PLAN TOWARD SUCCESS, NOT SIMILARITY: SUCCESS-ALIGNED COSTS ON FROZEN LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

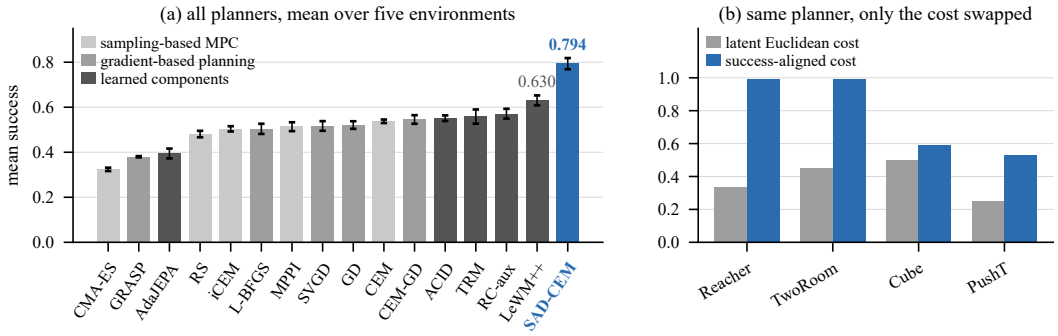


Figure 1: Overview. (a) Mean success over five goal-reaching environments of SAD-CEM and the fifteen planners of Table 1 on the same frozen LeWM checkpoints, with error bars giving the SD over five seeds. (b) One CEM planner with one head and re-planning after every block, with only the cost swapped (validation split, same frozen LeWM checkpoints, mean over seeds 0–2, Table 3).

ABSTRACT

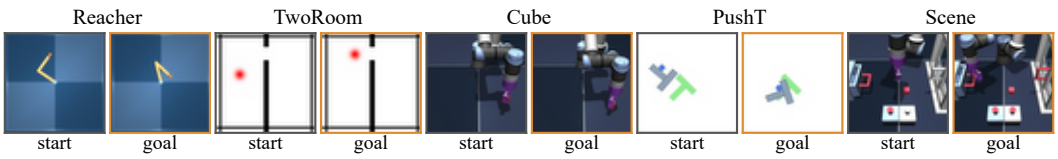
A frozen latent world model serves control through test-time search, where an optimizer seeks the plan of lowest cost. We argue that the cost is the main lever, and that latent and temporal distances measure the wrong quantity, embedding similarity or elapsed steps. We propose success-aligned distances, heads that regress the logged time until the task predicate first holds, a label that is zero exactly on successful states. SAD-CEM optimizes a pessimistic ensemble of these heads toward the encoded goal with cross-entropy search, and consults generated waypoints when its plan leaves the generated paths. With frozen LeWM backbones, SAD-CEM has the highest or tied-highest mean success on each of five goal-reaching environments, and a five-environment mean of 0.794 against 0.630 for the strongest of fifteen external planners. The success-aligned cost alone, without generated waypoints, already reaches 0.742, while a step-gap label or one head in place of the ensemble lowers success significantly. Code and demos are available at `anonymous.4open.science/r/sad-cem`.

1 INTRODUCTION

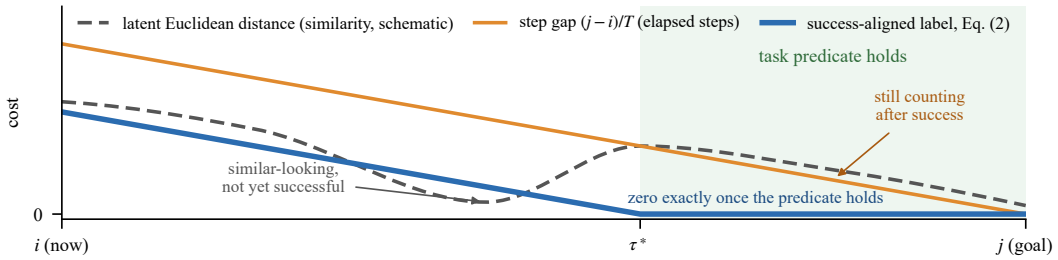
Planning with a learned model ranks imagined futures by a cost and hands that ranking to an optimizer that searches for the lowest-cost actions (Chua et al., 2018; Williams et al., 2017). When the world model is trained once on offline trajectories and then frozen, every later gain or loss comes from how this search is designed (Maes et al., 2026; Zhou et al., 2025; Terver et al., 2026). An optimizer presses on whatever cost it receives, including the errors that a learned cost makes on inputs unlike the pairs it was trained on (Yu et al., 2020). Decision analysis names the resulting bias the optimizer curse, an optimism of the chosen estimate that grows with the uncertainty of the compared estimates (Smith & Winkler, 2006). We therefore read test-time planning first as a question of what the optimized cost measures and how accurate it is on the queries the search compares.

Latent Euclidean distance, the default cost of these planners, measures embedding geometry, which can misrank goals that lie far apart in reachable time (Li et al., 2026b; Bai & Xiong, 2026). Learned

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.



(a) start and goal frames



(b) three costs along one logged trajectory

Figure 2: Tasks and the quantity each cost measures. (a) Start frame and goal frame, 50 steps apart, of the first validation goal of each environment, taken from the offline datasets. As in LeWM, a goal is a later state of the same logged trajectory (Maes et al., 2026), so PushT asks for the block pose of the goal frame rather than the drawn green target. (b) Illustration along a trajectory whose task predicate first holds at step τ^* before the goal step j. The step gap and the success-aligned label follow their definitions, and the latent Euclidean curve is a schematic of a similarity cost that dips at a similar-looking failing state and stays positive on states where the predicate already holds.

temporal and reachability costs repair this ranking, yet they hand it to a full-strength search that optimizes their errors along with their signal (Li et al., 2026a;b; Bai & Xiong, 2026). The TRM authors report that contact-rich PushT exposes limits of their temporal cost (Li et al., 2026a). Hierarchical planners replace a distant goal with nearer generated subgoals, which LeWM++ tracks up to the final approach although no encoder produced them (Zhang et al., 2026; LeWM++, 2026). Other lines change the consistency term, the predictor or the optimizer, and keep the latent distance of the base planner as their cost (Seo et al., 2026; Wang et al., 2026; Psenka et al., 2026).

Seen through this lens, the optimized cost should measure progress toward task success rather than embedding similarity or step counts. A step count keeps growing after the goal already holds, while a latent distance can be small for states that fail the task, so neither vanishes exactly on success. Because the cost is optimized over thousands of candidates, it should score queries like its training pairs, with an ensemble to temper the errors of any one member (Lakshminarayanan et al., 2017). Logged trajectories with a task predicate contain this signal, the time until the predicate first holds. To our knowledge, no planner on a frozen latent world model trains its cost on this time and optimizes it toward the encoded goal (Li et al., 2026a; LeWM++, 2026; Seo et al., 2026).

We propose success-aligned distances, heads that regress the logged time until the task predicate first holds, and SAD-CEM, which optimizes a pessimistic ensemble of them toward the encoded goal with cross-entropy search. Trained on encoded state pairs, the heads learn a label that is zero exactly on successful states. SAD-CEM also consults waypoints from a pretrained path generator when its plan leaves the generated paths, a component detailed in Appendix A. A selection bound motivates optimizing only a cost that is accurate on the queries it ranks, since the optimism of a search grows with that error. With this cost alone, SAD-CEM (direct mode), which omits the waypoint branch, has a higher five-environment mean than every external planner. Replacing the success-aligned label by a step gap, or the ensemble by one head, lowers the success of SAD-CEM significantly. SAD-CEM has the highest or tied-highest mean success on each of Reacher, TwoRoom, Cube, PushT and Scene (Table 1, Figure 1a), with a five-environment mean of 0.794 against 0.630 for the runner-up.

2 Related Work

Planning with frozen latent world models. LeWM trains a joint-embedding predictive world model end to end from pixels, while DINO-WM predicts future patch features of a pre-trained DI-

NOv2 encoder (Maes et al., 2026; Zhou et al., 2025). The cross-entropy method is the standard planner in this line, with iCEM, MPPI and the gradient-based GRASP as common alternatives (Rubinstein, 1999; Pinneri et al., 2021; Williams et al., 2017; Psenka et al., 2026). PETS and TD-MPC2 pair learned dynamics with sampling-based planning in online reinforcement learning, where the model keeps training alongside the planner (Chua et al., 2018; Hansen et al., 2024). AdaJEPA also updates the predictor during deployment from each observed transition, whereas SAD-CEM keeps the world model fixed throughout every episode and changes only how it plans (Wang et al., 2026).

Learned planning costs and subgoals. TRM trains a temporal pairwise cost on logged trajectories, raising LeWM success on TwoRoom from 0.070 to 0.967 in its own evaluation (Li et al., 2026a). RC-aux adds budget-conditioned reachability supervision, while Temporal-Distance JEPA mines a directed temporal cost with cross-trajectory pairs as heuristic negatives (Li et al., 2026b; Bai & Xiong, 2026). These costs are supervised by the steps between logged states, whereas the success-aligned label is the time until the task predicate first holds (Li et al., 2026a;b; Bai & Xiong, 2026). ACID folds an inverse-dynamics residual into the planning cost, asking that the action inferred from a predicted transition recover the conditioned action (Seo et al., 2026). Hierarchical planning passes latents of a long-horizon model to a low-level planner as subgoals, motivated by the greedy shortcuts of local optimizers (Zhang et al., 2026). LeWM++ tracks the waypoints of a flow-matching path generator with a sampling-based search over the world model (LeWM++, 2026). In SAD-CEM, generated latents serve as candidates for one selection, while the learned cost is optimized toward the encoded goal alone (LeWM++, 2026; Zhang et al., 2026).

Model exploitation, pessimism and selection bias. Offline model-based reinforcement learning has long observed that policy optimization exploits regions in which the learned model is inaccurate (Yu et al., 2020). MOPO penalizes rewards by ensemble uncertainty, whereas the PETS authors note that ensemble variance can take arbitrary values on out-of-distribution inputs (Yu et al., 2020; Chua et al., 2018). Decision analysis formalizes this bias as the optimizer curse, which affects any choice among noisy estimates (Smith & Winkler, 2006). SAD-CEM applies both arguments inside test-time search, using an ensemble where the learned distance is optimized and candidates drawn near the data where it only selects among them (Yu et al., 2020; Smith & Winkler, 2006).

3 METHOD

3.1 PRELIMINARIES

Frozen latent world model. A latent world model pairs an encoder E, which maps an image frame to a latent in $\mathbb{R}^D$, with a predictor f that advances a short latent history by one action block (Maes et al., 2026; Zhou et al., 2025). Here a block holds five environment steps, with z_t the latent of the current frame o_t. For a plan $\mathbf{a} = (a_1, \dots, a_H)$ of H blocks, the predicted block latents are $\hat{z}_b = f(\hat{z}_{<b}, a_b)$ for $b = 1, \dots, H$, starting from the encoded history at time t. Frozen means that E and f keep identical parameters throughout every run, so a planner changes only how it searches.

Goal-reaching evaluation. A goal is an observation o_g with encoding $z_g = E(o_g)$, together with the simulator state s_g that the environment uses to judge success. Each environment defines a success predicate $\chi(s, s_g) \in \{0, 1\}$ on simulator states, and an episode succeeds if χ holds within T steps.

Sampling-based model-predictive control. Model-predictive control plans a short horizon of actions through the model, executes the first part, and plans again from the next observation (Williams et al., 2017; Chua et al., 2018). The cross-entropy method (CEM) samples N_s plans from a Gaussian, scores them with a cost, and refits the Gaussian to the k best plans for n_{it} iterations (Rubinstein, 1999). The planners here execute the first block of the final mean, warm-starting the next search from the remaining blocks of that plan. The latent Euclidean cost of LeWM-style planners with a min-over-horizon rule is

$$C_{\mathrm{L2}}(\mathbf{a}; y) = \min_{b \le H} \|\hat{z}_b - y\|_2^2, \tag{1}$$

where the target y is either the goal latent z_g or a waypoint latent. We call $N = N_s\, n_{\mathrm{it}}$ the *search pressure* of one decision, the number of candidate plans the search compares before it acts.

Learned costs, training support and pessimism. Learned parts are trained on a pool $\mathcal{D}$ of logged trajectories that excludes every episode of the validation and final splits. A learned cost $d(x, y)$ regresses a distance label on pairs (z_i, z_j) of encoded states from one trajectory with $0 < j - i \leq \Delta_{\max}$ (Li et al., 2026a). We call the distribution of these pairs the *training support* of the cost, the region in which data constrain its errors. A search that compares many candidates tends to pick one on which the cost errs low, a bias that grows with the number of candidates (Smith & Winkler, 2006). An ensemble of M heads trained from different initializations and data draws exposes part of this error through its disagreement (Lakshminarayanan et al., 2017). A pessimistic value adds κ times the spread to the mean, penalizing candidates on which heads disagree (Yu et al., 2020).

Conditional path generators and waypoints. A conditional generative model samples a future given a context, for instance by flow matching, which regresses a velocity field that carries noise to data (Lipman et al., 2022). Trained on logged trajectories, such a generator draws latent paths that the data make plausible from the current history toward a goal, so a handful of samples outlines the local support of the data. Their intermediate latents can also serve as *waypoints*, nearer targets that a planner tracks in place of a distant goal (Zhang et al., 2026). A direct search queries $d(\hat{z}_b, z_g)$, a predicted latent against an encoded goal, the query type closest in form to the encoded training pairs. A generated waypoint w is a latent that no encoder produced, so a query $d(\cdot, w)$ differs more in form from the training pairs than a query toward z_g, which motivates the design below.

3.2 Proposed Method

SAD-CEM keeps the frozen model and the CEM search of Section 3.1, with Figure 3 summarizing one decision (Algorithm 1 in Appendix A). Its core is a success-aligned cost that the search optimizes toward the encoded goal, and a secondary branch limits the same cost to comparing candidates that a pretrained path generator builds from the current state when the plan leaves its paths.

Success-aligned distance (SAD). Optimizing toward the goal needs a cost that measures progress toward task success and whose training pairs resemble the direct query, a predicted state against a later encoded state. For a pair (i, j) of steps in one trajectory of $\mathcal{D}$ with states s_i, s_j, the success-aligned label is

$$y_{ij} = \frac{1}{T} \min\{\tau \in \{0, \dots, j - i\} : \chi(s_{i+\tau}, s_j) = 1\}, \tag{2}$$

the data time until the task predicate first holds on the logged states, zero exactly when the predicate already holds at step i. A head $d_m(z_i, z_j)$ with the cost-head architecture of TRM (Li et al., 2026a) regresses y_{ij} on encoded latents under a smooth-ℓ_1 loss. Each head trains for 100,000 updates at batch 256 with the learning rate of TRM, on two million pairs with $\Delta_{\max} = 64$ drawn once from $\mathcal{D}$.

Pessimistic ensemble. Predicted states still drift from encoded ones, so the direct cost uses the pessimistic value of Section 3.1 over heads that differ in their pair draws and initializations, with the weight κ a hyperparameter (Appendix I). The planner scores a latent x against a target y by

$$U(x, y) = \mathrm{mean}_{m \leq M}\, d_m(x, y) + \kappa\, \mathrm{std}_{m \leq M}\, d_m(x, y), \qquad M = 5,\ \kappa = 1. \tag{3}$$

In *direct mode* a CEM search with the size of the CEM row of Table 1 minimizes $C_{\mathrm{D}}(\mathbf{a}) = \min_{b \leq H} U(\hat{z}_b, z_g)$ over $H = 5$ blocks, warm-started and re-planned after every block. SAD-CEM (direct mode) denotes the planner that runs this solve alone, without the branch below.

Waypoint branch. When $U(z_t, z_g) > \rho = 10/T$, a goal beyond the ten-step horizon of the reused LatentPathFlow generator of LeWM++, the planner draws $n = 8$ latent paths from the current state (LeWM++, 2026). It then tests whether the first predicted block of the direct plan lies among them. If it does not, the same ensemble picks the waypoint $w^\star = \arg\min_{j \leq n} U(b_j, z_g)$ among the ten-step path ends b_j, and a two-block CEM tracks it with the latent L_2 cost of Equation (1). Appendix A gives the test, the algorithm, the formal results of this branch and all of its ablations.

Formal results. The results below motivate the success-aligned ensemble cost, with proofs in Appendix B under named error models, and Appendix A states the results of the waypoint branch.

Proposition 1 (A choice is anchored by its candidate set)**.** *Let candidates $x_1, \dots, x_N$ have true costs c_i and estimated costs $\hat{c}_i = c_i + \varepsilon_i$, where every ε_i is zero-mean and σ-sub-Gaussian, with*

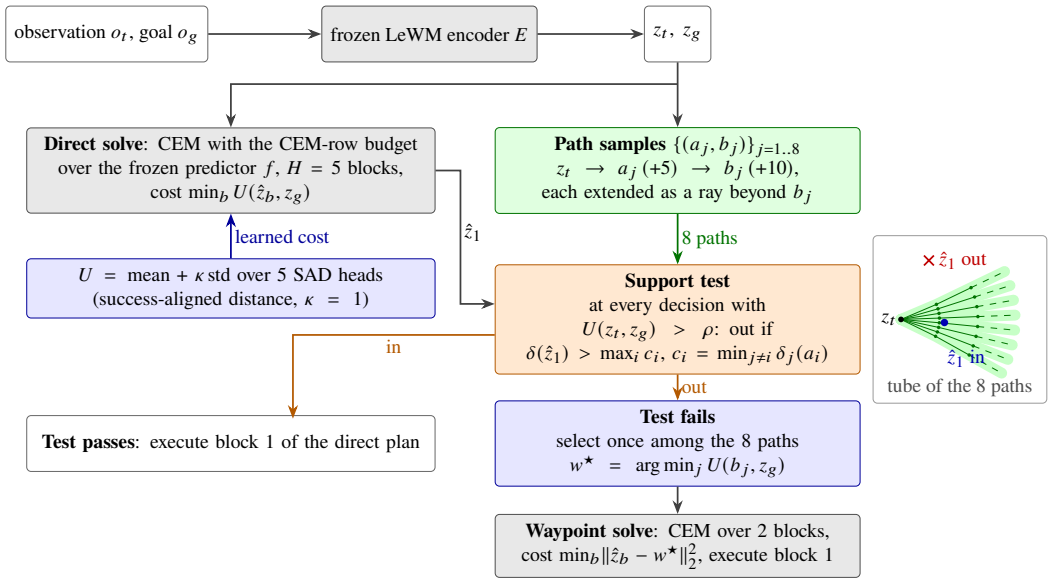


Figure 3: SAD-CEM. The direct solve optimizes the pessimistic success-aligned distance toward the encoded goal, the query closest to its training pairs. When the planned first block leaves the sampled latent paths, the same distance chooses the waypoint for the next block among them. Grey marks frozen parts and plain searches, blue the learned costs, green the path samples and orange the support rule. The test quantities δ, a_j and c_i follow Equation (6) in Appendix A.

arbitrary dependence between candidates. If the candidate set does not depend on the errors and $\hat{\imath} \in \arg\min_i \hat{c}_i$*, then (i) the selection optimism and (ii) the true cost of the choice satisfy*

$$\mathbb{E}\big[c_{\hat{\imath}} - \hat{c}_{\hat{\imath}}\big] \le \sigma\sqrt{2\ln N}, \qquad \mathbb{E}\big[c_{\hat{\imath}}\big] \le \frac{1}{N}\sum_{i=1}^{N} c_i + \sigma\sqrt{2\ln N}. \tag{4}$$

For i.i.d. Gaussian errors and equal costs, the left side of (i) equals $\sigma\sqrt{2\ln N}\,(1+o(1))$ *as* $N \to \infty$.

The bound of part (i) grows with the error σ of the learned cost on the compared candidates, reaching a nominal $4.38\,\sigma$ for the 15,000 candidates of a 500-sample CEM. The direct solve compares predicted states against the encoded goal, so it needs a small error on these queries. The heads train on encoded pairs of this form, and the ensemble penalizes the part of the remaining gap on which they disagree. This nominal value guides the design without bounding CEM itself, whose later samples depend on earlier cost estimates. Part (ii) serves the waypoint branch (Appendix A).

Lemma 1.1 (Ensemble spread cannot see a shared error)**.** *Let every head satisfy* $d_m(x, y) = \bar{d}(x, y) + \eta_m(x, y)$*, where* $\bar{d}$ *is shared by all heads and* η_m *is the deviation of head* m*. Then* $\mathrm{std}_m\, d_m = \mathrm{std}_m\, \eta_m$ *at every input, and for a predicted latent* $\hat{z}$ *of a true latent* z^**, with* $\Delta q = q(\hat{z}, y) - q(z^*, y)$ *for any function* q *of the input pair,*

$$\Delta U = \Delta\bar{d} + \Delta\,\mathrm{mean}_m\, \eta_m + \kappa\,\Delta\,\mathrm{std}_m\, \eta_m. \tag{5}$$

The part $\Delta\bar{d}$ *that moves all heads enters* U *with weight one and leaves the spread unchanged.*

In the terms of Proposition 1, averaging and the spread penalty act on the part of ε on which heads disagree, while the shared part motivates querying the heads toward the encoded goal.

4 Experiments

4.1 Experimental Settings

World models and goals. Every method plans on one frozen world model per environment, using the public LeWM checkpoints (Maes et al., 2026) and, for Scene, a LeWM that we train with the

Table 1: Planners on frozen LeWM world models, success rate as mean ± SD over 5 seeds for goals 50 steps ahead with a 50-step budget. Bold marks the column best with ties, and Mean averages the five environments within each seed. ▼/▲: significantly below/above SAD-CEM ($p < 0.05$, two-sided paired t-test over seeds). Only AdaJEPA adapts its world model, updating the last predictor layer at test time. LeWM++ (no ACP) runs the LatentPathFlow waypoints and min-over-horizon search of LeWM++ without its action chunk prior, and RC-aux trains its reachability head on the shared frozen checkpoint. †A LeWM trained by us with the official recipe (Appendix K).

Method	Reacher	TwoRoom	Cube	PushT	Scene†	Mean
SAD-CEM (ours)	$\mathbf{0.988 \pm 0.011}$	$\mathbf{0.976 \pm 0.017}$	$\mathbf{0.680 \pm 0.058}$	$\mathbf{0.696 \pm 0.050}$	$\mathbf{0.628 \pm 0.059}$	$\mathbf{0.794 \pm 0.025}$
Sampling-based MPC						
CEM	$0.788 \pm 0.023^{\blacktriangledown}$	$0.500 \pm 0.024^{\blacktriangledown}$	$0.520 \pm 0.024^{\blacktriangledown}$	$0.304 \pm 0.026^{\blacktriangledown}$	0.576 ± 0.033	$0.538 \pm 0.008^{\blacktriangledown}$
iCEM	$0.792 \pm 0.041^{\blacktriangledown}$	$0.616 \pm 0.022^{\blacktriangledown}$	$0.496 \pm 0.017^{\blacktriangledown}$	$0.224 \pm 0.033^{\blacktriangledown}$	$0.392 \pm 0.018^{\blacktriangledown}$	$0.504 \pm 0.012^{\blacktriangledown}$
MPPI	$0.716 \pm 0.038^{\blacktriangledown}$	$0.444 \pm 0.033^{\blacktriangledown}$	$0.524 \pm 0.026^{\blacktriangledown}$	$0.264 \pm 0.036^{\blacktriangledown}$	0.620 ± 0.028	$0.514 \pm 0.020^{\blacktriangledown}$
RS	$0.672 \pm 0.052^{\blacktriangledown}$	$0.656 \pm 0.036^{\blacktriangledown}$	$0.500 \pm 0.042^{\blacktriangledown}$	$0.100 \pm 0.014^{\blacktriangledown}$	$0.476 \pm 0.022^{\blacktriangledown}$	$0.481 \pm 0.014^{\blacktriangledown}$
CMA-ES	$0.528 \pm 0.048^{\blacktriangledown}$	$0.152 \pm 0.033^{\blacktriangledown}$	$0.400 \pm 0.037^{\blacktriangledown}$	$0.032 \pm 0.027^{\blacktriangledown}$	$0.508 \pm 0.030^{\blacktriangledown}$	$0.324 \pm 0.007^{\blacktriangledown}$
Gradient-based planning						
GD	$0.664 \pm 0.071^{\blacktriangledown}$	$0.548 \pm 0.018^{\blacktriangledown}$	$0.488 \pm 0.027^{\blacktriangledown}$	$0.344 \pm 0.026^{\blacktriangledown}$	0.560 ± 0.053	$0.521 \pm 0.017^{\blacktriangledown}$
GRASP	$0.568 \pm 0.011^{\blacktriangledown}$	$0.224 \pm 0.026^{\blacktriangledown}$	$0.424 \pm 0.017^{\blacktriangledown}$	$0.152 \pm 0.011^{\blacktriangledown}$	$0.532 \pm 0.030^{\blacktriangledown}$	$0.380 \pm 0.003^{\blacktriangledown}$
CEM-GD	$0.720 \pm 0.049^{\blacktriangledown}$	$0.560 \pm 0.053^{\blacktriangledown}$	$0.508 \pm 0.048^{\blacktriangledown}$	$0.356 \pm 0.033^{\blacktriangledown}$	0.584 ± 0.038	$0.546 \pm 0.019^{\blacktriangledown}$
SVGD	$0.708 \pm 0.046^{\blacktriangledown}$	$0.576 \pm 0.026^{\blacktriangledown}$	$0.488 \pm 0.046^{\blacktriangledown}$	$0.284 \pm 0.033^{\blacktriangledown}$	0.528 ± 0.050	$0.517 \pm 0.021^{\blacktriangledown}$
L-BFGS	$0.644 \pm 0.055^{\blacktriangledown}$	$0.484 \pm 0.030^{\blacktriangledown}$	$0.560 \pm 0.028^{\blacktriangledown}$	$0.280 \pm 0.058^{\blacktriangledown}$	0.552 ± 0.069	$0.504 \pm 0.023^{\blacktriangledown}$
Planners with learned components						
TRM	$0.820 \pm 0.077^{\blacktriangledown}$	$\mathbf{0.976 \pm 0.017}$	$0.324 \pm 0.017^{\blacktriangledown}$	$0.260 \pm 0.073^{\blacktriangledown}$	$0.412 \pm 0.018^{\blacktriangledown}$	$0.558 \pm 0.032^{\blacktriangledown}$
ACID	$0.776 \pm 0.036^{\blacktriangledown}$	$0.516 \pm 0.030^{\blacktriangledown}$	$0.576 \pm 0.009^{\blacktriangledown}$	$0.300 \pm 0.028^{\blacktriangledown}$	0.588 ± 0.061	$0.551 \pm 0.012^{\blacktriangledown}$
LeWM++ (no ACP)	$0.412 \pm 0.041^{\blacktriangledown}$	$0.892 \pm 0.046^{\blacktriangledown}$	0.672 ± 0.046	0.684 ± 0.089	$0.492 \pm 0.061^{\blacktriangledown}$	$0.630 \pm 0.022^{\blacktriangledown}$
RC-aux	$0.832 \pm 0.046^{\blacktriangledown}$	$0.632 \pm 0.041^{\blacktriangledown}$	$0.512 \pm 0.041^{\blacktriangledown}$	$0.284 \pm 0.026^{\blacktriangledown}$	0.596 ± 0.050	$0.571 \pm 0.022^{\blacktriangledown}$
AdaJEPA	$0.360 \pm 0.081^{\blacktriangledown}$	$0.404 \pm 0.043^{\blacktriangledown}$	$0.516 \pm 0.046^{\blacktriangledown}$	$0.124 \pm 0.033^{\blacktriangledown}$	0.568 ± 0.046	$0.394 \pm 0.022^{\blacktriangledown}$

official recipe. Each goal is the state reached 50 steps ahead along a logged trajectory (Figure 2), with success if the environment predicate holds within 50 environment steps. Planner hyperparameters are selected on 50 validation goals and reported on 50 disjoint final goals, both drawn with fixed seeds. The Cube final goals come from episodes disjoint from the training pool and the validation goals, and 8 of these 50 final goals already hold at the first step of their episodes.

Methods and shared conditions. We compare sampling-based MPC (CEM, iCEM, MPPI, random shooting (RS) (Nagabandi et al., 2018; Howell et al., 2022), CMA-ES (Hansen & Ostermeier, 2001)), gradient-based planning (GD, GRASP, CEM-GD (Bharadhwaj et al., 2020), SVGD (Liu & Wang, 2016; Lambert et al., 2021), L-BFGS (Liu & Nocedal, 1989)) and planners with learned components (TRM, ACID, LeWM++, RC-aux, AdaJEPA). In every row, learned modules score states or propose target states, and the executed actions come from test-time search over the frozen world model. All methods in a column share the checkpoint, goals, start states, success predicate and budget, which pairs every comparison within a column. Every learned module, including the SAD heads and the LeWM++ generator, trains for 100,000 updates at batch 256 on one pool without evaluation episodes. The SAD heads read the success predicate on the logged states of this pool, while the planner rows and the other learned costs train without it, and Table 2 gives a control that uses the same predicate. Planner rows use five planner seeds on a fixed backbone, and rows with learned components five independent training seeds (see Appendices K and N).

Statistics. We report success as a fraction, with the mean ± sample SD over five seeds. Paired tests use a one-sided cluster sign-flip test with 20,000 flips on (environment, goal) clusters, with Holm correction within each family. Table 1 marks use a two-sided paired t-test over seeds.

4.2 Experimental Results

Main comparison. Table 1 compares SAD-CEM with fifteen planners that all act through search over the same frozen LeWM checkpoints. SAD-CEM has the highest mean success on every environment, tying TRM at 0.976 on TwoRoom and reaching 0.988 on Reacher, 0.696 on PushT and 0.628 on Scene. On Cube it reaches 0.680, ahead of LeWM++ at 0.672, the strongest baseline on Cube and PushT. Averaged over the five environments, SAD-CEM reaches 0.794 against 0.630 for LeWM++, the runner-up. With four CEM iterations, SAD-CEM keeps a four-environment mean of 0.840 at 1.7 times the batch time of LeWM++, whose own setting gives 0.665 ($p = 5 \times 10^{-5}$,

Table 2: Label control on the final split (success rate, mean ± SD over five seeds). The other heads keep the pairs, draws, initializations and optimizer of the success-aligned heads and change only the label: a step gap $(j-i)/T$, or reachability $\mathbf{1}[\tau^* > h]$ with $h = 10$ and the same task predicate. The p values in Section 4.2 are one-sided paired tests pooled over the four environments.

Planner	Heads	Reacher	TwoRoom	Cube	PushT	Mean
SAD-CEM	success-aligned	$\mathbf{0.988 \pm 0.011}$	0.976 ± 0.017	$\mathbf{0.680 \pm 0.058}$	$\mathbf{0.696 \pm 0.050}$	**0.835**
SAD-CEM	step gap	0.956 ± 0.026	$\mathbf{0.988 \pm 0.011}$	0.628 ± 0.030	0.608 ± 0.098	0.795
SAD-CEM (direct mode)	success-aligned	0.980 ± 0.014	0.964 ± 0.033	0.608 ± 0.030	0.644 ± 0.055	0.799
SAD-CEM (direct mode)	reachability ($h = 10$)	0.904 ± 0.026	0.900 ± 0.037	0.532 ± 0.030	0.180 ± 0.040	0.629
SAD-CEM (direct mode)	step gap	0.940 ± 0.020	0.968 ± 0.023	0.492 ± 0.023	0.592 ± 0.023	0.748

Table 3: Cost and schedule factorial on the validation split with a single success-aligned head (success rate, mean over seeds 0–2). Receding 1 re-plans after every 5-step block, while receding 5 executes five blocks per decision as in the released LeWM schedule of the CEM row.

Cost	Schedule	Reacher	TwoRoom	Cube	PushT
Latent Euclidean	receding 1	0.333	0.453	0.500	0.247
Success-aligned	receding 5	0.947	0.980	**0.607**	0.367
Success-aligned	receding 1	**0.993**	**0.993**	0.593	**0.527**

Appendix L). With goals 75 and 100 steps ahead, SAD-CEM keeps a higher four-environment mean than LeWM++ and CEM, 0.798 and 0.762 against 0.685 for LeWM++ (Holm $p < 10^{-3}$), while CEM with the latent Euclidean cost falls to 0.494 and 0.365 (Figure 6b, Appendix M).

Success-aligned cost. SAD-CEM (direct mode) keeps only the success-aligned ensemble cost and its cross-entropy search, with the search size of the CEM row. It reaches a five-environment mean of 0.742 (Appendix E), above every external row of Table 1, including LeWM++ at 0.630 and the CEM row at 0.538, whose search size it shares. A validation factorial with one head separates the cost from the re-planning schedule of the CEM row (Table 3). With re-planning after every block, the success-aligned cost lifts every environment over the latent Euclidean cost, PushT from 0.247 to 0.527, and the schedule matters mainly on PushT. On validation pairs over 43 steps apart on PushT, the rank correlation with success-aligned time is 0.80 for the learned distance and 0.10 for the latent Euclidean cost (Appendix J). On the four environments of Table 2, a step-gap label lowers SAD-CEM (direct mode) from 0.799 to 0.748 ($p = 2 \times 10^{-4}$) and SAD-CEM from 0.835 to 0.795 ($p = 0.001$). A reachability label $\mathbf{1}[\tau^* > 10]$ built from the same task predicate lowers SAD-CEM (direct mode) further, to 0.629 ($p = 5 \times 10^{-5}$) with PushT at 0.180, so with the same predicate the time until success outperforms this fixed-horizon binary label. Replacing the ensemble by one head throughout lowers SAD-CEM to 0.807 (Holm $p = 0.032$, Figure 4), with PushT falling to 0.612 and Cube to 0.668. Keeping the five heads without the spread penalty ($\kappa = 0$) gives 0.826, so averaging the heads recovers most of the gap between one head and the pessimistic ensemble at 0.835 (Appendix I). The waypoint branch lifts the four-environment mean from 0.799 to 0.835 at the default search size ($p = 0.005$), and Appendix A reports its ablations and candidate sources.

Where the learned cost is optimized (Proposition 1). Proposition 1 bounds an optimism that grows with the error of the learned distance on the targets it ranks, and targets that no encoder produced differ most in form from its training pairs. Optimizing toward the chosen generated waypoint with one SAD head instead of the pessimistic ensemble (V1) lowers the four-environment mean from 0.838 to 0.812 (Holm $p = 0.039$), with Cube falling from 0.688 to 0.608 (Figure 5a). The ensemble thus absorbs head errors that the optimizer would exploit on generated targets, and Lemma 1.1 places this benefit in the errors on which heads disagree. Figure 5(b) lets the search optimize the ensemble over training-pool latents for a midpoint, which drops validation PushT to 0.296 against 0.696 for SAD-CEM (direct mode), consistent with Proposition 1.

Search budget (Proposition 1). Proposition 1 bounds an optimism that grows with the number of compared candidates, so a larger search can exploit more of the cost error. With four CEM iterations, SAD-CEM (direct mode) reaches a four-environment mean of 0.828 at 1.10 times the batch time of LeWM++, which reaches 0.665 (Figure 6c). More iterations lower it to 0.799 at 30, consistent

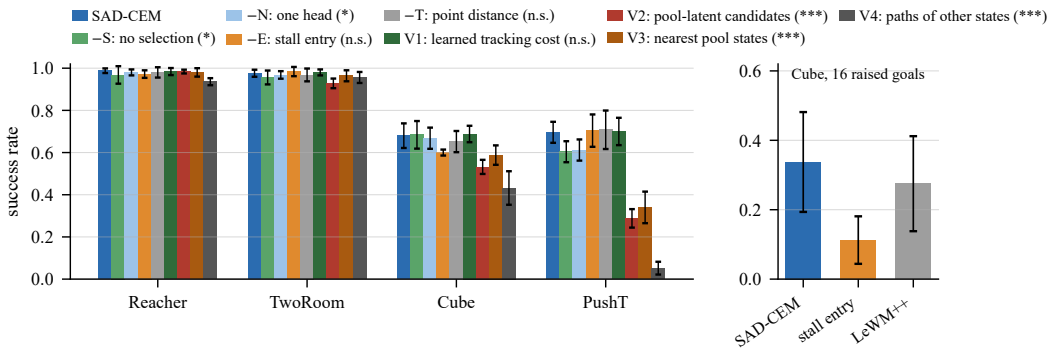


Figure 4: Component ablation, candidate-source arms V2–V4 and the tracking-cost arm V1 of Appendix A, final split, mean ± SD over five seeds. Markers give Holm-corrected one-sided paired tests of SAD-CEM against each variant ($^{*}p < 0.05$, $^{***}p < 10^{-3}$). Right: the 16 raised goals among the Cube final goals (Proposition 2), shown without a test for reference.

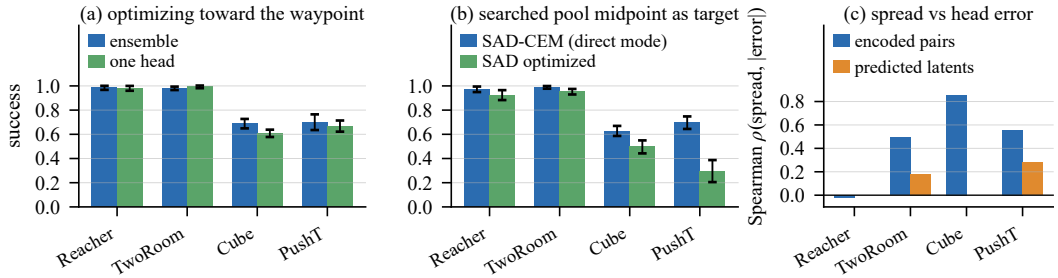


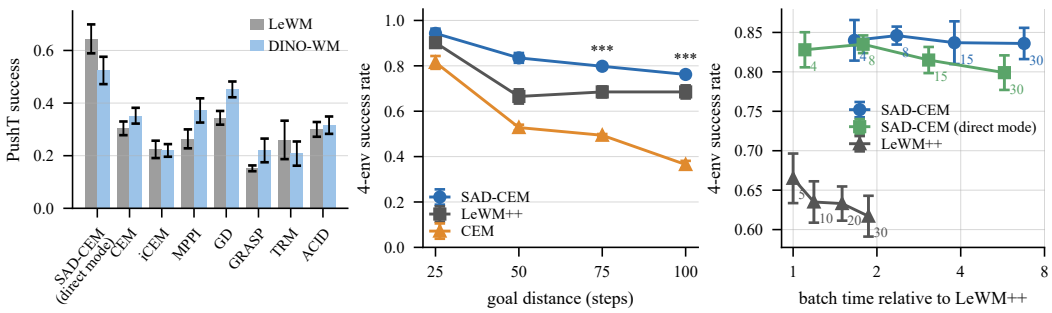
Figure 5: Mean ± SD over five seeds. (a) Final split: SAD-CEM optimizing toward the chosen generated waypoint with the pessimistic ensemble (V1) or with one SAD head (Lemma 1.1). (b, c) Validation split. (b) SAD-CEM (direct mode) or a search for an ensemble-optimal pool midpoint (Proposition 1). (c) Rank correlation of spread with absolute head error on 30,000 encoded validation pairs, and with predicted latents on TwoRoom and PushT only (Lemma 1.1).

with this growing optimism. SAD-CEM stays between 0.836 and 0.846 over the same budgets, and LeWM++ moves between 0.617 and 0.665 over its four budgets (Table L.1 in Appendix L).

What the ensemble spread detects (Lemma 1.1). On 30,000 encoded validation pairs, the rank correlation between spread and absolute head error is 0.56 on PushT and 0.86 on Cube (Figure 5c). When the first latent of a PushT pair is predicted instead, the correlation drops to 0.28, consistent with an error shared by all heads that Lemma 1.1 places outside the spread.

Remark 1 (Backbone requirements). SAD-CEM (direct mode) needs only an encoder and a predictor on which the heads are trained, while the support test and the waypoint solve also need path samples in the same latent space, as the LeWM++ generator provides for LeWM latents.

The cost on another backbone (Remark 1). Remark 1 lets SAD-CEM (direct mode) move to another backbone, so we test it on the official DINO-WM PushT checkpoint (Zhou et al., 2025). Every method keeps its LeWM-selected configuration on DINO-WM, with the learned parts reading the mean patch token of the frozen DINO-WM encoder. Figure 6(a) shows that SAD-CEM (direct mode) stays the best row on both backbones, at 0.644 on LeWM and 0.524 on DINO-WM. Both columns use 50 final-split PushT goals set 50 steps ahead with a 50-step budget, and the LeWM column repeats Table 1 for the baselines and Table 2 for SAD-CEM (direct mode).



(a) backbone swap on PushT (b) goal distance (c) search budget

Figure 6: The success-aligned cost beyond the main setting, mean ± SD over five seeds. (a) The same methods on the frozen LeWM and DINO-WM PushT checkpoints, final split (Remark 1). GRASP on DINO-WM optimizes the per-patch visual and proprioceptive channels, and the DINO-WM runs seed every reset per goal. (b) Four-environment mean success against goal distance, with a 50-step budget up to 50-step goals and a budget equal to the distance beyond (Appendix M). ***: Holm-corrected one-sided paired test of SAD-CEM against LeWM++, $p < 10^{-3}$. (c) Four-environment mean success against the batch time relative to LeWM++ at 5 iterations of the same job, with labels giving the CEM iterations of each point (Appendix L, Table L.1).

Table 4: Final-split hyperparameter checks (success rate, mean ± SD over five seeds).

Setting	Reacher	TwoRoom	Cube	PushT	Mean
SAD-CEM ($\kappa = 1$, 8 paths)	$\mathbf{0.988 \pm 0.011}$	0.976 ± 0.017	$\mathbf{0.680 \pm 0.058}$	0.696 ± 0.050	$\mathbf{0.835}$
$\kappa = 0$	0.976 ± 0.009	$\mathbf{0.984 \pm 0.017}$	0.664 ± 0.055	0.680 ± 0.095	0.826
64 paths	0.980 ± 0.000	0.960 ± 0.024	0.676 ± 0.036	$\mathbf{0.720 \pm 0.032}$	0.834
256 paths	$\mathbf{0.988 \pm 0.011}$	0.968 ± 0.011	$\mathbf{0.680 \pm 0.047}$	0.668 ± 0.069	0.826

Hyperparameter sensitivity (Proposition 1, Lemma 1.1). The path count n plays the role of N in Proposition 1, which it enters only through $\sqrt{2 \ln N}$, and Lemma 1.1 limits κ to the error on which heads disagree, so neither should move success much. Setting κ to 0.5 or 2 or n to 4 or 16 keeps the five-environment validation mean between 0.806 and 0.825 (Figure I.2 in Appendix I). On the final split, $\kappa = 0$ and 64 or 256 paths give 0.826, 0.834 and 0.826 against 0.835 (Table 4).

5 Conclusion

We argued that planning on a frozen latent world model depends first on what the optimized cost measures. Success-aligned distances regress the logged time until the task predicate first holds, a label that vanishes exactly on success. Optimized as a pessimistic ensemble toward the encoded goal, this cost alone lifts SAD-CEM (direct mode) above every external planner in the five-environment mean. A step-gap label or a single head in its place costs significant success, and a waypoint branch that consults generated paths adds success at the default search size. The label also beats a reachability label built from the same predicate, and the cost keeps its lead on DINO-WM and for goals 100 steps ahead. With frozen LeWM backbones SAD-CEM has the highest or tied-highest mean success on each of five goal-reaching environments under one shared protocol.

The support test and the waypoint candidates reuse the LeWM++ path generator, limiting the full planner to backbones that have such a generator. With the search budget of each variant chosen on validation, the waypoint branch adds 0.010 over SAD-CEM (direct mode) on five environments ($p = 0.25$, Appendix L). Deriving success-aligned labels when the task predicate is unavailable on the logged states of the pool is an open question. The reachability control fixes one horizon, $h = 10$, so whether other horizons narrow its gap to the success-aligned label remains to be tested. All five environments are simulated, so transfer of the learned heads to physical robots is the next question.

AI USE STATEMENT

In this work, we used generative AI tools for research execution, including writing and running experiment code, to aid and polish the writing, for retrieval and discovery of related work, to draft sections of the paper and to help prove the mathematical claims. We have reviewed all AI-assisted work. Every reported number was traced to the result files of the corresponding runs, the experiment code is released for inspection, every reference was checked against its source, and every proof in Appendix B was checked step by step by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix N give the environments, goal cohorts, training budgets and planner settings of the rows, and Appendix K and Appendix D give the protocol of Table 1. The proofs of Proposition 1 and the other results are in Appendix B, and Appendix A details the waypoint branch with Algorithm 1. Anonymous code with the training and evaluation scripts is available at `https://anonymous.4open.science/r/sad-cem`.

REFERENCES

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

Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 277–286. PMLR, 2020.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. arXiv:1805.12114.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. arXiv:2310.16828.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with MuJoCo. *arXiv preprint arXiv:2212.00541*, 2022.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017. arXiv:1612.01474.

Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein variational model predictive control. In *Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1278–1297. PMLR, 2021.

LeWM++. LeWorldModel++ (LeWM++). `https://github.com/pickxiguapi/LeWorldModel-pp`, 2026. Code repository, commit 7cec3dff.

Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics. *arXiv preprint arXiv:2605.22164*, 2026a.

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*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989. doi: 10.1007/BF01589116.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, volume 29, pp. 2378–2386. Curran Associates, Inc., 2016.

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.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation*, pp. 7559–7566. IEEE, 2018. doi: 10.1109/ICRA.2018.8463189.

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*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065, 2021. arXiv:2008.06389.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models. *arXiv preprint arXiv:2602.00475*, 2026.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Gawon Seo, Dongwon Kim, and Suha Kwak. ACID: Action consistency via inverse dynamics for planning with world models. *arXiv preprint arXiv:2607.02403*, 2026.

James E. Smith and Robert L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026. arXiv:2512.24497.

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

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation*, pp. 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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. *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. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. arXiv:2411.04983.