AIM: ACTION-SUPERVISED INTENT AS A SEARCH PRIOR FOR WORLD MODEL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models let an agent plan goal-directed behavior from visual observations by evaluating candidate action sequences inside learned dynamics. Under a finite compute budget, however, effective sampling-based planning depends not only on how accurately candidates are evaluated but also on whether the search starts in the region of action space relevant to the current task. Planners such as CEM and MPPI typically start from a generic proposal distribution, so even when offline trajectories already record the actions that connect the current observation to a future goal, the planner must rediscover the coarse structure of goal-directed behavior. A substantial share of the sampling budget can therefore be spent locating a useful action region before the world model begins to refine candidate plans. To address this, we propose AIM (Action-Supervised Intent Module), which uses offline goal-conditioned action supervision to build an informed search prior for a pretrained, frozen world model. Given frozen current and goal latents, AIM predicts a compact temporal intent for the connecting action block—the leading coefficients of its orthonormal discrete cosine transform (DCT)—and, through the inverse transform, uses it only to initialize the planner's action-sequence mean, leaving the representation, dynamics, rollout, sampling rule, and planning objective unchanged. This frequency-domain parameterization gives a compact, coarse description of goal-directed behavior and splits the initialization error exactly into a prediction term and a truncation term, which ranks intent dimensions before any planning run. Across three frozen world-model hosts, AIM consistently improves finite-budget planning, raising the five-task average from 47.5% to 92.9% with a single 300-candidate CEM iteration in the largest-gain setting, while retaining its advantage across larger search budgets and under MPPI.

1 INTRODUCTION

Latent world models let an agent plan goal-directed behavior from visual observations by predicting the consequences of candidate actions in a learned representation space (Hafner et al., 2019; Zhou et al., 2025; Maes et al., 2026b). Much recent work makes these predictions more accurate or cheaper (Maes et al., 2026b; Ivashkov et al., 2026; Gao & Xu, 2026), but under a finite compute budget a sampling-based planner must also decide where in action space to begin searching. Planners such as CEM and MPPI start from a generic proposal distribution and improve it by sampling candidates, rolling them out through the model, and refitting toward the best (Rubinstein, 1999; Pinneri et al., 2021; Williams et al., 2017). Because the world model can only evaluate the sequences that are proposed, and each refit moves the proposal only locally, a substantial share of the budget goes to locating the task-relevant action region before any plan is refined (Figure 1; Appendix E.1).

Drawing more samples or running more iterations is an intuitive but incomplete remedy. More search raises the rollout cost of every replanning step, and optimizing a learned objective more thoroughly does not necessarily improve closed-loop control: past a task-dependent budget, further search lowers success on some tasks even from the default proposal, while the predicted cost of the selected plan keeps falling (Section 6.2). The more fundamental issue is that the planner rediscovers information already present in the offline data: its trajectories record the actions connecting each observation to later ones, yet the world model uses these actions only as inputs to its dynamics. Existing methods spend this supervision on the representation, for example through inverse-dynamics objectives (Ivashkov et al., 2026), which changes how candidates are scored but not where the search

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

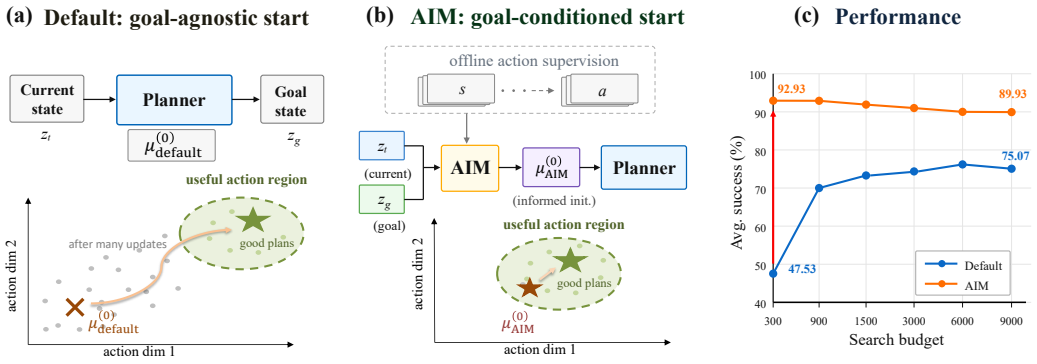


Figure 1: **Where the planner begins searching.** (a) Default CEM starts from the goal-agnostic mean $\mu^{(0)}_{\text{default}}$ and spends iterations reaching the useful action region. (b) AIM uses offline action supervision to map the current and goal latents (z_t, z_g) to an informed initial mean $\mu^{(0)}_{\text{AIM}}$ for the same planner. (c) Five-task average success on our main frozen host (Ours; Section 5) against the CEM search budget, in sampled trajectories with 300 candidates per iteration, so $B = 300$ is one iteration and $B = 9{,}000$ is thirty.

starts, or expose it to the planner through learned policies and action priors that add a policy module or change how the planner samples and searches (Wang & Ba, 2020; Cheng et al., 2026; Sun et al., 2026; Wang et al., 2026). We therefore ask a stricter question: *can offline goal-conditioned actions give a sampling-based planner an informed starting point, while leaving the pretrained world model and the planner's search unchanged?*

To this end, we propose AIM (Action-Supervised Intent Module), a lightweight goal-conditioned module that turns offline connecting actions into a search prior for a pretrained, frozen world model (Figure 2). From the frozen current and goal latents, AIM predicts a compact temporal summary of the connecting action block and uses it only as the planner's initial mean. AIM thus separates proposing candidates from evaluating them: it tells the planner where to begin, the frozen world model evaluates every candidate, and the planner, keeping its sampling variance, rollout, objective, and update rule, stays free to leave the proposal. Unlike PRISM, which also sets MPPI's sampling width (Wang et al., 2026), AIM leaves how far the search moves to the planner, so the same module serves CEM and MPPI unchanged at about 1% of a host's parameters, and removing it recovers the original planner.

A key design choice is to predict a coarse temporal intent rather than a complete executable action sequence. A search prior need not reproduce the demonstrated trajectory; it only has to place the search in a region that already contains the coarse structure of goal-directed behavior, which the planner then refines through the learned dynamics. The fine detail is also what the data cannot supply: it varies across the many sequences that reach the same goal, so a model trained on whole sequences returns at best their average, whose detail belongs to none of them. We therefore parameterize the intent by the leading coefficients of the block's orthonormal discrete cosine transform along time, which capture its mean action and coarsest temporal variation and leave finer variation to the search. The DCT orders a block from slow to fast variation and, for correlated signals, concentrates energy in the leading coefficients nearly as well as the optimal Karhunen–Loève transform without a basis to estimate (Ahmed et al., 1974), the property FAST uses to tokenize actions (Pertsch et al., 2025).

This parameterization also lets us analyze how much temporal structure the prior should keep. Because the DCT is orthonormal, the squared error between the recorded action block and the initialization reconstructed from the first K predicted coefficients splits exactly into a prediction term for the kept coefficients and a truncation term for the dropped ones (Section 4.3). Starting from the default zero mean, which bears the full error, each added temporal mode removes its share of the offline action energy times the validation R^2 with which the frozen current and goal latents predict it. Both factors are measured without planning, so candidate intents can be ranked in advance, and this ranking picks out the temporal modes that improve planning (Section 6.4).

Our contributions are:

- AIM, a goal-conditioned module that turns offline connecting actions into a search prior for a frozen world model and changes only the planner's initial mean.
- A frequency-domain parameterization of the prior whose initialization error splits exactly into a prediction and a truncation term, ranking candidate intents before any planning run.
- Evidence across three frozen hosts, a thirtyfold budget range, and two planners that a better start is worth more than more search: on every host's five-task average, one AIM iteration beats thirty default iterations, and most of the gain is already in the prior.

2 Related Work

Latent world models and sampling-based planning. PETS plans through probabilistic dynamics ensembles (Chua et al., 2018), PlaNet in a latent state learned from pixels (Hafner et al., 2019), and GLAM models global and local state variation during imagination (He et al., 2025). Joint-embedding models instead predict in a learned representation space (Assran et al., 2023). DINO-WM plans over frozen DINOv2 patch features (Zhou et al., 2025; Oquab et al., 2024). LeWorldModel trains such a model stably end to end from pixels (Maes et al., 2026b), Fast-LeWM predicts the latents of all action prefixes in parallel (Gao & Xu, 2026), and Terver et al. (2026) study what makes planning with these models succeed. CEM, iCEM, and MPPI update their sampling distributions in different ways but start from a generic proposal (Rubinstein, 1999; Pinneri et al., 2021; Williams et al., 2017).

Action supervision for representation learning. Inverse dynamics can bias a representation toward controllable information and provide an anti-collapse signal; SMWM applies this by predicting actions from consecutive latent observations (Ivashkov et al., 2026). Related state-grounded objectives calibrate latent structure: DeepMDP preserves reward and transition information (Gelada et al., 2019), bisimulation connects latent distance to behavioral equivalence (Zhang et al., 2021), and SCALE calibrates JEPA geometry to task-relevant state distance (Hu et al., 2026).

Learned search priors and proposal guidance. POPLIN initializes action-sequence optimization from policy networks (Wang & Ba, 2020); SAGE generates actions conditioned on predicted latent subgoals (Cheng et al., 2026); INTACT couples an isomorphic intent-to-action map with plan-centered CEM, and can execute its conditional mean without search (Sun et al., 2026); and PRISM fuses a goal-conditioned Gaussian action prior learned on frozen JEPA features into MPPI's sampling distribution (Wang et al., 2026). Diffusion Policy models action sequences for direct execution (Chi et al., 2023). AIM changes the least of these: it writes only the planner's initial mean and leaves the proposal variance, sampling rule, and terminal objective as they are, so it attaches to an existing planner without modifying it; Section 4.3 examines what such a minimal interface can and cannot buy. Appendix G details the overlap with the closest of these methods and what our setting adds.

3 Preliminaries

We consider goal-conditioned planning with a latent world model learned from offline trajectories of image observations x_t and actions $a_t \in \mathbb{R}^{d_a}$. The model encodes each observation and predicts the next latent given an action:

$$z_t = f_\theta(x_t) \in \mathbb{R}^D, \qquad \hat{z}_{t+1} = F_\omega(z_t, a_t). \tag{1}$$

Given a goal image x_g with latent $z_g = f_\theta(x_g)$, planning seeks an action sequence $A = (a_t, \ldots, a_{t+H-1})$ over H model steps whose rollout under F_ω from $\hat{z}_t = z_t$ minimizes the terminal cost $J(A) = \|\hat{z}_{t+H} - z_g\|_2^2$. CEM (Rubinstein, 1999) searches over standardized actions with a diagonal Gaussian proposal of mean $\mu \in \mathbb{R}^{H \times d_a}$ and standard deviation σ, by default initialized at $\mu^{(0)} = 0$ and $\sigma^{(0)} = 1$. Each iteration samples N candidates, rolls each out through F_ω, and refits μ and σ to the N_e lowest-cost candidates, the elites, so I iterations spend a budget of $B = NI$ sampled trajectories. The final mean is executed for a fixed number of steps, after which the planner replans from the new observation.

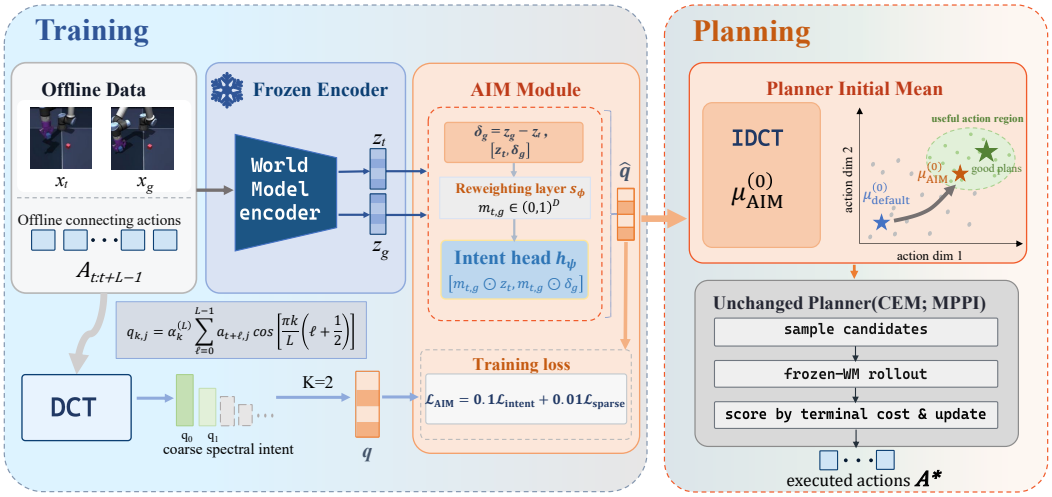


Figure 2: **AIM changes only where the planner begins searching.** Offline current–goal pairs and their connecting actions train the intent predictor on frozen latents. The predicted leading DCT coefficients $\hat{q}$ are inverse-transformed into the initial mean $\mu^{(0)}_{\text{AIM}}$. The world model stays frozen, and the planner—CEM in our main experiments—keeps its sampling variance, candidate budget, full-latent rollout, terminal objective, and update rule.

AIM attaches to a pretrained world model of this form, which we call the *host*. The host remains frozen throughout: AIM updates none of its parameters, and planning always rolls out its full latent. Section 5 describes the hosts used in our experiments.

4 Method

4.1 AIM intent module

Current–goal action supervision. For a goal $x_g = x_{t+L}$ sampled $L \in \{1, \dots, 5\}$ model steps ahead, the offline trajectory supplies the connecting action block $A_{t:t+L-1} = (a_t, \dots, a_{t+L-1})$. We summarize this block with an orthonormal type-II DCT (Ahmed et al., 1974) along time, applied to each action coordinate j:

$$q_{k,j} = \alpha_k^{(L)} \sum_{\ell=0}^{L-1} a_{t+\ell,j} \cos\left[\frac{\pi k}{L}\left(\ell + \frac{1}{2}\right)\right], \quad \alpha_k^{(L)} = \begin{cases} L^{-1/2}, & k = 0, \\ (2/L)^{1/2}, & k > 0. \end{cases} \tag{2}$$

By default AIM keeps the first $K = 2$ coefficients, which capture the mean action and one coarse temporal mode, as a fixed $K \times d_a$ target. When $L < K$, only $K_{\text{eff}} = \min(K, L)$ coefficients are defined, and the remaining target entries are excluded from the loss.

Intent prediction. Given the current latent z_t and the goal displacement $\delta_g = z_g - z_t$, AIM computes

$$m_{t,g} = s_\phi([z_t, \delta_g]) \in (0,1)^D, \qquad \hat{q}_{t,g} = h_\psi([m_{t,g} \odot z_t,\ m_{t,g} \odot \delta_g]), \tag{3}$$

where s_ϕ is an input-reweighting layer whose output $m_{t,g}$ rescales the frozen features of the current–goal pair, and h_ψ is the intent head that reads them. Because a frozen latent encodes the whole scene, much of it unrelated to the action that reaches the goal, this auxiliary reweighting lets the head emphasize the directions that matter for the current–goal pair.

The intent loss is the mean-squared error over the set $\mathcal{V}$ of valid entries (b, k, j) in a batch,

$$\mathcal{L}_{\text{intent}} = \frac{1}{|\mathcal{V}|} \sum_{(b,k,j) \in \mathcal{V}} (\hat{q}_{b,k,j} - q_{b,k,j})^2, \tag{4}$$

and the training objective combines it with a sparsity penalty on the reweighting:

$$\mathcal{L}_{\text{AIM}} = 0.1\mathcal{L}_{\text{intent}} + 0.01\mathcal{L}_{\text{sparse}}, \qquad \mathcal{L}_{\text{sparse}} = D^{-1}\sum_i m_{t,g,i}. \tag{5}$$

Only s_ϕ and h_ψ are trained. The sparsity penalty encourages the reweighting to concentrate on fewer directions (Section 6.4).

4.2 Search prior for the planner

At each MPC step in the environment, AIM predicts $\hat{q}_{t,g}$, zero-pads it to the planning horizon H, and inverts the orthonormal DCT of Equation 2 at length H:

$$\mu^{(0)}_{\text{AIM},\ell,j} = \sum_{k=0}^{\min(K,H)-1} \alpha_k^{(H)} \hat{q}_{t,g,k,j} \cos\left[\frac{\pi k}{H}\left(\ell + \frac{1}{2}\right)\right], \quad \ell = 0, \ldots, H-1. \tag{6}$$

This sequence replaces CEM's default initial mean $\mu^{(0)} = 0$ and is not executed directly. Action bounds, $\sigma^{(0)}$, the sampling rule, full-latent rollout, the terminal cost J, the elite refit, and receding-horizon execution are unchanged, so AIM alters only $\mu^{(0)}$; Appendix F.2 applies the same prior to MPPI. With $H = 5$, $N = 300$, and $N_{\text{e}} = 30$, the budget $B = 300$ used in our main results is a single iteration, whose elite mean is executed. Appendix D.2 describes a variant that additionally uses m to weight the terminal comparison.

4.3 What the initial mean can and cannot buy

Two observations delimit what this interface can achieve; Appendix F derives the first and states the scope of both. First, because the transform in Equation 2 is orthonormal, Parseval's theorem splits the expected squared distance between the recorded action block and an initial mean built from K predicted coefficients (the rest set to zero) into a prediction and a truncation term:

$$E(K) = \underbrace{\textstyle\sum_{k<K} \mathbb{E}\big(\hat{q}_k - q_k\big)^2}_{\text{prediction}} + \underbrace{\textstyle\sum_{k\geq K} \mathbb{E}\, q_k^2}_{\text{truncation}} = \textstyle\sum_{k<K} \big(1 - R_k^2\big)e_k + \sum_{k\geq K} e_k, \tag{7}$$

where e_k is the fraction of the offline blocks' energy in coefficient k, which equals $\mathbb{E}\, q_k^2$ for centered, standardized actions, and R_k^2 is the validation R^2 of coefficient k. The first equality is exact; the second follows from the definition of R_k^2, with e_k as the coefficient variance. The default $\mu^{(0)} = 0$ has $E(0) = 1$, and $E(K+1) = E(K) - R_K^2 e_K$: each added mode is worth its energy times its predictability. The fractional reduction $R_K^2 e_K / E(K)$ therefore ranks candidate modes from offline data and validation fits alone, although the threshold τ that decides which modes to keep must be set from planning results. Because $E(K)$ measures distance to the recorded block rather than to a good plan, we use it to rank parameterizations and tasks, not to predict success rates.

Second, a better initial mean helps only while the search still depends on where it started. A finite-budget refiner evaluates only the candidates it draws, so its result can depend on its initialization. Once searches from two initial means converge to the same point of the learned objective, however, the initialization no longer matters, and closed-loop success depends only on how well that point agrees with the task. How long this dependence lasts, and whether additional search helps at all, are empirical questions that Section 6.2 addresses.

5 Experimental Protocol

All experiments attach AIM to a frozen, pretrained host and measure closed-loop success on image-goal control tasks. Main results use a single CEM iteration of $B = 300$ candidates, and the budget analysis varies only the iteration count, up to thirty ($B = 9{,}000$).

Tasks. We use the four image-goal control tasks on which LeWM (Maes et al., 2026b) and later work report planning results, from the stable-worldmodel ecosystem (Maes et al., 2026a): planar object pushing (PushT), point navigation through two rooms (TwoRoom), joint-configuration matching

(Reacher), and single-object manipulation adapted from OGBench (Park et al., 2025) (Cube). To these we add a harder two-object variant of Cube (Cube-Double), which evaluates the object selected in the start state against its actual pose in the goal frame. Observations are 224×224 pixels, and with frame skip five the 25-step evaluation goal offset matches the five-step model-space horizon.

World models. Our main host, **Ours**, is a LeWM-style world model (Maes et al., 2026b): a ViT-tiny encoder ($D = 192$) and a transformer predictor, trained end to end on each task's offline data with latent prediction plus Sketched Isotropic Gaussian Regularization (SIGReg; Balestriero & LeCun, 2025) at weight 0.09. **SMWM** keeps this architecture and replaces SIGReg with a one-step inverse-dynamics regularizer, following Ivashkov et al. (2026); the two share task datasets, observation history, goal construction, evaluator, and CEM configuration, and differ only in pretraining objective. The **DINO-WM-style** host (Zhou et al., 2025) swaps this encoder for a frozen, non-JEPA DINOv2 encoder (Oquab et al., 2024), an encoder ablation that tests whether AIM is encoder-agnostic (Appendix B). For each task only the intent predictor is trained, for three epochs, after the backbone is frozen.

Comparisons and evaluation. The primary comparison contrasts Default CEM with AIM-CEM, which differ only in the initial mean. Every condition is run on three independently trained seeds (0, 1, 3072), each evaluated on 100 rollouts with a 50-step interaction budget; we report the mean and sample SD across seeds.

Ablations. On frozen Ours backbones, with all other settings fixed, we remove search by executing the intent head's proposal open loop, compare DCT targets with $K \in \{1, 2, 3, 5\}$ against Direct-H5 prediction of the whole block, and replace the reweighting layer with $m \equiv \mathbf{1}$.

Table 1: **Planning success and search budget.** Top: our frozen hosts (†) at $B = 300$, a single CEM iteration, where AIM-CEM (bold) differs from Default CEM only in its initial mean; mean ± sample SD. Ours is a LeWM-style world model trained with SIGReg. SMWM shares its architecture but replaces SIGReg with an inverse-dynamics regularizer. DINO-WM-style replaces the encoder with a frozen DINOv2. Bottom: published results. $(N \times I)$ denotes N candidates in each of I iterations, so the budget is $B = NI$.

Method	Planner	Budget	PushT	TwoRoom	Reacher	Cube	Cube-Double	Avg.
Our frozen-backbone evaluation at the reported operating point, $B = 300$ in one CEM iteration								
Ours†	Default CEM (300×1)	300	14.00±1.00	58.33±2.52	82.33±2.31	41.33±0.58	41.67±2.52	47.53
	AIM-CEM (300×1)	300	**84.67±2.08**	**99.00±0.00**	**86.67±3.21**	**99.00±1.00**	**95.33±0.58**	**92.93**
SMWM†	Default CEM (300×1)	300	33.67±0.58	85.00±7.81	84.33±2.08	44.33±2.08	39.67±0.58	57.40
	AIM-CEM (300×1)	300	**82.33±0.58**	**96.33±2.31**	**88.33±2.08**	**100.00±0.00**	**99.00±0.00**	**93.20**
DINO-WM-style†	Default CEM (300×1)	300	1.67±1.15	59.00±1.00	20.67±2.08	42.33±1.15	35.33±0.58	31.80
	AIM-CEM (300×1)	300	**1.67±0.58**	**98.67±0.58**	**27.33±5.51**	**62.33±0.58**	**51.67±1.53**	**48.33**
Published results on these tasks								
LeWM (Maes et al., 2026b)	CEM (300×30/10)	9k/3k	96.0±2.83	87.0±2.0	86.0±5.0	74.0±3.0	–	–
Fast-LeWM (Gao & Xu, 2026)	CEM (300×30/10)	9k/3k	96	98	88	80	–	–
DINO-WM (Zhou et al., 2025)	CEM (300×30/10)	9k/3k	74±4.0	100±0	79.0±5.0	86±5.0	–	–
INTACT (Sun et al., 2026)	CEM (128×3)	384	92.22±0.69	98.00±1.15	97.44±0.77	99.78±0.19	–	–
PRISM (Wang et al., 2026)	MPPI (128×30)	3,840	89±4	–	–	79±6	–	–

6 RESULTS

6.1 AIM IMPROVES FROZEN WORLD-MODEL PLANNING IN ONE 300-CANDIDATE ITERATION

At $B = 300$ CEM runs a single iteration, and with it AIM raises mean success in 14 of the 15 task–host cells, ties one, and lowers none (Table 1). The five-task average rises from 47.53% to 92.93% on Ours, from 57.40% to 93.20% on SMWM, and from 31.80% to 48.33% on the DINO-WM-style host, whose single tie is PushT, which it barely solves under either initializer. On every host this one AIM iteration also beats thirty default iterations, and on Ours it does so in 5–12× less evaluation time (Appendix E.3): at $B = 9{,}000$ default CEM reaches 75.07%, 81.80%, and 45.67% (Table 12, Appendix E.2). The smallest trained-host gains, $+4.33$ and $+4.00$ pp, are on Reacher, whose temporally white action blocks are barely predictable from the frozen latent (Section 6.4; Appendix F.1).

AIM closes the gap between the two pretraining objectives. Under default CEM, SMWM's inverse-dynamics pretraining pays off: its five-task mean exceeds Ours by $9.87\,\mathrm{pp}$ at $B = 300$ and by $6.73\,\mathrm{pp}$ at 9,000 (Table 12, Appendix E.2). With AIM attached the gap closes, to 93.20% against 92.93% at $B = 300$ and 89.53% against 89.93% at 9,000. Because the two hosts share architecture and data and differ only in pretraining objective, this comparison isolates the objective's effect: in the five-task mean, spending action supervision on the planner's initial mean alone matches spending it on both the representation and the initial mean.

6.2 Search is not free

AIM's advantage largely persists as the budget grows, though it narrows. From 300 to 9,000 candidates on the frozen Ours host (Table 2), AIM has a higher mean in 26 of 30 task–budget comparisons, ties once, and is lower three times by at most $1.67\,\mathrm{pp}$, all four exceptions on Reacher. The advantage does not rely on CEM narrowing its sampling width around the initial mean: under MPPI, whose softmax update never narrows it, AIM is ahead in all ten task–budget cells (Table 3). On the CEM grid, AIM reaches each task's 9,000-candidate Default mean with $10\times$ fewer candidates on PushT (900) and $30\times$ fewer on the other four (300, a single iteration)—on Reacher because Default at 300 already exceeds it. Every AIM row peaks at or below 3,000 candidates.

What the extra budget does once the proposal is informed matters more. Raising B from 300 to 9,000 *lowers* mean success in 9 of the 15 task–host cells under AIM initialization, against 2 of 15 under the default proposal (Table 12, Appendix E.2). Reacher declines in all four trained-host cells, each starting at 82–88%, well below saturation; DINO-WM-style Cube, also far from saturation, loses $5.00\,\mathrm{pp}$. Letting the proposal also set the sampling width, through PRISM's precision-weighted rule (Wang et al., 2026), raises success on Ours but does not remove this decline (Appendix G), and neither does switching to MPPI, under which more budget lowers success on all five tasks under AIM initialization and on four of five even under the default proposal (Table 3).

Table 2: **Closed-loop success across search budgets on the frozen Ours host.** Columns are sampled trajectories per solve. **Bold** marks each row's best budget; the last column gives the drop from that best to 9,000 candidates where it exceeds $1\,\mathrm{pp}$. Every cell is an independent closed-loop evaluation, not an intermediate outcome of one long search, and all solver settings other than the iteration count are fixed.

Task	Initializer	Candidate trajectories per solve						Drop from
		300	900	1,500	3,000	6,000	9,000	best
PushT	Default CEM	14.00	64.00	77.00	85.33	**88.00**	87.33	–
	AIM-CEM	84.67	88.67	92.67	**94.33**	94.33	92.33	−2.00
TwoRoom	Default CEM	58.33	84.67	90.00	90.67	**93.33**	89.67	−3.67
	AIM-CEM	99.00	98.67	**100.00**	99.67	98.67	100.00	–
Reacher	Default CEM	**82.33**	81.33	75.67	70.33	71.67	69.33	−13.00
	AIM-CEM	**86.67**	80.33	75.67	69.33	70.00	72.00	−14.67
Cube	Default CEM	41.33	56.67	61.67	62.33	**62.67**	60.67	−2.00
	AIM-CEM	99.00	**99.67**	99.00	98.33	96.00	95.67	−4.00
Cube-Double	Default CEM	41.67	65.67	**72.33**	71.67	70.00	68.33	−4.00
	AIM-CEM	95.33	**97.33**	94.67	95.00	91.67	89.67	−7.67

Optimization is not failing while this happens. Over the same 30 iterations of the first real MPC step the predicted terminal cost of the selected sequence falls by factors of 4.8 to 42.4 on every task and initializer (Figure 5, Appendix E.2)—on Reacher under default initialization from 52.23 to 1.23—while closed-loop success there drops by $13.00\,\mathrm{pp}$ (Table 2). The planner is minimizing the learned objective successfully; past a task-dependent budget that is no longer the same as controlling the environment better.

The same traces show where initialization stops mattering (Table 14, Appendix F). At the start of the search, AIM's initial mean already has a lower predicted cost than the default mean on 78–98% of paired starts per task (Table 11, Appendix E.1). AIM therefore needs less budget than the default to reach the stage where further search optimizes the model rather than the task: its success peaks at a

Table 3: **The same module under MPPI.** MPPI replaces CEM's elite refit with a softmax-weighted update and keeps its sampling width fixed; all other settings are unchanged.

	B=300			B=9,000		
Task	Default	AIM	Δ pp	Default	AIM	Δ pp
PushT	52.67	84.00	+31.33	64.00	75.33	+11.33
TwoRoom	90.33	97.00	+6.67	70.67	72.00	+1.33
Reacher	62.67	67.33	+4.67	42.00	47.00	+5.00
Cube	47.00	91.00	+44.00	38.00	64.67	+26.67
Cube-Double	42.67	88.00	+45.33	41.67	68.33	+26.67

smaller budget than the default's on every task except Reacher, where both peak at 300 (Table 2). This is why more budget lowers success more often under AIM. After 30 iterations, Reacher's two initializations reach the *same* predicted plan cost, and there both lose 13–15 pp of success: a better point of the learned objective, a worse point of the task. We report $B = 300$ as one low-budget operating point; the budget at which AIM's success peaks varies by task, and on PushT AIM still gains from 84.67 at 300 to 94.33 at 3,000 before declining.

6.3 Ablation: executing the proposal without search

Most of AIM-CEM's performance is already in its proposal. The intent head can run as a controller on its own: predict $\hat{q}_{t,g}$ from the same frozen latents, inverse-transform it, and execute the block open loop—rollout-free and search-free, though not world-model-free, since it reads z_t, z_g from the frozen encoder. As in AIM-CEM, the head is called twice per 50-step episode, on the same episodes.

Under this matched protocol the prior alone reaches 89.40%, against 93.07% for AIM-CEM at $B = 300$ in the same cohort (Table 4), so rollout and search are worth 3.67 pp—concentrated rather than uniform, and negative on Reacher, where the learned objective agrees least with the task. They are worth keeping: on PushT they raise success from 73.33% to 85.33%. But here they correct a good proposal rather than finding one.

Table 4: **What the learned dynamics and the search over them contribute.** Prior-only rows execute the block predicted by the intent head open loop, with no rollout or search; the last row is AIM-CEM minus the two-call prior. Mean ± sample SD.

Condition	PushT	TwoRoom	Reacher	Cube	Cube-Double	Avg.
Prior only, 1 call / 25 env. steps	61.33±2.08	90.33±2.08	49.67±3.51	94.00±1.00	79.67±1.53	75.00
Prior only, 2 calls / 50 env. steps	73.33±4.04	93.00±0.00	89.67±1.53	97.67±0.58	93.33±3.06	89.40
AIM-CEM (300×1), same cohort	85.33±2.08	99.00±1.00	85.67±1.53	99.67±0.58	95.67±2.31	93.07
What search adds (pp)	+12.00	+6.00	−4.00	+2.00	+2.33	+3.67

6.4 Ablation: temporal parameterization of the intent

On an independent frozen-backbone cohort, we compare DCT targets with $K \in \{1, 2, 3, 5\}$ against Direct-H5 (Appendix C). Adding one temporal mode to the DC-only target is worth +17 to +24 pp on PushT, Cube, and Cube-Double but little on TwoRoom and Reacher; beyond that the marginal gains are −1.00 to +3.00 pp and Direct-H5, the full five-step block, lands within 3 pp of $K = 2$ everywhere. One coarse mode already carries most of what the proposal can contribute.

The 15 increments of K in Table 5, three per task, can be ranked before any planning run by the error reduction that Equation 7 predicts. The three it ranks highest, $K = 1 \to 2$ on PushT, Cube, and Cube-Double, are the three that improved success, by +17 to +24 pp, while no other increment gains more than 3 pp; the threshold that separates the two groups is set on these same increments. TwoRoom and Reacher, the two tasks that gain little, differ from the other three mainly in predictability rather than energy: their first temporal mode holds 0.192 and 0.201 of the block energy, the lowest of the five but within 1.7× of the rest, yet the frozen latent predicts it at validation R^2 of only 0.008 and 0.003, against 0.45–0.68 elsewhere. Their near-white action spectra (Appendix F.1) suggest why:

that energy is step-to-step variation, not a temporal shape the frozen latent could predict. Beyond $K = 2$ the predicted reduction on these two tasks turns slightly negative, and the observed changes, -1.00 to $+3.00$ pp, have no consistent sign: a mode predicted no better than zero adds error rather than removing it (Appendix F).

Table 5: **Ranking temporal modes before planning.** $E(1)$ is the squared initialization error of a DC-only proposal, with the default $\mu^{(0)}=0$ at 1. For each increment of K, *reduction* is the fractional error reduction predicted by Equation 7 and Δ pp the observed change in success at $B=300$; **bold** marks increments that raised success by more than 10 pp.

		$K=1 \to 2$		$K=2 \to 3$		$K=3 \to 5$	
Task	$E(1)$	reduction	Δ pp	reduction	Δ pp	reduction	Δ pp
PushT	0.7078	**0.2553**	+22.33	0.1690	+0.67	0.0859	+1.67
Cube	0.7570	**0.1983**	+23.67	0.0700	+0.00	0.0326	+0.00
Cube-Double	0.7469	**0.1927**	+17.00	0.0474	+0.67	0.0112	+0.33
TwoRoom	0.9335	0.0014	+3.00	−0.0004	+0.00	−0.0012	−0.33
Reacher	0.9703	0.0006	−1.67	−0.0002	−1.00	−0.0012	+3.00

The reweighting layer. The learned weighting fits the intent slightly better than $m \equiv \mathbf{1}$ on all five tasks, while the five-task (300×1) mean on Ours stays at 92.93% under both, with learned-minus-uniform differences of -2.33 to $+2.67$ pp per task (Table 8, Appendix D). Raising the sparsity weight from 0 to 0.03 concentrates the weighting from an effective 86–95% to 34–45% of the latent dimensions on PushT and Cube with almost no change in fit or success, and success falls only at 0.1 and above (Appendix D.1), consistent with the intent depending on a subset of latent directions.

7 Limitations and Discussion

Our evaluation covers five simulated image-goal tasks with short planning horizons and a single implementation of each planner; long-horizon, partially observed, and real-robot settings remain untested. Like other learned action priors (Sun et al., 2026; Wang et al., 2026), AIM needs offline trajectories whose actions connect observations to later goals; its predictor is trained per host and task, and it cannot compensate for a weak backbone. What transfers unchanged is the interface, which improves all five tasks at both budgets under a second sampling-based planner (Table 3), though the size of the gain depends on the planner. Beyond Table 1 and the three-host budget comparison in Appendix E.2, every analysis runs on Ours, and the DINO-WM-style rows come from a separate implementation, so the evidence supports within-host contrasts rather than cross-host ranking.

Two findings bear on world-model planning beyond AIM. The learned terminal objective can disagree with the task, so more search can lower success even from the default proposal (Section 6.2); a result reported at one budget therefore mixes the quality of a world model with how far its objective is searched, and reporting the budget response would separate the two. In our matched pair, action supervision spent on the planner alone matched spending it on both, so the two uses proved redundant; whether this holds for other pretraining objectives is open.

8 Conclusion

AIM turns offline current-to-goal actions into a search prior for a frozen world model, changing only the planner's initial mean. With a single CEM iteration it raises or matches success on every task and host, and lifts each host's five-task average beyond what thirty default iterations reach; with it attached, the two pretraining objectives plan alike. Most of that gain lies in the proposal: run open loop, the intent head comes within a few points of AIM-CEM. The advantage largely persists at thirty times the budget and under MPPI, yet search is not free: past a task-dependent budget, more of it lowers closed-loop success while the predicted cost keeps falling, as the planner optimizes the model rather than the task. The proposal's exact error split ranks temporal parameterizations before planning. Offline actions can tell a planner where to begin; how far to search from there depends on the task.

Reproducibility Statement

Appendix A describes the architectures, training, planner settings, and evaluation protocol, and the remaining appendices report additional results and analyses. Code is available at `https://anonymous.4open.science/r/AIM-code-CB5B/`.

Ethics Statement

This work uses simulated control and offline trajectories, without human participants, personal data, or physical deployment. Action guidance may encode demonstrator bias, and simulation success does not establish real-world safety or reliability. We report negative and boundary evidence and limit claims to the evaluated protocol.

AI Use Disclosure

OpenAI Codex was used to assist with result verification and code editing, while Claude was used for literature retrieval, citation checking, and language polishing. These tools were used only in supporting roles. The authors independently formulated the research, designed and conducted the experiments, verified all sources and results, and take full responsibility for the final content.

References

N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. doi: 10.48550/arXiv.2511.08544.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning. *arXiv preprint arXiv:2607.17973*, 2026. doi: 10.48550/arXiv.2607.17973.

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

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. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf`.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026. doi: 10.48550/arXiv.2606.26217.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179, 2019. URL `https://proceedings.mlr.press/v97/gelada19a.html`.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019. URL `https://proceedings.mlr.press/v97/hafner19a.html`.

Qian He, Wenqi Liang, Chunhui Hao, Gan Sun, and Jiandong Tian. GLAM: Global-local variation awareness in mamba-based world model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17105–17113, 2025. doi: 10.1609/aaai.v39i16.33880.

Jiaming Hu, Yan Zheng, and Tian Wang. SCALE: State-calibrated latent embeddings for JEPA planning in the right geometry. *arXiv preprint arXiv:2608.16287*, 2026. doi: 10.48550/arXiv.2608.16287.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026. doi: 10.48550/arXiv.2606.20104.

Lucas Maes, Quentin Le Lidec, Dan Haramati, Nassim Massaudi, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel-v1: Reproducible world modeling research and evaluation. *arXiv preprint arXiv:2602.08968*, 2026a. doi: 10.48550/arXiv.2602.08968.

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. doi: 10.48550/arXiv.2603.19312.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

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

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012.

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 *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065, 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999. doi: 10.1023/A:1010091220143.

Junhan Sun, Hao Zhao, and Guofeng Zhang. INTACT: Isomorphic intent-to-action learning for search-free world models. *arXiv preprint arXiv:2607.26056*, 2026. doi: 10.48550/arXiv.2607.26056.

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

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020. URL https://iclr.cc/virtual_2020/poster_H1exf64KwH.html.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided imagination sampling in world Models. *arXiv preprint arXiv:2606.07974*, 2026. doi: 10.48550/arXiv.2606.07974.

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 *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

APPENDIX

APPENDIX CONTENTS

A. Implementation and Evaluation Details

B. The DINO-WM-style Host

C. Intent Parameterization

D. The Reweighting Layer

D.1 Sparsity sweep

D.2 Reusing the layer for terminal ranking

E. Search Budget and Cost

E.1 Budget sweep and proposal diagnostics

E.2 Over-search

E.3 Wall-clock cost

F. What the Initial Mean Can and Cannot Buy

F.1 Offline action-block spectra

F.2 A second host planner

G. Relation to the Closest Prior Work

A IMPLEMENTATION AND EVALUATION DETAILS

Table 6 lists the shared configuration. The Ours and SMWM hosts pair a ViT-tiny encoder with a six-layer transformer predictor; the DINO-WM-style host keeps every planning setting and differs in encoder and latent width (Appendix B). The reweighting layer s_ϕ and the intent head h_ψ are two-layer MLPs with 256 hidden units and together hold 0.25M parameters beside the 18.2M-parameter frozen host.

Datasets. PushT, TwoRoom, Reacher, and Cube use the offline datasets released with LeWM (Maes et al., 2026b). We collected the Cube-Double dataset ourselves in the two-cube configuration of the OGBench-derived Cube environment (Park et al., 2025): 9,000 episodes of 200 actions each, rendered as single-view 224×224 RGB frames and generated by OGBench's scripted pick-and-place plan oracle with temporally smoothed action noise (scale 0.1, smoothing 0.5). Each dataset is split by episode, 90% for training and 10% held out, and planning evaluation draws its start and goal frames from the held-out episodes (Figure 3).



Figure 3: **Example evaluation episodes.** Current observation x_t and goal image x_g from one held-out episode each of Cube and Cube-Double.

Table 6: Shared training and planning hyperparameters; I CEM iterations give the budget $B = 300I$.

Training	Value	Planning	Value
Image size	224×224	Horizon	5
Patch size	14	Receding/action block	5/5
Latent dimension	192	CEM samples	300
Batch size	256	CEM elites	30
Frozen AIM epochs	3	CEM iterations	1–30
World-model updates	none	Initial variance	1.0
AdamW weight decay	10^{-3}	Evaluation seed	42
Frozen AIM learning rate	10^{-4}	Rollouts per condition	100
Intent/sparse weight	$0.1/0.01$	Frame skip	5 env. steps
SIGReg weight	0.09	Goal offset/budget	25/50
DCT type/norm/K	II/ortho/2		

B THE DINO-WM-STYLE HOST

The DINO-WM-style host replaces the trained ViT-tiny encoder with a frozen DINOv2 ViT-S/14 (Oquab et al., 2024), following DINO-WM (Zhou et al., 2025), and reads its class token, so each frame becomes a single latent of width $D = 384$. This global token tracks the pose of the T block but is largely insensitive to the pusher's position, which the Ours latent and pooled DINOv2 patch tokens both reflect; since most short candidate sequences change only where the pusher ends up, the terminal cost can barely rank them, consistent with PushT being the task this host barely solves. Its six-layer transformer predictor is trained on the same offline data as the other hosts and then frozen. AIM is attached with the same definition, loss weights, learning rate, and three-epoch schedule; planning uses the settings of Table 6. The host was trained and evaluated by a separate implementation that shares the task datasets, goal construction, success criterion, and planner configuration.

C INTENT PARAMETERIZATION

For each task, all five variants are trained on the same frozen Ours (SIGReg) checkpoints: only the reweighting layer and intent head are trained, for three epochs, with every other setting fixed. DCT-$K1/K2/K3/K5$ predict the first K DCT coefficients and Direct-H5 the whole five-step block. Planning compares default with intent-guided initialization on each checkpoint. This cohort is trained independently of Table 1, so its DCT-$K2$ values differ slightly from that table; the prior-only comparison of Section 6.3 uses the same cohort. Figure 4 and Table 7 report the results at $B = 300$.

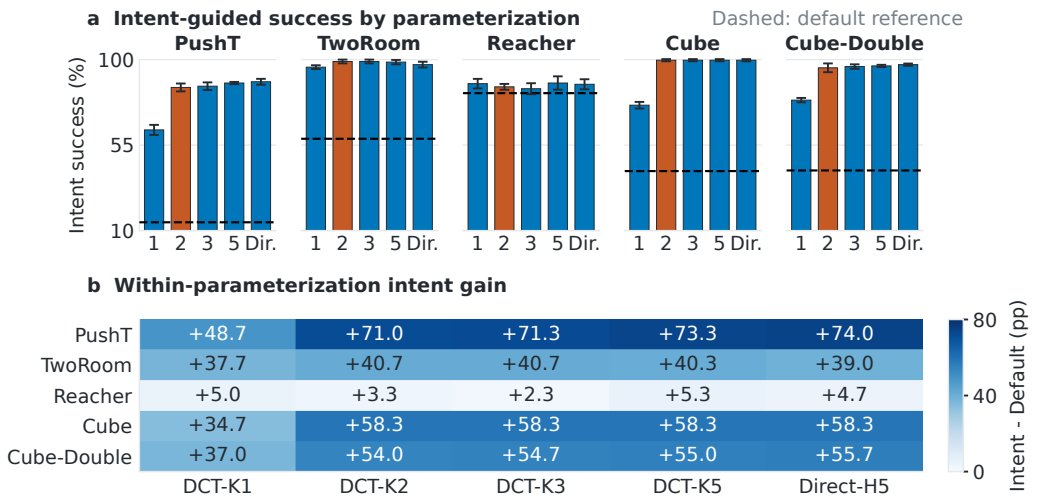


Figure 4: **Intent parameterization at $B = 300$.** (a) Intent-initialized success (mean $\pm$ sample SD); orange marks DCT-$K2$, and dashed lines mark each task's mean default-initialization success. (b) Intent minus default, from unrounded means.

Table 7: Intent parameterization at $B = 300$: success rate (%, mean $\pm$ sample SD). Δ is Intent minus Default in percentage points.

Task	Intent target	Default	Intent	Δ (pp)
PushT	DCT-$K1$	14.33±1.15	63.00±2.65	+48.67
	DCT-$K2$	14.33±1.15	85.33±2.08	+71.00
	DCT-$K3$	14.67±1.53	86.00±2.00	+71.33
	DCT-$K5$	14.33±1.15	87.67±0.58	+73.33
	Direct-H5	14.33±1.15	88.33±1.53	+74.00
TwoRoom	DCT-$K1$	58.33±2.52	96.00±1.00	+37.67
	DCT-$K2$	58.33±2.52	99.00±1.00	+40.67
	DCT-$K3$	58.33±2.52	99.00±1.00	+40.67
	DCT-$K5$	58.33±2.52	98.67±1.15	+40.33
	Direct-H5	58.33±2.52	97.33±1.53	+39.00
Reacher	DCT-$K1$	82.33±2.31	87.33±2.52	+5.00
	DCT-$K2$	82.33±2.31	85.67±1.53	+3.33
	DCT-$K3$	82.33±2.31	84.67±2.89	+2.33
	DCT-$K5$	82.33±2.31	87.67±3.51	+5.33
	Direct-H5	82.33±2.31	87.00±2.65	+4.67
Cube	DCT-$K1$	41.33±0.58	76.00±1.73	+34.67
	DCT-$K2$	41.33±0.58	99.67±0.58	+58.33
	DCT-$K3$	41.33±0.58	99.67±0.58	+58.33
	DCT-$K5$	41.33±0.58	99.67±0.58	+58.33
	Direct-H5	41.33±0.58	99.67±0.58	+58.33
Cube-Double	DCT-$K1$	41.67±2.52	78.67±1.15	+37.00
	DCT-$K2$	41.67±2.52	95.67±2.31	+54.00
	DCT-$K3$	41.67±2.52	96.33±1.15	+54.67
	DCT-$K5$	41.67±2.52	96.67±0.58	+55.00
	Direct-H5	41.67±2.52	97.33±0.58	+55.67

D THE REWEIGHTING LAYER

For continuous weights we summarize effective support by

$$N_{\text{eff}}(m) = \frac{(\sum_i m_i)^2}{\sum_i m_i^2}. \tag{8}$$

Table 8 compares the learned layer with $m \equiv \mathbf{1}$. The learned layer fits the intent better on all five tasks, by up to 0.018 in validation R^2, and both reach the same five-task success of 92.93% at

$B = 300$, with per-task differences from -2.33 to $+2.67$ pp. At the default sparsity weight, the learned m concentrates on an effective 26 to 128 of the 192 latent dimensions, depending on the task.

Table 8: **Learned reweighting against $m \equiv \mathbf{1}$.** Both variants train the same intent head on the same frozen Ours (SIGReg) backbones under the same protocol; $m \equiv \mathbf{1}$ fixes every weight to one. Both Δ columns are learned minus $m \equiv \mathbf{1}$. N_{eff} and top-4 mass (the share of $\sum_i m_i$ on the four largest weights) describe the learned m of one checkpoint over D=192 latent dimensions, averaged over current–goal pairs; they are not available for Cube-Double.

	Validation spectral R^2			Planning success at B=300 (%)			Learned m	
Task	learned	$m \equiv \mathbf{1}$	Δ	learned	$m \equiv \mathbf{1}$	Δ	N_{eff}	top-4
PushT	0.8080	0.7897	+0.0182	84.67	82.00	+2.67	104.0	0.099
TwoRoom	0.1746	0.1734	+0.0012	99.00	99.67	−0.67	67.6	0.142
Reacher	0.0861	0.0850	+0.0011	86.67	89.00	−2.33	26.1	0.304
Cube	0.6529	0.6496	+0.0034	99.00	99.67	−0.67	127.6	0.065
Cube-Double	0.6367	0.6227	+0.0140	95.33	94.33	+1.00	–	–
Mean				92.93	92.93	+0.00		

D.1 Sparsity sweep

The sweep retrains the reweighting layer and intent head on one frozen Ours checkpoint each for PushT and Cube, varying λ_{sparse}, the coefficient of $\mathcal{L}_{\text{sparse}}$ in Equation 5 (0.01 by default), over 0 and 10^{-4} to 3×10^{-1} in half-decade steps. From 0 to 3×10^{-2} the reweighting concentrates from $N_{\text{eff}}/D = 0.861$ to 0.343 on PushT and from 0.949 to 0.445 on Cube, while validation R^2 moves from 0.808 to 0.795 and from 0.654 to 0.643. Each checkpoint is planned under two interfaces. Over the same range, with AIM initialization and the full terminal cost at $B = 300$, success stays within 84–89% on PushT and 99–100% on Cube, with no monotone trend. With default initialization and the reweighted terminal cost of Appendix D.2 at $B = 9{,}000$, it stays within 88–92% and 60–64%. Up to 3×10^{-2}, the sparsity penalty thus more than halves the effective support at essentially no cost to prediction or planning. Success under AIM falls only at the two largest weights, $\lambda_{\text{sparse}} = 0.1$ and 0.3: to 82 and 78% on PushT and to 95 and 85% on Cube.

D.2 Reusing the layer for terminal ranking

AIM changes only the initial mean of Equation 6. As a separate planner-interface analysis, m can also weight the terminal comparison that ranks candidates:

$$c_m(\hat{z}_{t+H}, z_g) = \frac{\sum_{i=1}^{D} m_i (\hat{z}_{t+H,i} - z_{g,i})^2}{\sum_{i=1}^{D} m_i + 10^{-8}}, \tag{9}$$

in place of the full terminal cost J of Section 3, used in all main results. The weights are computed once from the real current–goal pair at each MPC step and held fixed throughout that CEM solve; the dynamics rollout stays on the full latent.

Table 9 crosses the two choices at $B = 9{,}000$. AIM initialization raises success on every task under either cost, by $+2.67$ to $+35.00$ pp. Reweighting the terminal cost changes success by $+1.33$, $+5.00$, -2.67, $+2.00$, and $+1.67$ pp across PushT, TwoRoom, Reacher, Cube, and Cube-Double under default initialization, and by $+2.00$ pp on PushT and -0.33 pp on each other task under AIM initialization. The main results therefore keep the full cost J.

Table 9: **The within-checkpoint factorial, as paired differences.** Each column is one contrast on the frozen Ours (SIGReg) backbone, computed on each checkpoint and then averaged, with the sample SD of the paired differences. All contrasts are at B=9,000; the PushT entries come from runs independent of Table 2 and differ slightly from it.

	AIM initialization		Reweighted terminal cost	
Task	full cost	reweighted cost	default init	AIM init
PushT	$+4.33\pm0.58$	$+5.00\pm1.00$	$+1.33\pm0.58$	$+2.00\pm1.00$
TwoRoom	$+10.33\pm2.08$	$+5.00\pm2.65$	$+5.00\pm3.61$	-0.33 ± 0.58
Reacher	$+2.67\pm4.93$	$+5.00\pm6.24$	-2.67 ± 4.51	-0.33 ± 7.37
Cube	$+35.00\pm1.00$	$+32.67\pm2.52$	$+2.00\pm1.73$	-0.33 ± 4.04
Cube-Double	$+21.33\pm4.16$	$+19.33\pm6.11$	$+1.67\pm1.15$	-0.33 ± 1.15

E Search Budget and Cost

E.1 Budget sweep and proposal diagnostics

Closed-loop budget results. The budget study uses the frozen Ours backbone throughout. Horizon, receding horizon, and action block are $5/5/5$; samples, elites, and initial variance are $300/30/1.0$. Goal offset is 25 and interaction budget is 50. Only iteration count varies across budgets; both initializers use the same count. Budgets below 30 iterations are independently executed closed-loop evaluations, not prefixes extracted from a long search.

Candidate counts exclude intent inference; the evaluation times of Appendix E.3 include it.

Table 10: Frozen-Ours budget sweep: success rate (%, mean ± sample SD). Columns give the sampled trajectory budget per solve, $B = 300I$; Δ is Intent minus Default in percentage points.

		Sampled trajectories per solve					
Task		300	900	1,500	3,000	6,000	9,000
PushT	Default	14.00 ± 1.00	64.00 ± 4.00	77.00 ± 1.00	85.33 ± 3.79	88.00 ± 2.00	87.33 ± 0.58
	Intent	84.67 ± 2.08	88.67 ± 0.58	92.67 ± 2.89	94.33 ± 1.53	94.33 ± 2.08	92.33 ± 1.53
	Δ	$+70.67$	$+24.67$	$+15.67$	$+9.00$	$+6.33$	$+5.00$
TwoRoom	Default	58.33 ± 2.52	84.67 ± 1.53	90.00 ± 2.65	90.67 ± 2.89	93.33 ± 1.15	89.67 ± 2.08
	Intent	99.00 ± 0.00	98.67 ± 0.58	100.00 ± 0.00	99.67 ± 0.58	98.67 ± 0.58	100.00 ± 0.00
	Δ	$+40.67$	$+14.00$	$+10.00$	$+9.00$	$+5.33$	$+10.33$
Reacher	Default	82.33 ± 2.31	81.33 ± 2.08	75.67 ± 1.15	70.33 ± 3.79	71.67 ± 5.51	69.33 ± 2.08
	Intent	86.67 ± 3.21	80.33 ± 3.21	75.67 ± 1.53	69.33 ± 0.58	70.00 ± 2.65	72.00 ± 4.00
	Δ	$+4.33$	-1.00	$+0.00$	-1.00	-1.67	$+2.67$
Cube	Default	41.33 ± 0.58	56.67 ± 1.15	61.67 ± 1.15	62.33 ± 2.08	62.67 ± 4.16	60.67 ± 1.15
	Intent	99.00 ± 1.00	99.67 ± 0.58	99.00 ± 1.00	98.33 ± 0.58	96.00 ± 2.65	95.67 ± 0.58
	Δ	$+57.67$	$+43.00$	$+37.33$	$+36.00$	$+33.33$	$+35.00$
Cube-Double	Default	41.67 ± 2.52	65.67 ± 3.51	72.33 ± 2.31	71.67 ± 2.08	70.00 ± 1.73	68.33 ± 3.79
	Intent	95.33 ± 0.58	97.33 ± 0.58	94.67 ± 0.58	95.00 ± 1.00	91.67 ± 2.52	89.67 ± 0.58
	Δ	$+53.67$	$+31.67$	$+22.33$	$+23.33$	$+21.67$	$+21.33$

Initial proposals and within-search optimization. Figure 5 follows CEM through the 30 iterations of the first MPC decision on the frozen Ours host. After each iteration CEM selects the mean of its elite candidates as its plan, the sequence it would execute if it stopped there; the figure plots the terminal cost that the world model predicts for this plan, averaged over all evaluation starts.

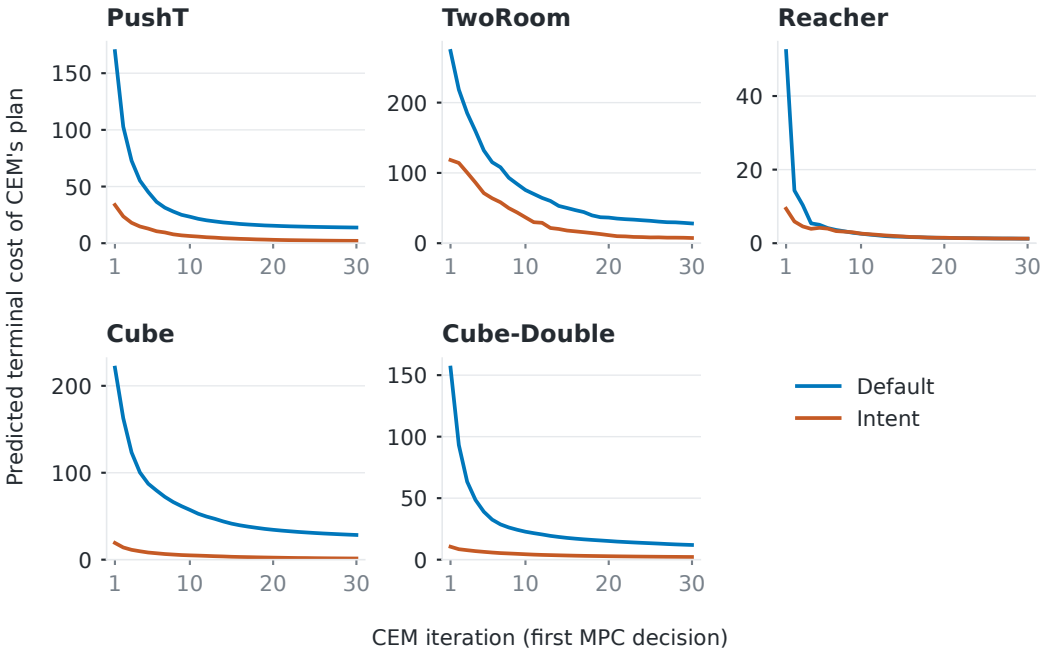


Figure 5: **First-decision CEM traces on the frozen Ours host.** Predicted terminal cost of CEM's plan, the mean of its elite candidates, after each of 30 iterations under Default and Intent initialization, averaged over all evaluation starts. Lower is better; vertical scales differ across tasks.

Table 11: Predicted terminal cost J of the two initial means at the first MPC decision, on the same starts as Figure 5, with $\Delta J = J(\mu_0^{\text{intent}}) - J(\mu_0^{\text{default}})$ on each start. Costs are on each task's own scale.

Task	Share of starts with $\Delta J < 0$	Mean ΔJ
PushT	0.943	-188.673
TwoRoom	0.783	-159.276
Reacher	0.973	-216.778
Cube	0.977	-243.873
Cube-Double	0.973	-207.356

On every task AIM's initial mean has the lower predicted cost on 78–98% of paired starts, so the search begins from a better point of the learned objective. On the same start-averaged traces of the first MPC decision, default CEM needs three to eight iterations to reach the predicted cost of AIM's plan after one iteration on PushT, TwoRoom, and Reacher; on Cube it does not reach it within 30 iterations on any checkpoint, and on Cube-Double on only one, at iteration 28.

E.2 Over-search

Table 12 reports closed-loop success at $B = 300$ and $B = 9{,}000$ for every task, host, and initializer. Table 10 gives the full six-point Ours sweep with sample SDs.

While success declines with budget on several tasks, optimization of the learned objective does not stall. Averaged over all starts at the first real MPC step, the selected sequence's predicted terminal cost falls from iteration 1 to iteration 30 by $12.2\times$ (PushT default), $16.2\times$ (PushT AIM), $9.7\times$ (TwoRoom default), $16.0\times$ (TwoRoom AIM), $42.4\times$ (Reacher default), $7.5\times$ (Reacher AIM), $7.8\times$ (Cube default), $14.1\times$ (Cube AIM), $13.1\times$ (Cube-Double default), and $4.8\times$ (Cube-Double AIM). Reacher shows the largest reduction in predicted cost of any task and the largest loss in closed-loop success.

Table 12: **Success at 300 against 9,000 sampled trajectories.** Each cell is B=300 → B=9,000; Avg. is the five-task mean, and **bold** marks a decrease.

Host	Initializer	PushT	TwoRoom	Reacher	Cube	Cube-Double	Avg.
Ours (SIGReg)†	Default CEM	14.0 → 87.3	58.3 → 89.7	**82.3 → 69.3**	41.3 → 60.7	41.7 → 68.3	47.5 → 75.1
	AIM-CEM	84.7 → 92.3	99.0 → 100.0	**86.7 → 72.0**	**99.0 → 95.7**	**95.3 → 89.7**	**92.9 → 89.9**
SMWM†	Default CEM	33.7 → 87.3	85.0 → 90.7	**84.3 → 74.3**	44.3 → 86.3	39.7 → 70.3	57.4 → 81.8
	AIM-CEM	82.3 → 89.7	96.3 → 97.0	**88.3 → 69.0**	**100.0 → 96.7**	**99.0 → 95.3**	**93.2 → 89.5**
DINO-WM-style†	Default CEM	1.7 → 8.0	59.0 → 91.0	20.7 → 40.7	42.3 → 46.3	35.3 → 42.3	31.8 → 45.7
	AIM-CEM	1.7 → 7.7	**98.7 → 95.3**	27.3 → 43.7	**62.3 → 57.3**	**51.7 → 50.0**	48.3 → 50.8

E.3 WALL-CLOCK COST

Table 13 reports the recorded evaluation time per condition. Running AIM at $B = 300$ instead of Default CEM at $B = 9{,}000$ takes 5.2–12.2× less time, and Table 2 shows AIM at $B = 300$ ahead of Default at $B = 9{,}000$ on four of five tasks.

Table 13: **Recorded evaluation time per condition**, in seconds. Overhead is AIM divided by Default at the same budget; the last column is Default at $B = 9{,}000$ divided by AIM at $B = 300$. PushT times come from a dedicated timing run on an idle machine, the others from the runs that produced the success rates.

	$B = 300$			$B = 9{,}000$			
Task	Default	AIM	overhead	Default	AIM	overhead	$\text{Default}_{9k}/\text{AIM}_{300}$
PushT	29.2	28.9	0.989×	351.2	351.2	1.000×	12.2×
TwoRoom	34.7	35.6	1.026×	432.8	434.1	1.003×	12.1×
Reacher	76.7	96.7	1.260×	665.7	647.1	0.972×	6.9×
Cube	129.5	125.4	0.968×	664.7	662.6	0.997×	5.3×
Cube-Double	127.4	123.2	0.967×	642.2	625.7	0.974×	5.2×

All times include environment stepping and rendering, which do not scale with the candidate budget, so the ratio in the last column is smaller than the 30× reduction in sampled trajectories.

F WHAT THE INITIAL MEAN CAN AND CANNOT BUY

This appendix derives the first statement of Section 4.3 and reports the evidence for both.

Derivation of the decomposition. Write the connecting block as $a \in \mathbb{R}^{L \times d_a}$ and its coefficients as $q = \mathcal{D}a$ under the orthonormal transform of Equation 2. AIM's initial mean is $\mu^{(0)} = \mathcal{D}^{-1}\hat{q}^{(K)}$, where $\hat{q}^{(K)}$ carries the K predicted coefficients and zeros elsewhere. Because $\mathcal{D}$ is orthonormal, Parseval gives $\|\mu^{(0)} - a\|^2 = \|\hat{q}^{(K)} - q\|^2$, and since the two vectors agree on no coefficient outside the first K,

$$\mathbb{E}\|\mu^{(0)} - a\|^2 = \sum_{k<K} \mathbb{E}\big(\hat{q}_k - q_k\big)^2 + \sum_{k\geq K} \mathbb{E}\, q_k^2.$$

The first sum is prediction error on the retained coefficients, the second the energy thrown away by truncation. Because $R_k^2 = 1 - \mathbb{E}(\hat{q}_k - q_k)^2 / \operatorname{Var}(q_k)$ is the validation coefficient of determination, the k-th prediction term equals $(1 - R_k^2)\operatorname{Var}(q_k)$, which gives Equation 7 once $\operatorname{Var}(q_k)$ is taken from the offline spectra. The action spectra of Appendix F.1 are per-block energy fractions computed on centered, per-coordinate standardized actions, so the coefficients are zero-mean, $\operatorname{Var}(q_k) = \mathbb{E}q_k^2 = e_k$, the five e_k sum to 1, and the default proposal $\mu^{(0)} = 0$ sits at $E = 1$ exactly. Moving from K to $K + 1$ therefore changes the error by $-R_K^2 e_K$: a mode is worth its energy times its predictability, and a mode predicted no better than zero costs error rather than saving it.

What the criterion separates. Table 5 evaluates every step of the parameterization sweep. The three steps with a fractional error reduction of at least 0.1927 each gained +17.00 to +23.67 pp;

the other twelve are at or below 0.1690 and change success by -1.67 to $+3.00\,\mathrm{pp}$, so any τ in $(0.1690, 0.1927]$ separates the two groups. The same quantity computed against the default proposal ranks the headline result: error reductions of 0.473, 0.397, 0.393, 0.068, and 0.030 on PushT, Cube-Double, Cube, TwoRoom, and Reacher against AIM gains of $+70.67$, $+53.67$, $+57.67$, $+40.67$, and $+4.33\,\mathrm{pp}$ at $B = 300$ on Ours, a rank correlation of 0.90. As in Section 4.3, the decomposition ranks tasks and parameterizations rather than predicting success rates.

Why the advantage persists, and where it cannot. A sampling-based refiner on a finite budget evaluates only the candidates it draws, so the region its iterate settles in can depend on where it started. How much that matters is an empirical question: the advantage persists under CEM's elite refit (Section 6.2) and under MPPI's softmax reweighting, which never narrows its sampling width at all (Appendix F.2). Where the two initializations do end up at the same point of the learned objective, no initialization advantage can remain, and the closed-loop outcome is set entirely by how well that learned optimum agrees with task success. Table 14 measures how far each task is from that situation, using the ratio of the predicted cost of CEM's plan under default and under AIM initialization after all 30 iterations of the first real MPC decision. Reacher is the converged case, with ratios of 0.97–1.00: its advantage moves from $+4.33$ to $+2.67\,\mathrm{pp}$, within the sample SD, while *both* initializations lose 13–$15\,\mathrm{pp}$ of success—the search reaches a better point of the learned objective and a worse point of the task. A persisting cost gap, conversely, does not by itself preserve the advantage: PushT keeps a ratio of 4.7–$8.3\times$ yet retains 5.00 of its $70.67\,\mathrm{pp}$, while Cube, at 20.6–$21.0\times$, retains $35.00\,\mathrm{pp}$.

On the two tasks also run with the width-fused prior of Appendix G, the proposal does not decide the direction of the budget response: on Reacher success falls with budget under both initializations and with the fused prior, while on PushT it rises under all three. The loss of AIM's advantage between the two budgets also ranks the tasks alike under CEM and MPPI ($\rho = 0.90$, Appendix F.2).

Table 14: **Where CEM's two initializations converge.** The ratio is the predicted cost of CEM's plan under default over that under AIM initialization, after all 30 iterations of the first real MPC decision, as its range across checkpoints; at 1 the two initializations have reached the same point of the learned objective. AIM advantage is AIM-CEM minus Default CEM success on the frozen Ours host, from Table 2.

	Cost ratio	AIM advantage (pp)	
Task	at iteration 30	at B=300	at B=9,000
Reacher	0.97–1.00	+4.33	+2.67
TwoRoom	2.90–4.60	+40.67	+10.33
Cube-Double	3.76–8.44	+53.67	+21.33
PushT	4.66–8.30	+70.67	+5.00
Cube	20.63–21.00	+57.67	+35.00

Scope. The ranking needs only offline data and validation fits; the threshold τ is set on the same 15 steps it separates. The first equality of Equation 7 is exact. The second takes the coefficient variances from the offline spectra of $L = 5$ blocks, whereas validation pairs also span shorter separations, which use $K_{\mathrm{eff}} = \min(K, L)$. The convergence analysis uses the frozen Ours host and CEM.

F.1 Offline action-block spectra

Section 6.4 distinguishes two reasons a temporal mode may be unhelpful: the frozen latent may not predict it, or the offline actions may not contain it. Table 15 measures the second directly, with no model involved. Following the dataset's action-chunk construction, an action block is $L \cdot \mathrm{frameskip} = 25$ consecutive environment actions reshaped to $L = 5$ model steps, each the concatenation of five environment actions. The orthonormal DCT of Equation 2 is applied along the model-step axis and per-coefficient energy is averaged over macro-action coordinates and over 20,000 sampled windows per task.

Table 15: **Fraction of action-block energy per temporal mode.** Computed on the evaluation split of each task's offline dataset after per-coordinate standardization; raw and standardized spectra agree to within 0.02 on every entry. A temporally white block places 0.200 in every coefficient; the last column is the largest deviation from that value.

Task	k=0	k=1	k=2	k=3	k=4	$\max_k \lvert e_k - 0.2 \rvert$
PushT	0.3282	0.2692	0.1919	0.1273	0.0834	0.1282
TwoRoom	0.2388	0.1923	0.1896	0.1891	0.1901	0.0388
Reacher	0.2003	0.2007	0.1998	0.1997	0.1996	0.0007
Cube	0.3156	0.3162	0.1749	0.1123	0.0810	0.1190
Cube-Double	0.3410	0.3187	0.1659	0.1009	0.0735	0.1410

The first temporal mode, $k = 1$, holds 0.19 to 0.32 of the block energy on every task, so no task lacks it. What sets TwoRoom and Reacher apart is the shape of the spectrum: both are close to white, within 0.039 and 0.0007 of 0.200 in every coefficient, as step-to-step variation would be, whereas PushT, Cube, and Cube-Double concentrate 60–66% of the energy in the two retained coefficients. Table 5 in the main text combines these energies with the predictability of each mode through Equation 7.

F.2 A SECOND HOST PLANNER

Table 3 attaches the unchanged module to MPPI (Williams et al., 2017), whose softmax-weighted update replaces CEM's elite refit and which never narrows its sampling width—its recorded proposal standard deviation is 1.00 at the first iteration and 1.00 at the thirtieth on all five tasks. Everything else is held at the values used throughout: the same frozen backbones and intent predictors, the same evaluation starts, and the same terminal cost. AIM improves all five tasks at B=300 ($+4.67$ to $+45.33$ pp) and all five again at B=9,000 ($+1.33$ to $+26.67$ pp).

Two things follow. First, an initial mean is worth setting under either update rule, so its value does not depend on CEM's narrowing of the proposal. Second, the tasks that lose the most advantage between the two budgets are the same under both solvers: the per-task decline here ranks with the CEM decline of Table 12 at $\rho = 0.90$ over five tasks. AIM helps most where the default proposal leaves the most room: MPPI's default reaches 52.67% on PushT and 90.33% on TwoRoom at B=300, against 14.00% and 58.33% for CEM, and the same prior adds $+31.33$ pp on PushT under MPPI against $+70.67$ pp under CEM.

G RELATION TO THE CLOSEST PRIOR WORK

PRISM (Wang et al., 2026) is the closest method: it learns a goal-conditioned Gaussian action prior from frozen JEPA features and fuses it into MPPI's sampling distribution, setting both the mean and the width of the proposal, whereas AIM writes only the initial mean. PRISM reports that its mean-only variant declines as more samples are drawn and attributes this to samples drifting around an uncalibrated mean. On our host the decline persists when the width is set as well. Keeping AIM's mean and shrinking the initial CEM width to the task-level precision-weighted value $\sqrt{E/(1+E)}$, with $E = E(2)$ from Equation 7, raises success at both budgets but leaves the budget response unchanged: from $B = 300$ to 9,000, PushT moves $88.00 \to 95.67$ against $84.67 \to 92.33$ with the mean alone, and Reacher $93.33 \to 79.00$ against $86.67 \to 72.00$. Beyond the interface, our setting adds a host pair matched in everything but the pretraining objective, the error decomposition that ranks temporal parameterizations before planning, and a measurement of how much of the gain the proposal carries on its own.