arXiv:2607.17973*, 2026.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

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 control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: Rc-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, et al. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a.

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

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *Advances in neural information processing systems*, 32, 2019.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *Advances in Neural Information Processing Systems*, 38:43905–43941, 2026.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

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

# A METHOD DETAILS

**Learned temporal distance $F_\phi$.** Two-layer MLP on the 192-d state latent (hidden 128, ReLU) to 64-d; loss of Eq. (4) on same-episode pairs with $k \sim U[1, 75]$ ($\kappa = 25$) and a hinge of margin 3 on pairs more than 75 steps apart; Adam $10^{-3}$, batch 256, gradient clipping 10; 3,000 offline steps on the 20,000 expert windows, then one step per trainer update in every online round. Cost of Eq. (5): $\bar{d}_{25}$ is computed once per world model as the median norm of its 25-step latent displacement over

its own encoded training data, so no scale is set by hand. Because $F_\phi$ keeps learning online while the planner uses it, the embedding drifts between rounds; the planner re-solves from the current $F_\phi$ at every decision, so the drift is absorbed.

**World model training.** Every world model follows the 10-epoch protocol of Maes et al. (2026b) on stable-worldmodel (Maes et al., 2026a). A frame-skip of 5 groups the actions between two stored frames into one action block. Training uses sub-trajectories of 4 frames, i.e. 3 context frames and 1 prediction, with 4 action blocks of 5 actions, and the planner's horizon of 5 blocks is 25 environment steps. Frames are $224\times224$. LeWM's encoder is a HuggingFace Vision Transformer Tiny with patch size 14, trained from scratch (no pretrained weights), emitting a 192-d class token; its predictor is a causal transformer over the frame history (depth 6, 16 heads, MLP width 2048, dropout 0.1) that consumes the 192-d latents and the action blocks and emits the next latent. Batch size 128, AdamW at $5 \times 10^{-5}$, three seeds (42, 43, 44) per environment, frozen after training. The Wall environment itself is the released planning environment of Zhou et al. (2024), in which one unit of action moves the dot by two arena units, whereas the released Wall dataset was generated at one unit per unit of action; every Wall evaluation here, as in the original, therefore executes the planner's actions at twice the displacement the world model was trained on. All arms share the environment, so the pairs in Table 2 are unaffected, but the absolute Wall numbers should be read with this in mind. PLDM trains its own encoder and predictor. Neither LeWM nor PLDM receives any input beyond pixels and actions.

**Planner.** stable-worldmodel CEM over LeWM's predictor: 300 samples, 30 iterations, top 30, horizon 5 action blocks of 5 environment steps (actions StandardScaler-normalised in the planner's block layout), 3 context frames, warm start from the previous plan, cost of Eq. (5) on the last predicted latent (ours) or the latent distance alone (LeWM). Two settings differ from Maes et al. (2026b), identically for every arm: the CEM runs 30 iterations in every environment (the published setup uses 10 outside Push-T), and the predictor sees 3 context frames everywhere (1 on TwoRoom in the published setup). The flat planner still reproduces the published LeWM numbers on LeWM's own protocol (Appendix F).

**Data and schedule.** Every dataset frame is encoded once. Offline: 20,000 expert windows $[t_0, t_0+$ offset$]$ with $G = z_{t_0+\text{offset}}$, 3,000 gradient steps (Adam $10^{-3}$, batch 256). Online: 8 rounds of 50 training windows in 50 parallel environments; after each decision batch $F_\phi$ takes 5 gradient steps on the stored same-episode pairs of every episode in the buffer (expert and the planner's own), 50 at each episode end. *Collection policy.* The online rounds of the reported cells are collected by an exploratory earlier planner variant that aims the same CEM at intermediate latents sampled around the current state; collecting them with the planner of Eq. (5) itself gives the same success at goal 75 on TwoRoom, Push-T, Reacher, Cube and Scene, the largest difference over the five being 3.2 points. Evaluation: $F_\phi$ frozen, 5 fixed groups of 50 windows per world-model seed, three seeds, planner seed fixed, budget $2\times$ offset, success = the environment's criterion within budget. The LeWM and ours arms of one cell are evaluated in one job on identical windows.

## B PLANNING SOLVER

**Planning solver.** For planning we use the cross-entropy method (CEM) (Rubinstein & Kroese, 2004), the solver that performed best in prior work on these world models (Maes et al., 2026b; Zhou et al., 2024). At each planning step CEM samples 300 candidate action sequences and optimises them for 30 iterations in every environment; at each iteration the top 30 sequences are retained to update the sampling distribution, and the initial sampling variance is set to 1 in the normalised action space. The planning horizon is 5 steps, which corresponds to 25 environment timesteps due to a frame skip of 5. We employ a receding-horizon model predictive control scheme with a horizon of 5, meaning that the entire optimised action sequence is executed before replanning, and the next solve is warm-started from the previous mean. This configuration follows the LeWM setup (Maes et al., 2026b), which uses 30 iterations on Push-T and 10 elsewhere; we use 30 everywhere. The only difference between the arms of a pair is the cost $J$ in line 6 of Algorithm 1: the squared latent distance to the encoded goal for the LeWM and PLDM planners, and Eq. (5) for *TEMPO*. LeWM and PLDM solve 50 environments per batch.

**Algorithm 1** Cross-entropy method over action sequences (one planning step)

**Require:** frozen world model WM, current latent $z_t$, goal $G$, horizon $H$, samples $N$, elites $K$, iterations $T$, cost $J$

initialise $\mu_0$ (previous plan, or 0), $\sigma_0 = 1$
**for** $\tau = 1$ to $T$ **do**
  sample $N$ action sequences $a^{(i)}_{1:H} \sim \mathcal{N}(\mu_{\tau-1}, \sigma^2_{\tau-1})$
  **for** $i = 1$ to $N$ **do**
    roll out $a^{(i)}_{1:H}$ through WM from $z_t$ to obtain $\hat{z}^{(i)}_{t+1:t+H}$; $\quad J^{(i)} \leftarrow J(\hat{z}^{(i)}_{t+1:t+H}, G)$
  **end for**
  $\mathcal{E} \leftarrow$ the $K$ sequences with the lowest $J^{(i)}$
  $\mu_\tau \leftarrow \frac{1}{K} \sum_{i \in \mathcal{E}} a^{(i)}_{1:H}, \quad \sigma^2_\tau \leftarrow \mathrm{Var}_{i \in \mathcal{E}}\, a^{(i)}_{1:H}$
**end for**
**return** $\mu_T$, executed for $H$ blocks before the next planning step

**Compute.** Both arms roll the same $300 \times 30$ candidates through the predictor ($2.25 \times 10^5$ forwards per plan); *TEMPO* adds one two-layer MLP evaluation per candidate, $0.03$ to $0.3\%$ of the plan's arithmetic, and no second model or search.

**Hyperparameters.** These settings are constant across environments and planners: every planner (LeWM and PLDM, each with and without *TEMPO*) uses 300 CEM samples, 30 iterations, the top 30 as elites and a 25-step plan in every environment, and *TEMPO* blends its temporal distance with weight $w = 0.5$ everywhere. Goals lie 25, 50 and 75 steps ahead (one to three plans); Wall, whose episodes end at 50 steps, uses 25 and 40. Each cell is evaluated on five fixed groups of 50 windows per world-model seed, three seeds.

## C Environments and datasets

Figure 4 shows one training frame of each environment. Frames are $224 \times 224$ RGB, actions are grouped into blocks of 5 by a frame-skip of 5, and every world model is trained for 10 epochs on the sets below (Appendix A); Table 4 gives their sizes. Success is each environment's own criterion.

Table 4: **Training sets.** Episodes the world models were trained on, mean episode length in environment steps, and action dimension. Where a 90:10 episode split defines the protocol (Section 4.1) the count is the 90% training split.

| Environment | Episodes (trained on) | Mean length | Action dim. |
|---|---|---|---|
| TwoRoom | 10,000 | 92 | 2 |
| Push-T | 18,685 | 125 | 2 |
| Reacher | 10,000 | 200 | 2 |
| Cube | 10,000 | 200 | 5 |
| Scene | 10,000 | 200 | 5 |
| PushObj | 20,000 | 101 | 2 |
| Wall | 1,920 | 50 | 2 |
| PointMaze medium | 2,700 | 200 | 2 |
| PointMaze large | 2,700 | 200 | 2 |
| Ant-U-Maze | 6,467 | 100 | 8 |
| RoboCasa Pick-and-Place | 1,935 | 300 | 6 |
| RoboCasa Kitchen-Nav | 1,789 | 300 | 3 |

(a) **TwoRoom** is the continuous 2D navigation task of Sobal et al. (2026): two rooms separated by a wall with a single door. The agent, a coloured dot, must travel from a random start in one room to a randomly sampled goal in the other, which requires passing through the door. We use the 10,000 episodes of the PLDM/LeWM release, average length 92 steps, generated by a noisy heuristic that drives the agent to the door and then to the goal. Each world model is trained on this set for 10 epochs.

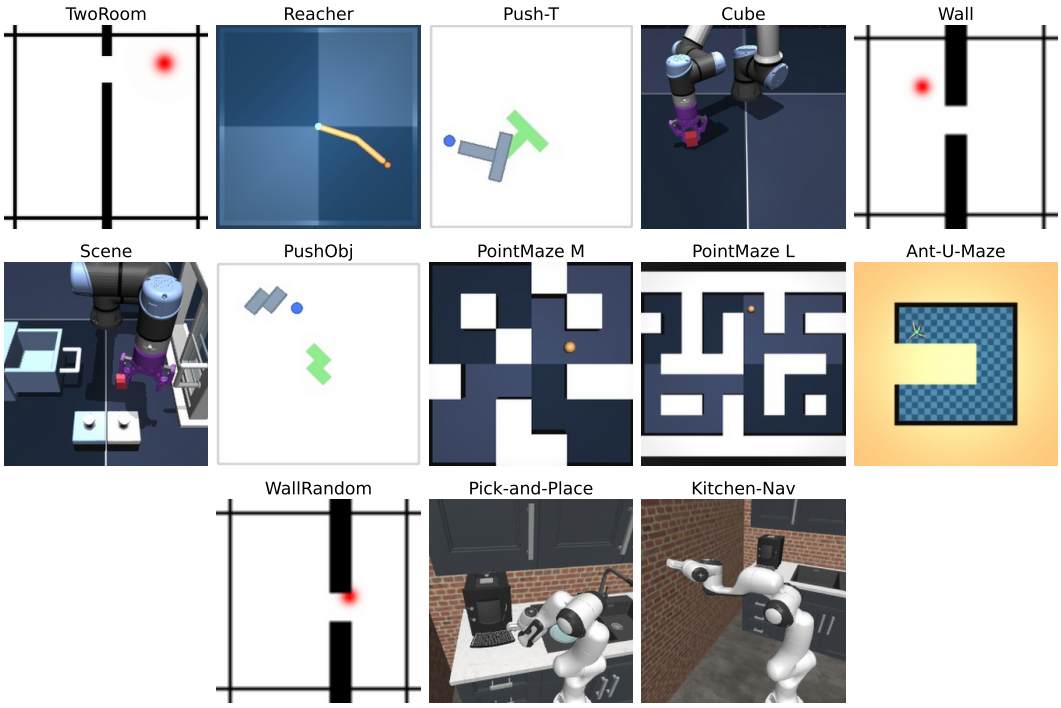


Figure 4: **The eleven environments and two RoboCasa kitchen tasks**, left to right, top to bottom: TwoRoom, Reacher, Push-T, Cube, Wall, Scene, PushObj, PointMaze medium and large, Ant-U-Maze, WallRandom, and the RoboCasa kitchen tasks Pick-and-Place and Kitchen-Nav. Each panel is one $224 \times 224$ training frame, the only observation the world models and the planner receive.

(b) **Push-T** is the continuous 2D manipulation task in which an agent (a blue dot) must push a T-shaped block to a target pose through pushing contacts only. We follow the setup and expert dataset of Zhou et al. (2024) through the platform's training split, 18,685 episodes of average length 125 steps, and, as LeWM does, train each world model for 10 epochs.

(c) **Reacher** is the two-joint arm of the DeepMind Control Suite (Tassa et al., 2018). Following Zhou et al. (2024), success requires the arm to match the joint configuration that reaches the target. The platform's 10,000 episodes of 200 steps are collected with a Soft Actor-Critic policy; each world model is trained on them for 10 epochs.

(d) **Cube** and **Scene** are the OGBench single-object manipulation tasks (Park et al., 2025): a robot arm must move a cube to a target pose (Cube), or operate a drawer, a window, two buttons and a cube into a target configuration (Scene). For each we collect 10,000 episodes of 200 steps with the benchmark's scripted expert and train each world model for 10 epochs.

(e) **PushObj** is Push-T with the block shape varied, from the generalisation suite of Zhou et al. (2024). We collect 20,000 episodes of 100 steps over the four training shapes (T, o, square, small tee) with the platform's Push-T collection policy, and hold out the original test shapes for Table 3; success requires both the agent and the block at their targets.

(f) **Wall** is the point agent of Zhou et al. (2024) that must cross a wall through its door. We use their 1,920 random-walk trajectories of 50 steps; because episodes end at 50 steps its horizons are 25 and 40 steps rather than 25, 50 and 75. **WallRandom** (Table 3) is the same task with the wall and door positions randomised, 10,240 random-walk trajectories of 50 steps for training and held-out layouts for evaluation, as in the original.

(g) **PointMaze medium** and **large** are the OGBench mazes with the benchmark's expert trajectories, 3,000 episodes of 200 steps each, split 90:10 by episode; the world models train on the 2,700 training episodes for 10 epochs.

(h) **Ant-U-Maze** is D4RL's antmaze-umaze-diverse (Fu et al., 2020), an 8-d quadruped in a U-shaped maze. The recorded action sequences are re-simulated in our simulator in 100-step chunks so that every stored state is reproducible at evaluation (Appendix C), giving 6,467 training and 698 held-out chunk-episodes of 100 steps.

(i) **RoboCasa Pick-and-Place** and **Kitchen-Nav** are two tasks in a RoboCasa kitchen (Nasiriany et al., 2024) with a mobile manipulator: moving an object between the counter and the sink, and driving the base to a target pose. Pick-and-Place succeeds when the object lies within $0.05$ m of its goal position; Kitchen-Nav when the base is within $0.20$ m of the goal (the task's own threshold) with a heading cosine of at least $0.95$ (the task uses $0.98$). Both are collected by our scripted experts. The Pick-and-Place expert repeatedly picks the object and places it, alternating between the sink and the counter, driving the mobile base into reach before each approach and carry; it retries a failed grasp up to three times and then falls back to random arm motion for the rest of the episode. The Kitchen-Nav expert is a waypoint follower over a floor grid rasterised from the fixtures' footprints. The scene of each episode is stored so that evaluation rebuilds it exactly. Pick-and-Place has 1,935 episodes of 300 steps (2,000 collected, 65 dropped for render faults) and Kitchen-Nav 1,988 episodes of 300 steps split 90:10; each world model is trained for 10 epochs. Windows already solved at their start row are excluded for every arm before the seeded draw: on Pick-and-Place those whose object lies within $0.05$ m of its goal ($332{,}570$, $227{,}350$ and $159{,}342$ of $582{,}435$ rows at one, two and three plans), on Kitchen-Nav those whose base lies within $0.20$ m ($17{,}301$, $12{,}038$ and $8{,}360$ of $59{,}899$).

**Ant-U-Maze.** Ant-U is the Ant-U-Maze of Sobal et al. (2026) (D4RL antmaze-umaze-diverse), planned from pixels. D4RL's recorded states come from a different simulator, so we re-simulate its recorded actions in our MuJoCo environment with D4RL's ant model, in 100-step chunks anchored at the recorded states, dropping chunks that start with the ant on its back or stuck; the split is by source episode (6,467 training and 698 held-out chunks). An evaluation window needs the goal at least 1 m away, the ant upright at the start and the goal, and a goal the recorded actions reach. Success is the maze goal within $0.5$ m, as in Sobal et al. (2026), whose Ant-U model is state-based, so this column is an internal comparison, not a reproduction.

## D BASELINE: PLDM AS TRAINED HERE

The PLDM baseline is trained on the stable-worldmodel platform (Maes et al., 2026a) with the configurations released there, for the same 10 epochs, on the same recorded trajectories, frame-skip and $224 \times 224$ frames as the LeWM boards (Appendix A), one board per seed (42, 43, 44). We did not tune it per environment; every coefficient below is the platform's default, and the same values were used on every environment. Its planner is the published one: sampling-based MPC over the frozen model with the squared latent distance to the encoded goal as the cost.

**PLDM.** PLDM (Sobal et al., 2026) learns encoder and predictor end to end and prevents collapse with a VICReg-style regulariser (Bardes et al., 2021) extended along time. The objective is a weighted sum of a prediction term and six regularisers,

$$\mathcal{L}_{\text{PLDM}} = \mathcal{L}_{\text{pred}} + \alpha\,\mathcal{L}_{\text{var}} + \beta\,\mathcal{L}_{\text{cov}} + \gamma\,\mathcal{L}_{\text{time-sim}} + \zeta\,\mathcal{L}_{\text{time-var}} + \nu\,\mathcal{L}_{\text{time-cov}} + \mu\,\mathcal{L}_{\text{IDM}}, \tag{9}$$

with each term as defined by Sobal et al. (2026). The variance and covariance terms hold the batch of latents at each step spread out and decorrelated; their temporal counterparts do the same along each trajectory; the time-similarity term keeps consecutive latents close; the inverse-dynamics term asks the pair $(z_t, z_{t+1})$ to determine the action. Maes et al. (2026b) report that the original coefficients were tuned per environment and dataset and re-tuned them by a grid search over 256 configurations on Push-T. What we did instead: we kept the platform's released coefficients for every environment, with no search: $\alpha = 18$, $\beta = 12$, $\gamma = 0.2$, $\zeta = 0.7$ and $\nu = \mu = 0$ (the temporal covariance and inverse-dynamics terms are present in the code but switched off), a projector loss of weight $10^{-3}$, and no SIGReg term. The encoder is a ViT-Ti/14 trained from scratch with a 192-d latent through a two-layer projector (hidden width 2048, batch norm); the predictor is a transformer of depth 6, 16 heads, MLP width 2048, dropout 0.1 over a three-frame history predicting one frame; optimisation is AdamW at learning rate $5 \times 10^{-5}$ and weight decay $10^{-3}$, batch size 128, bf16, gradient clipping at 1.0, 10 epochs. Its planning cost is the published squared latent distance, run through the same

CEM as LeWM's (Appendix E). Because no coefficient was tuned on our environments, the PLDM rows of Table 1 are a conservative baseline: a per-environment search of the kind Maes et al. (2026b) ran could raise them, and it would raise the PLDM $+F_\phi$ row with them, since $F_\phi$ reads whatever latent the board provides.

## E THE METHOD ON OTHER WORLD MODELS: WHAT $F_\phi$ ASKS OF THEM

The method is the cost, not the planner it was first measured on: every "$+ F_\phi$" row keeps the planner above it and changes only what that planner minimises. What follows is therefore not the portability of one system but the same method applied to a second frozen world model. The learned temporal distance touches a world model through two calls and nothing else. *An encoder* that maps one frame to one latent vector $z$; for LeWM this is the 192-d projected class token of its ViT, with no proprioception. *A rollout* that takes the context latents and a candidate action sequence and returns the predicted latents along the plan; the CEM of Eq. (5) scores its last entry. $F_\phi$ is a two-layer MLP whose input width is the latent dimension, its loss (Eq. (4)) reads only the time steps of the recorded trajectories, and the blend of Eq. (5) needs the model's own distance as its first term. Any frozen latent world model that exposes those two calls, one latent vector per frame and a latent rollout, can carry it.

**PLDM.** PLDM's model exposes the same two calls with the same shapes: a ViT class token through a projector to a 192-d latent, and an autoregressive predictor over a three-frame history, trained on the platform with the same data and schedule as the LeWM boards. The two differ in the training objective, an inverse-dynamics term with a variance–covariance regulariser for PLDM against LeWM's SIGReg, not in the interface. Running Eq. (5) on a PLDM board is therefore the same harness with the checkpoint swapped; the flat planner on the same board, PLDM's published latent-distance cost run through the same CEM, is the matched control. The results on PLDM are the PLDM rows of Tables 1 and 2.

**Why the baseline rows use the published costs.** PLDM enters Table 1 as the planner its authors published, run through the same CEM on the same windows; that is what a baseline row should mean. The question this section answers is a different one, whether the temporal distance is a property of the cost rather than of LeWM's latent, and it needs the matched pairs above to be settled.

## F ABLATIONS AND PER-OFFSET RESULTS

**Reproducing the published LeWM planner.** Run on LeWM's own protocol, our flat planner matches the published numbers of Maes et al. (2026b) on TwoRoom (87.7 against 88.8), Push-T (93.6 against 90.4) and Cube (74.9 against 72.4), and exceeds them on Reacher (94.7 against 81.2). The Reacher gap is the CEM budget rather than the cost: we run 30 CEM iterations in every environment, where the published setup uses 10 outside Push-T.

Table 5 separates the two terms of the cost (Section 4.5). With the world model and planner of each row fixed, it compares the blend of Eq. (5) with $F_\phi$ alone on the five environments of Table 1, one to three plans from the goal, for LeWM and PLDM; the blend has the higher average in every block.

**Two plans away.** At offset 50 (Table 1, middle block) the order is the same as at three plans, with smaller gaps on the environments the flat planner still solves: TwoRoom from 58.5 to 99.2, Push-T from 49.7 to 55.7, Reacher from 95.7 to 96.8, Cube from 57.9 to 58.7 and Scene from 56.4 to 57.9. Against the published PLDM planner, *TEMPO* improves every environment at two plans.

## G GENERALISATION TO UNSEEN CONFIGURATIONS AND TEST-TIME SHIFTS

Every evaluation here changes the environment rather than the world model, which stays frozen: unseen layouts and block shapes (Table 3), the cross-room task, pixel corruptions and factor shifts of the scene and its dynamics.

Table 5: **Ablation of the planning cost on each frozen world model.** The world model and planner are fixed and only the cost changes: latent + $F_\phi$ is the cost of Eq. (5) used in every other table, and $F_\phi$ alone drops the latent term ($w = 1$). Bold marks the better of each pair. Avg.: mean over the five environments.

| Ablation | TwoRoom | Push-T | Reacher | Cube | Scene | Avg. |
|---|---|---|---|---|---|---|
| **LeWM** | | | | | | |
| *One plan from the goal* | | | | | | |
| $F_\phi$ alone ($w = 1$) | **98.3**±0.3 | 89.1±0.9 | 93.5±0.6 | 75.9±2.1 | 68.5±1.5 | 85.0 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | 97.2±0.6 | **95.1**±0.6 | **95.5**±0.7 | **77.1**±1.7 | **71.1**±1.1 | **87.2** |
| *Two plans* | | | | | | |
| $F_\phi$ alone ($w = 1$) | 97.2±0.7 | 30.4±1.9 | 96.5±0.5 | 52.3±2.1 | 52.5±1.5 | 65.8 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | **99.2**±0.3 | **55.7**±1.2 | **96.8**±0.5 | **58.7**±2.0 | **57.9**±1.3 | **73.7** |
| *Three plans* | | | | | | |
| $F_\phi$ alone ($w = 1$) | 99.1±0.3 | 16.0±1.2 | 97.9±0.4 | 67.1±1.9 | 47.9±1.4 | 65.6 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | **99.2**±0.3 | **28.1**±2.0 | **98.4**±0.5 | **70.4**±1.2 | **50.5**±1.3 | **69.3** |
| **PLDM** | | | | | | |
| *One plan from the goal* | | | | | | |
| $F_\phi$ alone ($w = 1$) | 98.7±0.4 | 82.4±1.3 | 94.1±0.8 | **71.1**±2.0 | **61.6**±1.7 | 81.6 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | **98.9**±0.3 | **90.1**±1.2 | **94.5**±1.0 | 68.7±1.4 | 58.4±1.3 | **82.1** |
| *Two plans* | | | | | | |
| $F_\phi$ alone ($w = 1$) | 96.1±1.0 | 27.2±1.4 | 96.9±0.6 | 43.5±2.1 | **46.4**±1.4 | 62.0 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | **97.2**±0.5 | **39.5**±1.8 | **98.0**±0.4 | **48.7**±1.8 | 45.9±1.2 | **65.8** |
| *Three plans* | | | | | | |
| $F_\phi$ alone ($w = 1$) | 94.7±1.3 | 11.5±1.0 | **97.9**±0.2 | 61.6±2.2 | **43.5**±1.1 | 61.8 |
| **Latent + $F_\phi$ (ours,** $w = 0.5$) | **95.7**±1.1 | **17.1**±1.7 | 97.7±0.5 | **66.7**±2.0 | 41.1±1.5 | **63.7** |

**Generalisation suites (WallRandom, PushObj).** The generalisation study of Zhou et al. (2024) retrains the world model on randomised wall and door layouts (WallRandom) and on several block shapes (PushObj) and tests on layouts and shapes never seen in training. We rebuild both suites on the platform with our recipe (10 training epochs, the same planner settings as Table 1) and report, per world model, the planner with its own cost against the same planner + $F_\phi$, where $F_\phi$ is trained on the training layouts or shapes only. WallRandom: 10,240 random-walk episodes over 1,044 layouts; the 81 held-out layouts have wall and door positions outside every training range. PushObj: 20,000 episodes with the shapes T, Z, square and small tee; held-out shapes L, I and +. Figure 3 shows the training and test setups, and Table 3 in the main text gives the training and held-out columns.

uses the standard 25-step windows drawn from held-out episodes on the unseen layouts, 'cross-wall' is the original start-on-one-side, goal-on-the-other test. LeWM's WallRandom world model ($^\ell$) is trained with single-layout batches: every batch of 128 clips is drawn from one of the 1,044 training layouts, so that the batch-level SIGReg regulariser has to be satisfied by the agent's motion rather than by layout identity. Everything else, including the 10-epoch budget, is the shared recipe; the planner and + $F_\phi$ rows share the world model.

**Cross-wall diagnostic.** On the cross-wall test every arm of every 10-epoch board trained with mixed-layout batches is near zero on held-out layouts, and also on training layouts, so for those boards the test measures a limitation of the planners rather than layout generalisation (the single-layout-batch LeWM board of Table 3 is the exception). Pinning a single training layout (the fixed Wall layout) and the same 250 cross-wall trials for every board separates the two levers, training length and cost. The 10-epoch fixed-layout LeWM board crosses in $19.6 \pm 2.0\%$ of trials with its own cost and in $20.0 \pm 7.3\%$ with + $F_\phi$, the 10-epoch PLDM board in $13.2 \pm 3.0\%$ and $11.2 \pm 2.0\%$, the 10-epoch WallRandom LeWM board in $0.4\%$. Trained for 100 epochs (the budget of Zhou et al. (2024)), the fixed-layout board planned with its own cost reaches $0.8$, $14.0$, $13.2$, $19.6$ and $21.2\%$ at epochs 20, 40, 60, 80 and 100, and its mean final distance to the goal stays above 30 of the 37.5 arena units between start and goal: the planner rarely leaves its side of the wall. The same 100-epoch boards planned with + $F_\phi$ cross in $74.3 \pm 3.3\%$ of the identical trials against $19.5 \pm 1.3\%$ for their own cost (three seeds, $74.0$, $78.4$ and $70.4$ against $21.2$, $18.0$ and $19.2$), with a mean final distance of 13 units. The latent-distance cost does not find the door at any training length; the temporal metric does once the world model's rollouts are accurate enough for it to rank them. Table 3 reports the 10-epoch boards of the shared recipe; the 100-epoch WallRandom boards are trained with the same recipe otherwise.

**Cross-room TwoRoom.** A harder change of the task rather than of the pixels: the Two-Rooms evaluation of Sobal et al. (2026) places the start in one room and the goal in the other, so every trial

Table 6: **Same out-of-distribution protocol on every environment (goal 75 steps ahead).** The pixel corruptions are generic wrappers on the rendered frame, applied with identical parameters in every environment (values as in Wang et al., 2026); the reference row runs the same protocol with no corruption, so the first frame is re-rendered live in every row. World model frozen, $F_\phi$ trained online in the corrupted environment, blend cost of Eq. (5). Bold: the better of each pair.

| Condition | TwoRoom | Push-T | Reacher | Cube | Scene | Avg. |
|---|---|---|---|---|---|---|
| in-distribution — LeWM planner | 36.5±2.1 | 21.1±1.3 | 89.2±0.5 | 69.5±1.3 | 47.7±1.0 | 52.8 |
| in-distribution — **+ TEMPO (ours)** | **98.4**±0.3 | **26.1**±1.4 | **97.7**±0.4 | **70.0**±1.0 | **49.5**±1.7 | **68.3** |
| blur — LeWM planner | 25.9±2.2 | 20.7±1.1 | 88.5±0.5 | 68.0±1.6 | **42.0**±1.4 | 49.0 |
| blur — **+ TEMPO (ours)** | **38.1**±2.4 | **26.5**±1.7 | **96.0**±0.6 | **68.9**±1.5 | 40.5±1.5 | **54.0** |
| colour jitter — LeWM planner | 34.1±3.3 | 6.3±1.3 | **24.5**±5.2 | 59.2±4.3 | 41.1±2.1 | 33.0 |
| colour jitter — **+ TEMPO (ours)** | **74.3**±8.0 | **12.8**±3.2 | 23.2±6.5 | **61.5**±4.5 | **42.1**±2.2 | **42.8** |

Table 7: **The same pixel-corruption protocol on the more complex environments of Table 2 (goal 75 steps ahead).** The pixel corruptions are generic wrappers on the rendered frame, applied with identical parameters in every environment (values as in Wang et al., 2026); the in-distribution rows repeat the corresponding cells of Table 2. †Wall episodes end at 50 steps, so its goal is 40 steps ahead; cross-room uses its 150-step budget. World model frozen, $F_\phi$ trained online in the corrupted environment, blend cost of Eq. (5). Bold: the better of each pair.

| Condition | Wall† | TwoRoom cross-room | PointMaze M | PointMaze L | Ant-U | Avg. |
|---|---|---|---|---|---|---|
| in-distribution — LeWM planner | 57.7±2.4 | 17.1±1.1 | 10.8±0.8 | 0.3±0.2 | 28.1±1.6 | 22.8 |
| in-distribution — **+ TEMPO (ours)** | **65.9**±2.2 | **74.5**±1.7 | **31.7**±1.0 | **6.5**±1.0 | **34.0**±2.1 | **42.5** |
| blur — LeWM planner | 52.1±2.8 | 8.9±1.4 | 10.4±0.9 | 0.1±0.1 | **12.5**±1.4 | 16.8 |
| blur — **+ TEMPO (ours)** | **60.9**±2.4 | **23.3**±2.5 | **17.9**±1.7 | **4.1**±1.0 | 12.4±1.5 | **23.7** |
| colour jitter — LeWM planner | 56.4±2.6 | 16.1±1.5 | 9.6±1.0 | 1.6±0.6 | 17.7±1.7 | 20.3 |
| colour jitter — **+ TEMPO (ours)** | **68.0**±2.4 | **59.7**±6.9 | **10.4**±1.3 | **1.7**±0.7 | **19.7**±1.3 | **31.9** |

must pass through the door (50 identical start and goal pairs per group, five groups per seed, three seeds, the same for every arm), and the planner must commit to a route that moves away from the goal before it approaches it. Success is the environment's own criterion, within 16 px of the goal, and the world models, planner and $F_\phi$ are those of the TwoRoom column of Table 1, with $F_\phi$ not retrained. The cross-room column of Table 2 reports it at budgets of 50, 100 and 150 steps; the 50-step budget is tight, since the median successful traversal takes 48 steps, which is why both arms score lowest there. The gain is the largest in this paper: the latent distance alone cannot express a route that first increases it, and the temporal distance can.

**Pixel corruptions.** We keep the Table 1 world models unchanged and apply one of two generic corruptions to every rendered frame at test time (blur, colour jitter; parameters as in Wang et al., 2026); the world model is frozen, $F_\phi$ is trained in the training environment and kept fixed during evaluation, and the reference row runs the same setup with no corruption. Table 6 gives the per-environment numbers (goal 75 steps ahead, 5 groups $\times$ 3 seeds). *TEMPO* lifts the average under every condition, from $52.8$ to $68.3$ with no corruption, $49.0$ to $54.1$ under blur and $33.0$ to $42.8$ under colour jitter, and it doubles the planner on TwoRoom under colour jitter. Figure 5 shows the same runs as success over the episode. Table 7 applies the same corruptions to the environments of Table 2; *TEMPO* raises the average under every condition.

**Factor shifts.** We follow an established factor-shift protocol (Wang et al., 2026): the first frame is rendered live under the shift, the goal frame stays the clean dataset frame, $F_\phi$ is trained in the training environment and kept fixed during evaluation, and the world model stays frozen; goal 75 steps ahead. Table 8 reports geometry, dynamics and camera shifts per environment. Where the shift leaves the scene legible to the frozen encoder, *TEMPO* keeps its gain: on Push-T a larger agent goes from $18.3$ to $26.3$, on TwoRoom a larger agent goes from $36.4$ to $94.9$ and a moved door from $9.9$ to $30.3$, and on Reacher a heavier arm goes from $89.7$ to $97.2$ (Table 8). On Cube, where *TEMPO* remains competitive with the planner without a shift, it remains so under the shifts as well.

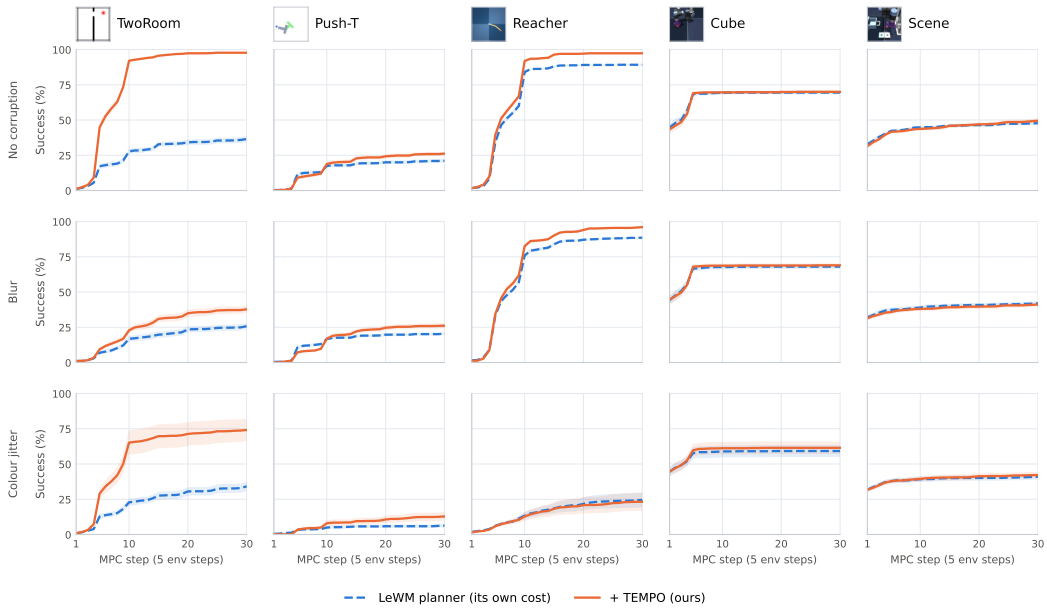


Figure 5: **Test-time pixel corruptions: cumulative success versus MPC step, goal 75 steps ahead (30 steps = budget 150).** Rows: condition; columns: environment. Dashed blue: the LeWM planner with its own cost; orange: the same planner + *TEMPO* (ours), on identical windows.

Table 8: **Factor shifts on Push-T, TwoRoom, Reacher, Cube and Scene, goal 75 steps ahead.** Protocol of Wang et al. (2026): the first frame is rendered live under the shift, the goal frame stays the clean dataset frame, the world model is frozen and $F_\phi$ is trained online in the shifted environment. Shifts change the geometry, the dynamics or the camera. Success (%), LeWM planner and the same planner with *TEMPO*; mean over three seeds; bold: the better of the pair.

| Environment | Shift | Planner | + *TEMPO* |
|---|---|---|---|
| Push-T | none (in distribution) | 21.1 | **26.1** |
| | larger agent | 18.3 | **26.3** |
| TwoRoom | none (in distribution) | 36.5 | **98.4** |
| | door moved | 9.9 | **30.3** |
| | larger agent | 36.4 | **94.9** |
| Reacher | none (in distribution) | 89.2 | **97.7** |
| | heavier arm | 89.7 | **97.2** |
| Cube | none (in distribution) | 69.5 | **70.0** |
| | larger cube | 66.4 | **67.6** |
| | camera tilted | 60.8 | **61.1** |
| Scene | none (in distribution) | 47.7 | **49.5** |
| | larger cube | 46.5 | **48.4** |
| | camera tilted | 35.9 | **36.9** |

## H ADDITIONAL RESULTS ON ROBOCASA KITCHEN TASKS

We additionally evaluate on two RoboCasa kitchen tasks with a mobile manipulator (Table 9), Pick-and-Place and Kitchen-Nav, with world models trained on our own scripted-expert collections. These tasks are harder than the core suite for every planner, and *TEMPO* improves both planners on both tasks at every goal distance. Details on the tasks and datasets can be found in Appendix C.

Table 9: **Planning results for offline world models on two RoboCasa kitchen tasks, one to three plans from the goal.** Success rate as a fraction of episodes. Bold marks the better of each pair.

| Planner | Pick-and-Place (SR ↑) | Kitchen-Nav (SR ↑) |
|---|---|---|
| **LeWM** | | |
| one plan | 0.32 | 0.16 |
| **+ TEMPO (ours)** | **0.36** | **0.22** |
| two plans | 0.20 | 0.19 |
| **+ TEMPO (ours)** | **0.21** | **0.27** |
| three plans | 0.14 | 0.23 |
| **+ TEMPO (ours)** | **0.16** | **0.26** |
| **PLDM** | | |
| one plan | 0.27 | 0.14 |
| **+ TEMPO (ours)** | **0.32** | **0.26** |
| two plans | 0.16 | 0.18 |
| **+ TEMPO (ours)** | **0.18** | **0.22** |
| three plans | 0.10 | 0.20 |
| **+ TEMPO (ours)** | **0.12** | **0.24** |

## I ANALYSIS OF THE TEMPORAL DISTANCE

**Goal distance measured in $k$ plans.** Every planner optimises one short plan at a time: $H = 5$ action blocks of five environment steps, 25 steps in all. The CEM search picks the best 25-step action sequence, executes it in full, re-encodes the new observation and plans again (receding-horizon MPC). A goal one plan away is the frame 25 steps later in a recorded trajectory, so a single plan can reach it. A goal three plans away is 75 steps out, so no single plan reaches it and the planner must chain at least three plans, replanning after each, within its step budget. Nothing about the planner changes across goal distances; only the distance to the goal does.

**Why the flat planner stalls at range.** The planner never scores the goal itself. In each round it can only choose the plan whose imagined end state, 25 steps ahead, scores best under its cost. The published cost asks of every candidate, "how similar does your end state look to the goal frame?" It behaves much like a straight-line distance: it prefers end states that point directly at the goal. One plan from the goal, that question and "how close are you to finishing?" have the same answer. Three plans away they do not. On TwoRoom the path must bend through the door, and in a maze it bends around walls, so the shortest route first moves away from the goal. The latent distance scores that detour as worse, and the imagined state most similar to the goal frame is the one pressed against the wall on the goal's side, dozens of steps from the goal through the door; the planner drives into the wall and stalls (Appendix J). Equivalently, the latent distance defines a convex basin around the goal, whereas the task is not convex: walls split the basin, and a planner that descends it greedily is trapped in a local minimum.

**What *TEMPO* changes.** $\|F_\phi(\hat{z}_{t+H}) - F_\phi(G)\|$ asks instead "how many steps are you from the goal?" $F_\phi$ is trained on real trajectories to predict how many steps separate two states, so its distance bends around walls: trajectories that reached the other room went through the door, so a state next to the door, even one farther from the goal in a straight line, receives a small step count, and a state pressed against the wall on the wrong side receives a large one. The result behaves like a shortest-path (geodesic) distance, monotone along the path in a way raw latent similarity is not. Descending it leads through the door rather than into the wall; each 25-step plan now picks an end state that makes real progress, and three such plans chain to the goal. Everything else stays as published, the frozen model, the CEM search, the 25-step plans executed in full and replanned in closed loop, so the gain over the flat planner isolates the effect of the objective alone.

**$F_\phi$ is a distance, not a clock.** Since $F_\phi$ is trained on pairs from the same episode, it could in principle learn the frame index, $F_\phi(z_t) \approx t$. Table 10 rules this out. States with the same frame

index but from different episodes lie farther apart than states 25 steps apart in one episode, the frame index is no more decodable from $F_\phi$ than from the raw latent, and in the navigation environments the $F_\phi$-distance between episodes tracks the agent's position (Spearman $\rho = 0.93$ on TwoRoom, 0.67 on Wall).

Table 10: $F_\phi$ does not encode the frame index. Median $F_\phi$-distance (units of 25 steps) between states with the same frame index in different episodes, against states 25 steps apart in the same episode, and $R^2$ of a linear probe predicting the frame index from $F_\phi(z)$ and from the raw latent $z$.

| | $F_\phi$-distance | | $R^2$ of frame index | |
|---|---|---|---|---|
| **Environment** | **same index, other episode** | **25 steps, same episode** | **from $F_\phi(z)$** | **from $z$** |
| TwoRoom | 1.48 | 1.03 | 0.11 | 0.13 |
| Push-T | 2.27 | 1.09 | 0.54 | 0.73 |
| Reacher | 2.27 | 1.95 | 0.01 | 0.03 |
| Cube | 1.63 | 1.07 | 0.03 | 0.13 |
| Scene | 2.45 | 1.03 | 0.21 | 0.32 |
| Wall | 1.09 | 0.70 | 0.07 | 0.10 |

**$F_\phi$ is goal-agnostic and generalises within an environment.** $F_\phi$ is trained per environment on the world model's latent, as the world model itself is, but its loss uses no goal, reward or success label. It learns the number of steps between arbitrary pairs of states of the same episode, and the goal is supplied only at test time, so it encodes the environment's temporal structure rather than which states are close to a particular goal. The same recipe is used unchanged across all environments. Within an environment $F_\phi$ also generalises beyond its training data. On PointMaze medium and large and on Ant-U, where the world model and $F_\phi$ are trained on 90% of the episodes, it measures step counts as accurately on the held-out 10% as on training episodes (Table 11), and without retraining it transfers to unseen configurations (Table 3) and to start and goal states from different episodes (cross-room, Table 2).

Table 11: **$F_\phi$ generalises to held-out episodes.** Spearman $\rho$ between the $F_\phi$-distance and the step gap $k$, and mean absolute error of the $F_\phi$-distance against its target $k/25$ (units of 25 steps), over about 24,000 same-episode pairs with $k \leq 75$ (LeWM, mean over three seeds).

| | Training episodes | | Held-out episodes | |
|---|---|---|---|---|
| **Environment** | $\rho$ | **error** | $\rho$ | **error** |
| PointMaze M | 0.83 | 0.28 | 0.83 | 0.28 |
| PointMaze L | 0.99 | 0.06 | 0.99 | 0.06 |
| Ant-U | 0.56 | 0.56 | 0.51 | 0.59 |

## J QUANTIFYING THE COLLAPSE OF THE LATENT DISTANCE

Executing many candidate plans from the same state and recording which reach the goal, we find that three plans from the goal the latent distance predicts where a plan ends but not whether it succeeds: its best-scored plans succeed far less often than the best ones in hindsight, whereas one plan away the same cost works. The latent distance is a good local cost and a poor global one, and the temporal distance supplies the coordinate it lacks at range. In this section we measure why the flat planner fails at range, by asking what its cost actually ranks. A failure can only be diagnosed where it happens, so we study the two environments where the flat LeWM planner collapses between one and three plans from the goal (Table 1: TwoRoom from 88 to 36%, Push-T from 94 to 22%), and use Reacher, where it does not collapse (95 to 90%), as the control. Cube barely drops (75 to 69%) and would only repeat the control, and Wall has no three-plan setting. The one-plan rows show the same cost close to the goal, where it still works. For 60 windows per cell we executed 50 CEM candidates from the same state and recorded the predicted latent distance, the realised end distance and eventual success (Table 12; protocol below).

**Three plans away the cost ranks where a plan ends, not whether it succeeds.** At offset 75 the planner's latent distance correlates with the realised end distance at $\rho = 0.66$ (Push-T), 0.66 (TwoRoom) and 0.85 (Reacher), and with eventual success at $+0.02$, $-0.07$ and $+0.22$: the predictor knows where a plan goes, and on the two collapse environments that says nothing about which

Table 12: **What the latent distance ranks.** We diagnose the two tasks where the flat planner collapses three plans from the goal (TwoRoom from 88 to 36%, Push-T from 94 to 22%); Reacher, where it does not collapse, is the control, and the one-plan rows show the same cost working close to the goal. 50 executed action sequences per window, scored by the planner's own latent distance to the target; $\rho$(score, realised distance) and $\rho$(score, eventual success) over the sequences of a window; top-5 = success of the five best-scored sequences, oracle = best five in hindsight.

| benchmark / offset | windows | window success | $\rho$(score, realised dist) | $\rho$(score, success) | top-5 success | oracle top-5 |
|---|---|---|---|---|---|---|
| PushT / 25 | 59 | 0.74 | +0.83 | +0.32 | 0.89 | 1.00 |
| TwoRoom / 25 | 48 | 0.80 | +0.49 | +0.21 | 0.84 | 0.91 |
| PushT / 75 | 60 | 0.28 | +0.66 | +0.02 | 0.29 | 0.65 |
| Reacher / 75 | 60 | 0.95 | +0.85 | +0.22 | 0.97 | 0.99 |
| TwoRoom / 75 | 60 | 0.42 | +0.66 | −0.07 | 0.41 | 0.75 |

plan leads to the goal. The five best-scored plans succeed in 29% of Push-T windows against 65% for the hindsight best five, and in 41 against 75% on TwoRoom. Among pairs of executed candidates with different outcomes, the one predicted closer to the target is the one that fails in 46% of pairs on Push-T and 55% on TwoRoom, no better than a coin.

**One plan away the same cost works.** At offset 25 the correlation of the latent distance with success is +0.32 on Push-T and +0.21 on TwoRoom, and the five best-scored plans succeed in 89 and 84% of windows against 100 and 91% for the hindsight best. The latent distance is a good local cost and a poor global one. The learned temporal distance supplies the coordinate the latent lacks at range, the number of steps a plan's end state still is from the goal: it turns 36 into 99 on TwoRoom, and it stops exactly where that number is not the task, on Push-T.

**Action ranking (Table 12).** For 60 windows per cell (20 per seed), 50 candidate action sequences of the CEM (the final population) are each executed from the same state for 25 steps and then followed by the flat pursuit of the goal, scored by the planner's own latent distance. Correlations are Spearman with average ranks over the sequences of a window (a binary outcome has many ties), pooled over windows. Among success-discordant pairs of sequences (one succeeds, one fails) the sequence predicted closer to the target is the failing one in 46.2% of 13,505 pairs on Push-T, 24.3% of 3,784 on Reacher and 55.0% of 23,068 on TwoRoom (offset 75).

**The same plans scored by *TEMPO* (Table 13).** Three plans from the goal, on all five environments, we execute 50 candidate plans per window from the same state and score each with both costs. Scored by *TEMPO*'s cost, the ranking tracks eventual success more closely on TwoRoom ($\rho$ from −0.08 to +0.17, the five best-scored plans succeeding in 35 against 26%), Push-T (+0.06 to +0.13) and Reacher (+0.25 to +0.33, 98 against 95%), and remains competitive on Cube and Scene.

Table 13: **Which executed plans each cost ranks first, three plans from the goal.** For 20 windows per seed, 50 candidate plans are executed from the same state for 25 steps and then finished by the flat planner; each candidate is scored by LeWM's latent distance and by *TEMPO*'s cost. $\rho$: Spearman correlation between the (negated) cost and eventual success; top-5: success rate (%) of the five best-scored candidates; best five: the success rate of the five candidates that actually succeed most. Mean over three seeds; bold: the better of the pair.

| | $\rho$(cost, success) | | top-5 success | | |
|---|---|---|---|---|---|
| **Environment** | latent | *TEMPO* | latent | *TEMPO* | best five |
| TwoRoom | -0.08 | **+0.17** | 26 | **35** | 57 |
| Push-T | +0.06 | **+0.13** | 30 | 30 | 51 |
| Reacher | +0.25 | **+0.33** | 95 | **98** | 99 |
| Cube | +0.39 | +0.38 | 32 | 32 | 42 |
| Scene | +0.02 | **+0.04** | 18 | **21** | 60 |