EXABS: LEARNING EXECUTION-AWARE ABSTRACTIONS FOR HIERARCHICAL WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical world models extend planning horizons by letting a high-level model propose waypoints for a short-horizon controller to reach. Their temporal abstractions, however, are typically learned from offline trajectories without regard to the controller that must realize them: a transition that is easy to predict may be infeasible for the controller within its budget, and similarity between full latent states can overlook the specific change a waypoint requests. We propose *execution-aware abstraction learning* (ExAbs), in which the controller shapes both what the hierarchy learns to request and how each request is pursued. For the former, an Achievability Model (AM) learns from closed-loop execution the probability that the controller completes a waypoint within a given budget; AM selects the segment boundaries that train the high-level model and scores candidate waypoints during planning. For the latter, Planner-Executor Transition Abstraction Learning (PETAL) distills the decisions of a frozen planner into compact, source-relative effect representations, so that local action selection compares requested and predicted changes rather than endpoint proximity alone. Collecting AM labels with the PETAL-enabled controller ties the learned temporal units to the executor that carries them out. Across seven simulated tasks on four world-model backbones and three physical-robot tasks, ExAbs improves success over the hierarchical baseline on every task without lengthening the local planning horizon, e.g., from 77.5% to 90.0% on hard Maze queries and from 0% to 21.7% on a sequential Button–Window task, while also shortening successful executions.

1 INTRODUCTION

When reaching for a cup behind a vase, we rarely move straight toward the cup. We first bring the hand to a point that clears the vase, and we choose that point knowing that the arm can get there. Such plans are not formed by the cortex alone: during movement preparation, motor cortex and cerebellum interact in a closed loop (Gao et al., 2018), and the cerebellum is thought to supply internal models that predict the consequences of motor commands (Wolpert et al., 1998; Miall et al., 2007). What is planned is thus informed by what the motor system can execute, and execution is guided by the change the plan requests.

Hierarchical world models adopt a similar division of labor. Latent world models plan by predicting the outcomes of candidate actions (Hafner et al., 2019; Zhou et al., 2025; Sobal et al., 2025; Maes et al., 2026), but search spaces grow and prediction errors compound as horizons lengthen. Hierarchies pair a high-level model that predicts temporally extended transitions with a short-horizon controller that plans primitive actions toward the resulting waypoints (Gumbsch et al., 2024; Zhang et al., 2026b). Unlike the cortico-cerebellar loop, however, the two levels are typically learned in isolation, which leaves two gaps between what the hierarchy requests and what its controller realizes.

The first gap concerns *which transitions to request*. Temporal abstractions are usually derived from predictive structure, such as adaptive intervals, predictable events, or changes in latent context (Neitz et al., 2018; Jayaraman et al., 2019; Gumbsch et al., 2024), or from skills trained for predictable outcomes (Hafner et al., 2022; Gürtler & Martius, 2025). A transition that is predictable in the data need not be reproducible by the deployed controller within its budget; for HWM, Caselli et al. (2026) find that unconstrained high-level search can select subgoals that are poor control targets. Hierarchical reinforcement learning mitigates this online by relabeling high-level experience or

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.

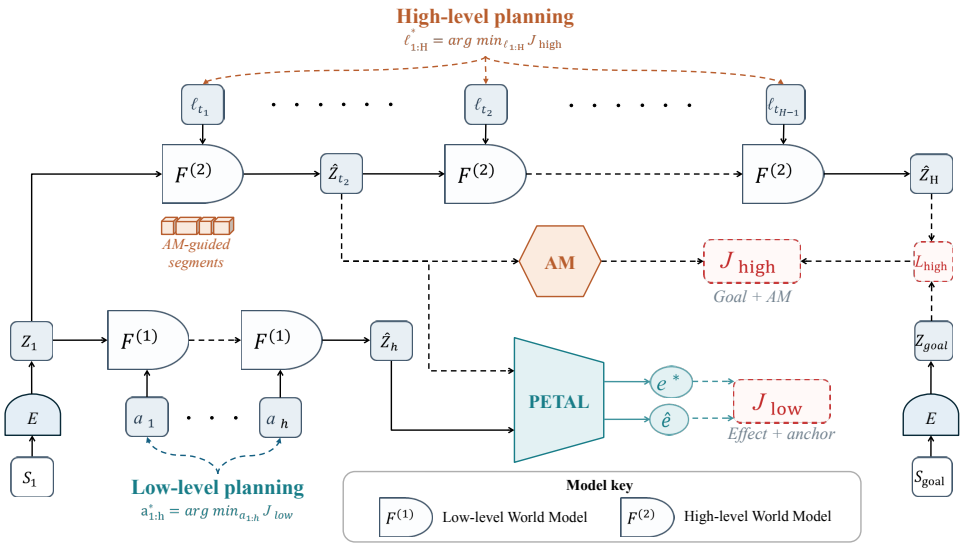

Figure 1: **Execution-aware hierarchical control.** AM scores high-level transitions by their completion probability under the local controller; PETAL compares requested and predicted effects for local control. E is the observation encoder; $F^{(1)}, F^{(2)}$ are the low- and high-level predictors; $J_{\text{high}}, J_{\text{low}}$ are the costs J_H, J_L, and L_{high} is the goal term. Source conditioning and backbone-specific goal aggregation are omitted.

testing subgoals under the current controller (Nachum et al., 2018; Levy et al., 2019). In offline predictive hierarchies the issue runs deeper, because segment boundaries define both the action chunks encoded as macro-actions and the endpoints the high-level model learns to predict. Choosing boundaries by what the controller can complete thus shapes the temporal units themselves.

The second gap concerns *how a request is pursued*. Predictive planners typically score candidate actions by the similarity between predicted and requested latent states. Full-state similarity is shaped largely by content that the actions do not change and can underweight the displacement or contact change that the waypoint actually requests. Planning supervision can shape goal metrics (Srinivas et al., 2018), and temporal embeddings support goal-directed behavior (Park et al., 2024), but neither isolates the change requested by a particular waypoint. What the controller needs instead is to compare the *requested effect* with the *predicted effect* of each candidate action.

We propose *execution-aware abstraction learning* (ExAbs), which closes both gaps by letting the controller shape the abstractions of the hierarchy (Figure 1). For the first gap, an Achievability Model (AM) learns from closed-loop outcomes the probability that a given controller completes a waypoint within a given budget; it selects the offline segment boundaries that train the macro-action encoder and high-level predictor, and scores candidate waypoints during planning. For the second, Planner-Executor Transition Abstraction Learning (PETAL) distills the decisions of a frozen planner into compact, source-relative effect representations; local search then compares requested and predicted effects while retaining the waypoint as a latent anchor. Because PETAL changes how the controller acts, AM labels are collected with the PETAL-enabled controller, so that the temporal units describe the executor that carries them out (Figure 2). In terms of the opening example, AM lets execution inform the plan, and PETAL expresses each request as the change to be produced.

We test whether closing these gaps improves long-horizon control on seven simulated tasks across four world-model backbones and three physical-robot tasks, covering blocked routes, adverse dynamics, contact-rich manipulation, and ordered interaction. With the same local planning horizon, ExAbs improves success over the hierarchical baseline on every task and reaches goals in fewer environment steps in Maze, PushT, and Cube. Our contributions are threefold:

1. **Execution-aware temporal abstraction.** AM models controller completion as a budget-dependent survival problem and uses it both to select high-level training boundaries and to guide online waypoint selection.

2. **Effect-based, executor-matched local control.** PETAL learns compact effect targets from planner decisions; collecting AM supervision with the PETAL-enabled executor keeps the temporal units consistent with how they are executed.
3. **Empirical validation.** Gains on four backbones and a physical robot, with ablations and diagnostics showing that AM identifies completable waypoints (AUROC 0.79 vs. 0.55 for latent distance) and that PETAL preserves motion information under appearance changes.

2 Related Work

World models and temporal abstraction. Latent world models plan through learned dynamics (Hafner et al., 2019; Hansen et al., 2024), and recent offline models enable visual goal-reaching (Zhou et al., 2025; Sobal et al., 2025; Maes et al., 2026); closed-loop evaluation shows that predictive quality does not guarantee control utility (Zhang et al., 2026a). To extend their horizons, temporal abstractions have been derived by segmenting demonstrations into reusable behaviors (Kipf et al., 2019), from changes in predictive context (Gumbsch et al., 2024), or through learned goal-setting and goal-reaching policies (Hafner et al., 2022). HWM couples primitive and macro-action predictors in a shared latent space (Zhang et al., 2026b), and Hi-LeWM transfers it to LeWM (Caselli et al., 2026). We build on this hierarchy but select its training boundaries by execution feedback, so that the high-level model learns transitions its controller can complete.

Subgoal feasibility and execution competence. Hierarchical RL corrects high-level experience for low-level behavior (Nachum et al., 2018), tests subgoals under the current controller (Levy et al., 2019), or constrains subgoals to be reachable within bounded steps (Zhang et al., 2020). Planning over subgoals with goal-conditioned values or distances (Eysenbach et al., 2019; Nasiriany et al., 2019), optimizing subgoals by local planning cost (Nair & Finn, 2020), and weighting high-level choices by skill affordances (Ahn et al., 2022) likewise ground plans in low-level competence. Time-to-goal estimates provide related signals (Hartikainen et al., 2020; Tiofack et al., 2026). AM differs in two respects: it models the budget-dependent completion distribution of a specific controller from censored closed-loop outcomes, and it shapes the temporal units learned by the high-level model, not only the waypoints selected at test time.

Goal representations for local control. Universal Planning Networks learn goal metrics through planning supervision (Srinivas et al., 2018), and HILP learns temporal embeddings for goal-directed control (Park et al., 2024). PETAL instead learns source-relative transition effects by distilling planner decisions, which lets local search compare requested and predicted changes while preserving the predictive state space of the world model.

3 Learning Execution-Aware Abstractions

ExAbs (Figure 1) learns two abstractions: temporal units for high-level prediction and compact effect representations for local control. The predictive models are trained on offline trajectories. AM is additionally trained on environment executions collected in a preparation stage; it selects high-level training boundaries and scores waypoints online. PETAL learns, from frozen-planner decisions, an effect space in which requested and predicted changes are compared.

3.1 Predictive hierarchy

Given offline observation–action trajectories, we seek to reach the state specified by a goal observation o_g from the current observation o_t. An encoder maps both to latents $z_t = E(o_t)$ and $z_g = E(o_g)$. Following the shared-state hierarchy of Zhang et al. (2026b), a primitive-action predictor $\mathcal{F}_L$ and a macro-action predictor $\mathcal{F}_H$ ($F^{(1)}$ and $F^{(2)}$ in Figure 1) operate in this latent space, and an encoder A_η maps variable-length action chunks to fixed-dimensional macro-actions. After backbone adaptation, E and $\mathcal{F}_L$ are frozen, and AM-selected segments train A_η and $\mathcal{F}_H$. High-level search predicts H transitions toward z_g and passes its first predicted state to the controller as waypoint w. Low-level search plans h actions toward w and executes the first $k \leq h$ before both levels replan from feedback (full predictor definitions in Appendix A). Success then hinges on two questions: whether the controller can complete the requested transition within its budget (Section 3.2), and whether its local objective emphasizes the requested change (Section 3.3).

3.2 TEMPORAL ABSTRACTION FROM EXECUTION FEEDBACK

Let ξ denote the local controller together with its waypoint-completion rule $\mathcal{C}_\xi$. In a labeling attempt, w is held fixed while the controller replans from feedback. The completion time is the first post-action success, $T_\xi = \inf\{b \geq 1 \mid \mathcal{C}_\xi(x_b, w) = 1\}$, where x_b is the measured state after action b and $T_\xi = \infty$ if completion never occurs; attempts still incomplete at the observation limit are right-censored. AM learns the budget-dependent completion distribution

$$F_\xi(B \mid z, w) = \Pr(T_\xi \leq B \mid z, w; \xi) \tag{1}$$

with a discrete-time survival model trained on first-hit and censored records. In contrast to time-to-goal formulations of goal-conditioned values (Tiofack et al., 2026), this distribution characterizes a fixed controller; in contrast to predictive-context criteria (Gumbsch et al., 2024), it is grounded in execution. Calibration and ensemble disagreement yield separate scores for segmentation and planning (Appendix B).

For a trajectory window $\mathcal{W}$, let $\Pi(\mathcal{W})$ be the set of partitions with fixed outer endpoints, a fixed number of segments, and integer durations in $[d_{\min}, d_{\max}]$. Each segment (i, j) is scored at a budget equal to its observed duration through the edge utility $s_{ij} = \log q_\xi^{\text{seg}}(j - i \mid z_i, z_j) - \lambda_u u_\xi^{\text{seg}}(j - i \mid z_i, z_j)$, where u^{seg} measures ensemble disagreement. Dynamic programming selects

$$\mathcal{P}_\xi^*(\mathcal{W}) = \operatorname*{argmax}_{\mathcal{P} \in \Pi(\mathcal{W})} \sum_{(i,j) \in \mathcal{P}} s_{ij}. \tag{2}$$

Fixing the span and segment count prevents a partition from raising its score by covering less of the trajectory or adding waypoints. The selected boundaries determine both the primitive-action chunks encoded by A_η and the observed endpoints predicted by $\mathcal{F}_H$, which are trained jointly with AM frozen. Execution feedback thus directly shapes the temporal units of the high-level model (loss and recurrence in Appendices A and C).

3.3 EFFECT-BASED LOCAL CONTROL

PETAL expresses a requested transition as a compact, source-relative effect target and encodes candidate transitions in the same space. A target-derived spatial support $\kappa^* = K(z_t, w)$ fixes a preprocessing shared by all candidates, and for any endpoint z' the shared effect encoder defines

$$e(z') = \Phi_\psi(z_t, z'; \kappa^*) \in \mathbb{R}^{d_e}. \tag{3}$$

The requested effect is $e^* = e(w)$, and candidate i has effect $\hat{e}_i = e(\hat{z}_i)$, computed from the endpoint already predicted by $\mathcal{F}_L$, so no additional world-model rollout is needed. The comparison scale $\tau_0 = h$ matches the local horizon, whereas variable-length segments define the high-level units.

The discrepancy $D_i = D_\psi(e^*, \hat{e}_i; \tau_0)$ measures effect agreement, while the anchor $A_i = d_{\kappa^*}(\hat{z}_i, w)$ ties candidates with similar effects to the requested destination. Within the lowest-anchor candidate set $\mathcal{I}$, scale matching maps discrepancies to $\widetilde{D}_i$ on the anchor's scale, and the local objective is

$$J_{L,i} = R_{L,i} + \begin{cases} [(1 - \omega)A_i + \omega \widetilde{D}_i]_+, & i \in \mathcal{I}, \\ A_i, & i \notin \mathcal{I}, \end{cases} \tag{4}$$

where $[x]_+ = \max(x, 0)$, ω sets the effect contribution, and $R_{L,i}$ retains the planner's non-goal costs. In the spatial implementation, this anchored terminal comparison replaces the original path-summed goal cost (normalization and ensemble handling in Appendix D).

PETAL is trained on candidate evaluations and decisions recorded from a frozen reference planner. The teacher evaluates predicted trajectories, whereas PETAL compares their endpoints through learned effects. For model predictive path integral (MPPI) optimization (Williams et al., 2017), teacher weights p^T induce an action-sequence update $\Delta\mathbf{a}^T$, and PETAL's comparison costs induce weights p^ψ and an update $\Delta\mathbf{a}^\psi$ over the same candidates. Over query records $\mathcal{Q}$, PETAL minimizes

$$\mathcal{L}_{\text{PETAL}} = \mathbb{E}_{\mathcal{Q}} \left[D_{\text{KL}}(p^T \| p^\psi) + \lambda_a \frac{\|\Delta\mathbf{a}^\psi - \Delta\mathbf{a}^T\|_F^2}{h d_a} \right] + \mathcal{L}_{\text{aux}}, \tag{5}$$

where the KL term matches candidate preferences and the second term matches the induced $h \times d_a$ update, with primitive-action dimension d_a. Auxiliary terms support candidate ranking and include variance and covariance regularization adapted from Bardes et al. (2022). Only PETAL is optimized in this stage (details and the CEM variant in Appendix D).

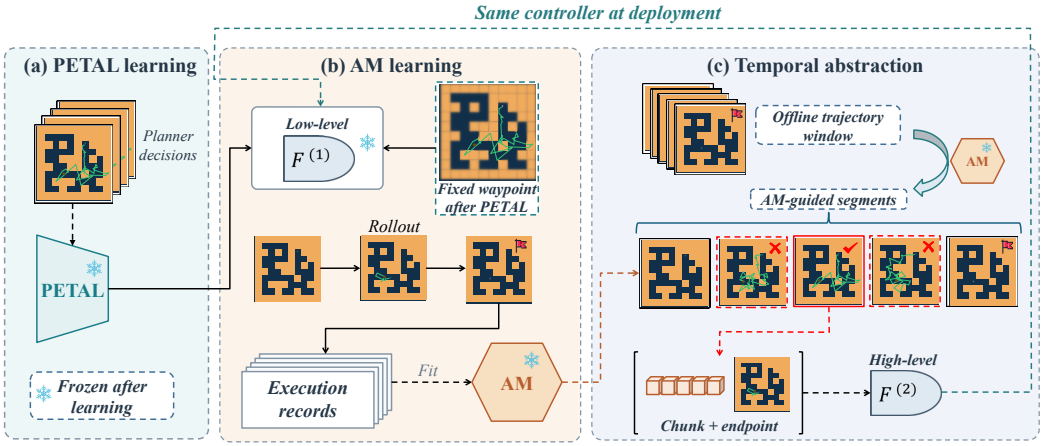

Figure 2: **Execution feedback shapes high-level training units.** (a) Frozen-planner decisions supervise PETAL. (b) Fixed-waypoint execution with the PETAL-enabled controller supplies AM labels. (c) AM selects the action chunks and endpoints that train the macro-action encoder and high-level predictor. Snowflakes mark frozen parameters; the upper dashed connection denotes reuse of the local controller at deployment.

3.4 Executor-matched learning

Because changing the local objective alters actions and completion times, AM must model the PETAL-enabled executor rather than the reference planner that supervised PETAL. Training therefore proceeds in three stages (Figure 2). (a) PETAL is trained on imagined candidates, independently of the final high-level model. (b) PETAL and the remaining local-control configuration are frozen as executor ξ, and fixed-waypoint attempts in the environment collect first-hit and censored outcomes; AM is fitted and calibrated on separate preparation groups and then frozen. (c) AM selects offline boundaries for training the macro-action encoder and high-level predictor. Deployment reuses the labeling executor, and all final evaluation groups are held out (Appendix E).

3.5 Closed-loop planning

At deployment, all learned parameters are fixed. A high-level candidate r has base cost C_H^{base} and first predicted waypoint $\tilde{z}_r^{(1)}$. At the reference budget B_0, AM supplies a risk-adjusted completion score $\rho_r = \rho_\xi^{\text{plan}}(B_0 \mid z_t, \tilde{z}_r^{(1)})$ with floored version $q_r = \max(\rho_r, \epsilon)$, and high-level search minimizes

$$J_H(r) = C_H^{\text{base}}(r; z_g) - \lambda_{\text{AM}} \log q_r + \lambda_{\text{bar}} \frac{[\vartheta_{B_0} - \rho_r]_+}{\max(1 - \vartheta_{B_0}, \epsilon_{\text{bar}})}. \tag{6}$$

The base term retains the backbone's goal aggregation and planning costs, the logarithmic term favors completable waypoints, and the threshold term softly penalizes scores below the calibrated level ϑ_{B_0} (Appendix B). The selected waypoint then defines PETAL's support and target effect, local search optimizes Equation (4), and after k actions both levels replan (Appendices E–G).

4 Experiments

We evaluate whether execution-aware abstractions improve goal attainment when progress depends on intermediate states that a short-horizon controller must realize. Seven simulated tasks with blocked routes, one-way passages, adverse flow, contact, and ordered interaction span four world-model backbones: PLDM (Sobal et al., 2025), LeWM (Maes et al., 2026), JEPA-WMs (Terver et al., 2026), and DINO-WM (Zhou et al., 2025). Three physical-robot tasks require grasping, obstacle clearance, and sequential interaction (Figure 3). Within each task, all methods use the same queries and execution budget, and world-model planners share the same backbone. ExAbs denotes the full framework (AM and PETAL), and HWM the HWM-style hierarchy (Zhang et al., 2026b) on each backbone, which on LeWM is Hi-LeWM online (Caselli et al., 2026).

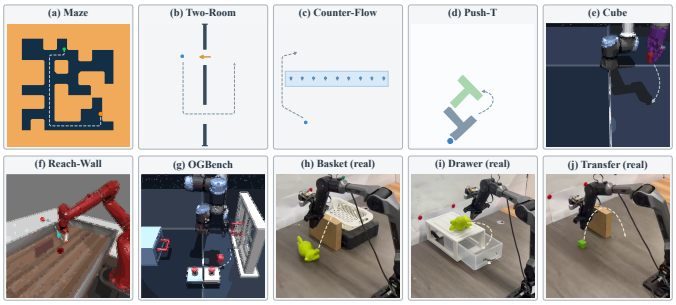

Figure 3: **Evaluation settings.** Seven simulated tasks and three physical-robot tasks. Panel (g) is the adapted Button–Window task.

4.1 Long-range route selection

Setup. In Diverse Maze, walls can separate nearby positions, so a locally attractive approach may lead nowhere, and reaching the goal may require first moving away from it. Queries are grouped into easy, medium, and hard by path length. We compare against goal-conditioned IQL (Kostrikov et al., 2022), HIQL (Park et al., 2023), HILP (Park et al., 2024), flat PLDM, and its HWM hierarchy.[1] Both hierarchical planners use a ten-action local horizon and execute four actions before replanning, so the comparison isolates long-range route selection under short local lookahead.

Table 1: **Maze success (%).**

Method	Easy	Medium	Hard
GCIQL	77.5	35.0	22.5
HIQL	85.0	62.5	42.5
HILP	40.0	12.5	7.5
PLDM	90.0	65.0	57.5
HWM	**100.0**	92.5	77.5
ExAbs	**100.0**	**97.5**	**90.0**

Results. ExAbs matches HWM on easy queries and outperforms all baselines on the harder groups (Table 1), with the largest gain on hard queries (77.5% → 90.0%). Figure 4 shows why: after entering a dead-end region (*Trap*), ExAbs requests a detour out of it (*Escape*) and reaches the goal (*Finish*), whereas HWM keeps requesting direct progress into the blocked region.

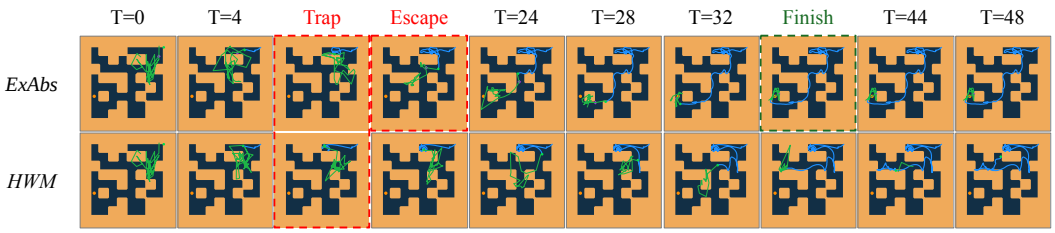

Figure 4: **Escaping an unproductive Maze route.** Observed rollouts from the same query, with ExAbs above HWM. Event labels mark the dead-end region, escape, and completion; overlays show plans and execution.

4.2 Ordered interaction and navigation constraints

Tasks. These tasks separate proximity to the goal from the ability to make progress toward it. In Button–Window, a robot must press a button to unlock a window before sliding the window to its target. In Reach-Wall, an obstacle blocks a robot arm's direct approach. In Two-room, the upper passage permits motion only opposite to the evaluated crossing, so the agent must use the lower passage. In Counterflow Navigation, adverse flow opposes the direct route, while narrow side corridors allow progress. Button–Window uses DINO-WM, Reach-Wall uses JEPA-WMs, and Two-room and Counterflow use separately adapted PLDM models; each compares the flat planner, HWM, and ExAbs (protocols in Appendix H).

[1] All methods are rerun from public code on our data and queries; rates thus differ from Zhang et al. (2026b).

Results. ExAbs achieves the highest success on all four tasks (Table 2). On Button–Window, it is the only method that solves the task, succeeding in 13 of 60 configurations by pressing the button before moving the window (Figure 5). On the other three tasks, it improves over HWM by 11.4, 12.5, and 19.0 points: in Two-room, it reaches the other room through the permitted passage while HWM stays near the blocked approach, and in Counterflow, it bypasses the adverse flow instead of pushing against it (Figure S5 in Appendix M). Together with Maze, these results show that the hierarchy composes intermediate changes that are useful for execution rather than merely close to the final goal.

Table 2: **Ordered-interaction and constraint-task success (%).** Flat is the backbone planner of each task.

Task	Flat	HWM	ExAbs
Button–Window	0.0	0.0	**21.7**
Reach-Wall	32.3	34.4	**45.8**
Two-room	0.0	7.8	**20.3**
Counterflow	26.0	35.5	**54.5**

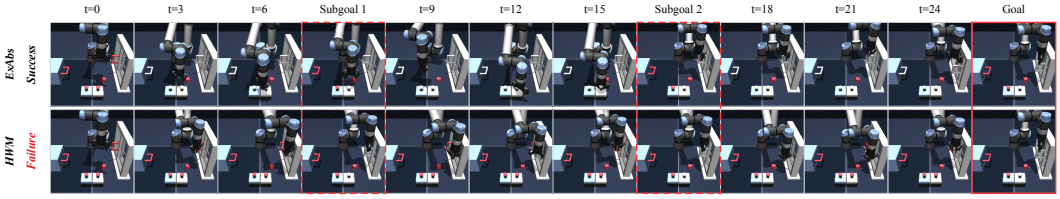

Figure 5: **Button–Window.** ExAbs (top) requests the unlocking button (Subgoal 1) before the window (Subgoal 2); HWM (bottom) fails from the same start and goal.

4.3 Contact-rich manipulation

Table 3: **Manipulation success (%).** Mean ± sample SD over three seeds; bold marks the highest mean. Hi-LeWM variants follow Caselli et al. (2026): C constrains high-level search; staged fixes the subgoal sequence. Full results in Table S1.

Method	$d = 50$	$d = 75$
PushT		
Flat LeWM	38.7 ± 5.0	19.3 ± 4.6
Hi-LeWM online	30.7 ± 5.2	14.0 ± 3.5
Hi-LeWM-C online	39.3 ± 4.2	16.7 ± 6.4
Hi-LeWM staged	38.7 ± 6.1	13.3 ± 6.4
Hi-LeWM-C staged	**44.7** ± 6.4	10.0 ± 2.0
ExAbs	40.7 ± 2.5	**24.0** ± 3.5
Cube		
Flat LeWM	52.0 ± 5.3	61.3 ± 9.0
Hi-LeWM online	56.0 ± 3.5	68.7 ± 8.3
Hi-LeWM-C online	**59.3** ± 3.1	70.0 ± 11.1
Hi-LeWM staged	53.3 ± 5.0	52.7 ± 7.6
Hi-LeWM-C staged	57.3 ± 8.1	48.7 ± 3.1
ExAbs	**59.3** ± 6.4	**72.7** ± 4.6

Tasks and comparisons. PushT and Cube test whether local actions produce the object change requested by the hierarchy. PushT requires translating and rotating a T-shaped block through contact, so useful actions depend on where contact is made and maintained. Cube requires moving an object to a three-dimensional target. Both use LeWM, with start–goal queries sampled at source-trajectory separations $d \in \{25, 50, 75\}$. We compare flat LeWM with the four Hi-LeWM configurations of Caselli et al. (2026), which plan over a frozen LeWM low level; all are rerun from released checkpoints over three seeds (Appendix H.2).

Results. ExAbs outperforms the HWM reference (Hi-LeWM online) at every separation in both tasks and achieves the highest mean among all methods at the longest separation (Tables 3 and S1). On PushT at $d = 75$, it raises success from 14.0% to 24.0%, above the best alternative (19.3%).

Execution efficiency. Higher success does not come at the cost of longer executions. Over each method's own successful trials, ExAbs uses fewer environment steps than both flat planning and HWM in every Maze group and in PushT and Cube, e.g., 20.4 vs. 31.1 on PushT and 219.9 vs. 244.2 on hard Maze (Figure S1, Appendix I).

4.4 Physical-robot execution

Platform and tasks. An AgileX PiPER arm with a parallel gripper performs three image-goal tasks from a fixed RGB camera and end-effector state. *Basket transfer* requires grasping an object and carrying it over an obstacle into a basket; *drawer loading* requires opening a drawer before placing an

object inside; *raised transfer* requires lifting an object over an obstacle without contact and releasing it at a target on the other side.

Setup. All methods share a V-JEPA 2-AC low-level predictor (Assran et al., 2025) adapted to the robot with its visual encoder frozen, and use matched goals, action limits, local horizons, and budgets. The planners use CEM, whose elite selection and distribution updates supervise PETAL; AM labels come from the frozen PETAL-enabled controller. Each method runs on ten held-out start–goal configurations with five repetitions, for 50 trials per task.

Table 4: **Physical-robot success (%).** Fifty trials per method and task. Flat denotes V-JEPA 2-AC.

Task	Flat	HWM	ExAbs
Basket transfer	4.0	20.0	**28.0**
Drawer loading	0.0	10.0	**16.0**
Raised transfer	0.0	24.0	**30.0**

Results. ExAbs achieves the highest success on all three tasks (Table 4), extending the simulation results to physical execution. Its predicted subgoals depict grasping and obstacle clearance (Figure 6), emerging from planning rather than from manually specified task stages (Appendix M.2).



Figure 6: **Physical basket transfer.** Observed execution interleaved with reconstructed model-predicted subgoals (dashed red), followed by the deployment goal (solid red). Display time is normalized.

5 Analysis

5.1 Does AM identify executable transitions?

We evaluate AM on 400 observed transitions from six Maze layouts held out from AM training, using a frozen controller, fixed waypoints, and a ten-action budget. AM separates completed from uncompleted requests far better than latent distance, the backbone's native matching rule (AUROC 0.79 vs. 0.55; Figure 7a), and its probabilities achieve lower Brier error than a validation prior (Appendix J). Better ranking also improves waypoint selection. Among candidates sharing the same source state, controller, and budget, choosing the waypoint with the highest AM score raises completion to 82.3%, compared with 71.5% for minimum latent distance and 67.1% for uniform selection (Figure 7b); the paired gain over latent distance is 10.8 points, with a 95% bootstrap interval of [4.6, 17.3]. Execution feedback therefore supplies completion information that endpoint similarity misses.

5.2 Do PETAL effects capture control-relevant change?

A useful effect target should retain the requested motion while remaining stable under appearance changes. With all control models frozen, we probe PETAL's action-conditioned predicted effects on Reach-Wall. They support more accurate end-effector displacement readout than dimension- and probe-matched random features (17.5 vs. 23.9 mm; Figure 7c), and far more accurate readout than full-state features (51.3 mm), even though PETAL is trained from planner decisions without displacement supervision.

On a separate Reach-Wall cohort, recoloring the scene while keeping the physical transitions fixed shifts PETAL's readout by 63–73% less than that of full-state features, and PETAL's displacement error remains lower under every color condition (Figure 7d). In same-start comparisons in Maze, PETAL's target-induced readout changes align with the executed changes in all 32 target pairs (Appendix K). PETAL effects therefore encode target-specific motion that is stable under the tested recolorings.

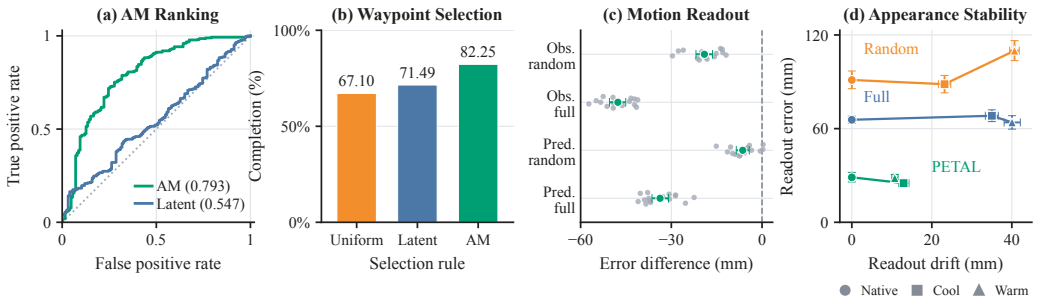

Figure 7: **Completion-aware selection and control-relevant effects.** (a) Executability ROC curves, with AUROC in the legend. (b) Fixed-budget waypoint completion. (c) PETAL-minus-reference displacement error; negative values favor PETAL, and gray dots are episode means. (d) Readout error versus recoloring-induced drift; Full denotes full-state features. Intervals are 95% bootstrap intervals. Protocols in Appendices J and K.

5.3 How do the modules contribute to closed-loop success?

Table 5 traces the framework from HWM to ExAbs. L2 denotes the high-level model retrained on AM-selected segments and deployed without AM or PETAL scoring; the remaining columns use L2 and add completion-aware waypoint selection (+AM), effect-based local control (+PETAL), or both. Retraining on AM-selected segments alone improves over HWM in 9 of 11 settings, indicating that execution-aware boundaries yield better temporal units. ExAbs improves over HWM in all 11 settings and attains the highest or tied-highest success in 9; in the other two (Reach-Wall and Counterflow), PETAL alone is best and ExAbs is within 2.1 points. The modules are complementary: AM helps most where route choice dominates (Two-room, Counterflow), PETAL where precise local motion matters (Reach-Wall, Counterflow), and their combination yields the most consistent gains.

Table 5: **Module contributions (success, %).** All augmented columns build on L2. PushT and Cube average three seeds, with Hi-LeWM online as HWM; full results in Appendix L.

Setting	HWM	L2	+AM	+PETAL	ExAbs
Maze Medium	92.5	95.0	**97.5**	92.5	**97.5**
Maze Hard	77.5	82.5	82.5	85.0	**90.0**
PushT $d = 25$	72.7	81.3	82.7	81.3	**83.3**
PushT $d = 50$	30.7	28.7	28.0	32.7	**40.7**
PushT $d = 75$	14.0	19.3	21.3	22.0	**24.0**
Cube $d = 25$	68.0	64.7	66.7	66.7	**68.7**
Cube $d = 50$	56.0	58.7	**59.3**	58.7	**59.3**
Cube $d = 75$	68.7	69.3	68.7	68.7	**72.7**
Reach-Wall	34.4	36.5	37.5	**47.9**	45.8
Counterflow	35.5	37.5	44.5	**56.5**	54.5
Two-room	7.8	9.4	15.6	15.6	**20.3**

6 Conclusion

We introduced execution-aware abstraction learning, which lets the local controller shape both the temporal units a hierarchical world model learns to request and the objective by which each request is pursued. AM grounds high-level training boundaries and waypoint selection in the controller's budget-dependent completion behavior, and PETAL expresses each request as a compact effect target. Across navigation, manipulation, and physical-robot tasks on five backbones, ExAbs improves long-range goal attainment and shortens successful executions, suggesting that hierarchical abstractions are best learned with explicit reference to the controller that executes them.

AI USE STATEMENT

We used large language models (LLMs) to polish writing, including language editing and translation; to retrieve and discover related work; and to assist research execution, such as experiment planning and code implementation. All reported results come from our own experiments. The authors reviewed and verified all AI-assisted content and take full responsibility for the scientific claims and the final manuscript.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices A–G specify the model components, training objectives, temporal conventions, and planning algorithm, including all reference hyperparameters. Appendices H and I define the evaluation tasks, protocols, and metrics, and Appendices J–L give the diagnostic protocols and complete configuration-level results. Source code implementing AM, PETAL, and hierarchical training for the Cube, Two-room, and Counterflow experiments is included in the supplementary material.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html.

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Niccoló Caselli, Francesco Massafra, Samuele Punzo, Salvatore Lo Sardo, Ippokratis Pantelidis, and Sathya Kamesh Bhethanabhotla. Mind the gap: Promises and pitfalls of hierarchical planning in LeWorldModel, 2026. URL https://arxiv.org/abs/2607.12547. arXiv:2607.12547.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Zhenyu Gao, Courtney Davis, Alyse M. Thomas, Michael N. Economo, Amada M. Abrego, Karel Svoboda, Chris I. De Zeeuw, and Nuo Li. A cortico-cerebellar loop for motor planning. *Nature*, 563(7729):113–116, 2018. doi: 10.1038/s41586-018-0633-x. URL https://doi.org/10.1038/s41586-018-0633-x.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TjCDNssXKU.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. *Reinforcement Learning Journal*, 6:1247–1272, 2025. URL https://rlj.cs.umass.edu/2025/papers/Paper136.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. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a766f56d2da42cae20b5652970ec04ef-Abstract-Conference.html.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://iclr.cc/virtual/2024/poster/18722.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lmhaVtvr.

Dinesh Jayaraman, Frederik Ebert, Alexei A. Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyzVb3CcFX.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. CompILE: Compositional imitation learning and execution. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3418–3428. PMLR, 2019. URL https://proceedings.mlr.press/v97/kipf19a.html.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312v3. arXiv:2603.19312, version 3.

R. Chris Miall, Lars O. D. Christensen, Owen Cain, and James Stanley. Disruption of state estimation in the human lateral cerebellum. *PLOS Biology*, 5(11):e316, 2007. doi: 10.1371/journal.pbio.0050316. URL https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.0050316.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.neurips.cc/paper/7591-data-efficient-hierarchical-reinforcement-learning.pdf.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gzR2VKDH.

Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/c8cc6e90ccbff44c9cee23611711cdc4-Abstract.html.

Alexander Neitz, Giambattista Parascandolo, Stefan Bauer, and Bernhard Schölkopf. Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.neurips.cc/paper_files/paper/2018/hash/0f0ee3310223fe38a989b2c818709393-Abstract.html.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34866–34891, 2023. doi: 10.52202/075280-1515. URL https://papers.neurips.cc/paper_files/paper/2023/hash/6d7c4a0727e089ed6cdd3151cbe8d8ba-Abstract-Conference.html.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39737–39761. PMLR, 2024. URL https://proceedings.mlr.press/v235/park24g.html.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ecd92623ac899357312aaa8915853699-Abstract-Conference.html.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 49126–49162, 2025. doi: 10.52202/085713-1465. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e7cf447f21cd11c846463affefce665-Abstract-Conference.html.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4732–4741. PMLR, 2018. URL https://proceedings.mlr.press/v80/srinivas18b.html.

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.

Franki Nguimatsia Tiofack, Fabian Schramm, Théotime Le Hellard, and Justin Carpentier. SVL: Goal-conditioned reinforcement learning as survival learning, 2026. URL https://arxiv.org/abs/2604.17551v2. arXiv:2604.17551, version 2.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information theoretic model predictive control: Theory and applications to autonomous driving, 2017. URL https://arxiv.org/abs/1707.02342. arXiv:1707.02342.

Daniel M. Wolpert, R. Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998. doi: 10.1016/S1364-6613(98)01221-2.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M. Patel, Paul Pu Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan Yuille, Yilun Du, and Jieneng Chen. World-In-World: World models in a closed-loop world. In *International Conference on Learning Representations*, 2026a. URL https://iclr.cc/virtual/2026/oral/10006575.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21579–21590, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5f3b8d720f34ebebceb7765e447268b-Abstract.html.

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, 2026b. URL https://arxiv.org/abs/2604.03208v2. arXiv:2604.03208, version 2.

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

A PREDICTIVE BACKBONE AND TEMPORAL CONVENTIONS

A.1 INHERITED AND LEARNED COMPONENTS

The macro-action encoder maps primitive-action chunks to $\ell_j = A_\eta(a_{t_j:t_{j+1}-1})$. For candidate plan r and primitive-action sequence $\mathbf{a}_i$, the predictors are

$$\tilde{z}_r^{(j)} = \mathcal{F}_{H,\phi}(z_t, \ell_{r,1:j}). \tag{A1}$$

$$\hat{z}_i = \mathcal{F}_L(z_t, \mathbf{a}_i). \tag{A2}$$

Observed waypoint history supplies teacher-forced inputs for the high-level prediction term

$$\mathcal{L}_H(\phi, \eta) = \mathbb{E}_{\mathcal{W}\sim\mu_\mathcal{D}}\left[\frac{1}{S}\sum_{j=0}^{S-1}\mathcal{D}_H\left(f_{H,\phi}((z_{t_m}, \ell_m)_{m=0}^{j}), \mathrm{sg}(z_{t_{j+1}})\right)\right]. \tag{A3}$$

Here $\mu_\mathcal{D}$ samples trajectory windows and sg stops target gradients. The discrepancy $\mathcal{D}_H$ follows the backbone configuration.

Following HWM (Zhang et al., 2026b), the hierarchy retains the adapted $E, \mathcal{F}_L$ and learns $A_\eta, f_{H,\phi}$ on selected windows with AM frozen. The boundaries in Section 3.2 jointly specify the action chunks and endpoint targets in Equation (A3). The high-level predictor consumes interleaved state and macro-action history, using observed waypoints during training and autoregressive predictions during planning. PETAL effects guide local control while the predictors retain their state representation.

The Maze action encoder is a Transformer with a pooled classification token and masked variable-length chunks. It uses width 64, four heads, two layers, feedforward width 128, and maximum length 32, without explicit duration context. The high-level discrepancy $\mathcal{D}_H$ in Equation (A3) is the mean squared error.

A.2 STEPS, HISTORY, AND PREDICTION INDICES

A primitive step is one action in the environment interface, with simulator substeps and action repetition converted explicitly. Duration d counts primitive steps between recorded endpoints, H counts macro-transitions, and h, k count low-level planning and execution steps. Macro-action latents do not encode an explicit duration.

For T actions, remove frame zero only from a $T+1$-frame output containing the context frame. Retain outputs with exactly T post-action frames. The Maze reference uses $h = 10$, $k = 4$, and $\tau = 10$.

The base objective C_H^{base} retains backbone-specific temporal aggregation, separately from AM terms. In Maze, the high-level goal cost is accumulated over the predicted waypoints.

B EXECUTION LABELS AND AM

B.1 COMPLETION IS A MEASURED EVENT

Each record binds a source initialization, fixed waypoint, completion rule, and low-level runtime. Initialization includes physical state, planner history, nominal actions, and the random-number convention. The conditional completion distribution averages over the initialization and execution randomness of this collection protocol. Physical measurements used to assign labels remain separate from the planner's observation inputs.

Maze supports physical-waypoint labels with observed targets and physical completion rules, and imagined-latent labels comparing executed observations with a supplied latent waypoint. The latter thresholds the raw effect discrepancy and the fixed-support endpoint error,

$$\mathcal{C}_\xi^{\mathrm{latent}}(o_b, w) = \mathbf{1}\left\{D_{\mathrm{raw}}\left(e_0^*, e_b; h\right) \le \theta_e \ \wedge\ d_{\kappa_0^*}\left(E(o_b), w\right) \le \theta_A\right\}. \tag{B1}$$

The source, target effect e_0^*, and support κ_0^* remain fixed throughout labeling. Initially completed queries are excluded, and completion is checked after each action for up to 32 steps. Lower-budget

views reuse the same attempt. In the observed-waypoint diagnosis of Appendix J, the frozen host controller completes a request at planar distance at most 0.5 or positive environment reward. That diagnosis uses raw CDF scores at budgets 8, 10, 16, 24, and 32. Task-level goal attainment is measured by the environment's task rule (Appendix H).

B.2 Hazard estimation and censored supervision

AM averages head logits $g_{m,b}(z, w)$ before computing its aggregate CDF, while disagreement uses individual head CDFs,

$$\begin{aligned} F_m(B) &= 1 - \prod_{b=1}^{B} \big(1 - \sigma(g_{m,b})\big), \\ \bar{g}_b &= \frac{1}{M} \sum_{m=1}^{M} g_{m,b}, \qquad \bar{F}(B) = 1 - \prod_{b=1}^{B} \big(1 - \sigma(\bar{g}_b)\big). \end{aligned} \tag{B2}$$

These raw CDFs are nondecreasing, starting at zero. Let δ_q indicate a first hit and $\bar{T}_q$ its time, or $B_{\max}$ for an unsuccessful valid attempt. With $f_q(b) = \bar{F}_q(b) - \bar{F}_q(b-1)$ and $N_{\text{risk}} = \sum_q \bar{T}_q$, the event form and its equivalent stepwise form are

$$\begin{aligned} \mathcal{L}_{\text{hit}} &= -\frac{1}{N_{\text{risk}}} \sum_q \big[\delta_q \log f_q(\bar{T}_q) + (1 - \delta_q) \log\big(1 - \bar{F}_q(\bar{T}_q)\big)\big] \\ &= \frac{\sum_{q,b} m_{qb}\, \ell_{\text{BCE}}\big(\sigma(\bar{g}_{qb}), y_{qb}\big)}{\max(1, \sum_{q,b} m_{qb})}, \qquad y_{qb} = \mathbf{1}\{\delta_q = 1,\ b = \bar{T}_q\}. \end{aligned} \tag{B3}$$

The risk mask m_{qb} includes valid steps through the first hit or censoring limit, and unsuccessful valid attempts enter as right-censored records observed through $B_{\max}$. Observation after every primitive action yields first-hit labels; observations at control-cycle boundaries yield interval labels.

B.3 Calibration and the two score maps

Held-out calibration fits $c_B(p) = \sigma(a_B \operatorname{logit}(\operatorname{clip}_{[\epsilon,1-\epsilon]} p) + b_B)$ with $a_B > 0$ for each supported budget. Segmentation uses $u^{\text{seg}}(B) = \text{Std}_m[c_B(F_m(B))]$, whereas planning uses $u^{\text{plan}}(B) = \text{Std}_m[F_m(B)]$, both with population standard deviations. Their corrected scores are

$$\begin{aligned} q^{\text{seg}}(B) &= \operatorname{clip}_{[\epsilon,1]}\big(c_B(\bar{F}(B)) - \beta_{\text{seg}} u^{\text{seg}}(B)\big), \\ \rho^{\text{plan}}(B) &= \operatorname{clip}_{[0,1]}\big(c_B(\bar{F}(B)) - \beta_{\text{plan}} u^{\text{plan}}(B)\big). \end{aligned} \tag{B4}$$

Online log risk uses $q^{\text{plan}}(B) = \max(\rho^{\text{plan}}(B), \epsilon)$; the threshold penalty uses unfloored ρ^{plan}. The reference has $\epsilon = 10^{-6}$ and threshold-denominator floor $\epsilon_{\text{bar}} = 10^{-3}$. The raw CDFs in Equation (B2) are monotone in the budget, calibration is fitted per budget, and the disagreement-corrected values serve as ranking scores.

The boundary builder also applies $-\lambda_u u^{\text{seg}}$ outside the logarithm in Equation (2). Its coefficient is separate from β_{seg}, which corrects the score before logarithmic transformation.

Threshold selection takes the largest descending-score prefix meeting the minimum count and one-sided Wilson failure-bound criterion on the calibration population, and we record the accepted count and failure rate for each budget. Calibration is tied to the executor and query distribution that produced these records.

B.4 Auxiliary AM objectives

The auxiliary losses available for AM training are listed below; each checkpoint enables a subset of them.

Term	Definition
Integrated Brier	Mean squared error between $\bar{F}_q(b)$ and $\mathbf{1}\{1 \le T_q \le b\}$ over valid records and observed budgets
Progress	Smooth-L1 regression of predicted progress to the recorded normalized progress target
Terminal residual	Smooth-L1 regression to the recorded log-residual target
Realized value	Smooth-L1 regression to $v_q = \mathbf{1}\{1 \le T_q \le d_q\} v_q^{\text{target}}$
Listwise value	Cross-entropy from $\mathrm{softmax}(v/\lambda_v)$ to $\mathrm{softmax}(\hat{v})$ within a valid source group
Relative value	Smooth-L1 regression of $\hat{v}_i - \hat{v}_{i_0}$ to $v_i - v_{i_0}$, where i_0 is the group's maximum-gap candidate
Same-gap success ranking	For success p and failure n in one gap stratum, average $\mathrm{softplus}(-(l_p - l_n)/\lambda_s)$ with $l = \mathrm{logit}(\mathrm{clip}\,\bar{F}(d))$
Same-gap time ranking	Pairwise logistic loss over successful attempts with unequal hit times, preferring shorter predicted truncated hit time for earlier hits
Member supervision	Mean member survival loss plus member realized-value regression under the deterministic group masks

Each head is supervised on a fixed, head-specific two-thirds of the source groups, which defines the deterministic group masks. Truncated time is $\mathbb{E}[\min(T, B+1)] = 1 + \sum_{b=1}^{B}(1 - \bar{F}(b))$. Pair losses average within gap strata, then across available strata; empty sets contribute zero.

C Boundary optimization and training windows

Maze windows end at demonstrated terminal frame T, with starts from $\max(0, T - Sd_{\max})$ to $T - Sd_{\min}$. A seeded permutation visits starts without replacement, repeating cycles as needed. The Maze configuration samples 40 windows per episode with six macro-transitions each, so boundaries are selected per window.

Let $V_s(j)$ be the maximum score reaching j with s segments. Initialize $V_0(t_0) = 0$ and other entries to negative infinity. The implementation also supports optional seeded edge perturbations ζ_{ij},

$$V_s(j) = \max_{d_{\min} \le d \le d_{\max}} \left[V_{s-1}(j-d) + s_{j-d,j} + \zeta_{j-d,j} \right], \qquad s = 1, \dots, S. \tag{C1}$$

Mask invalid predecessors and positions beyond the endpoint, then backtrack from $V_S(t_S)$. Zero perturbation recovers Equation (2). Ascending duration order and strictly improving predecessor updates resolve ties.

With W visited positions and $D = d_{\max} - d_{\min} + 1$ durations, recurrence cost is $O(SWD)$ after edge scoring. Edge scores are cached and shared across overlapping windows.

Selected chunks use duration-aware padding masks and observed endpoint targets. Reference durations include every integer from 10 to 32. These intervals determine the action chunks and endpoint targets used to retrain the macro-action encoder and high-level predictor.

D PETAL representation, runtime, and supervision

D.1 Spatial context and encoder

The separately pretrained action-grounded selector K uses spatial features $z, w \in \mathbb{R}^{C \times H_z \times W_z}$ and $\Delta = w - z$. With learned map S_χ and change energy $E_u = C^{-1} \sum_c \Delta_{c,u}^2$, it selects

$$s_u^K = S_\chi([z, w, \Delta])_u + \log\left(\frac{E_u}{\max(\max_v E_v, \epsilon_K)} + \epsilon_K \right),$$
$$K_u(z, w) = \mathbf{1}\{u \in \mathrm{TopK}(s^K, n_K)\},$$

Here $n_K = \max(1, \min(H_z W_z, \lceil \alpha_K H_z W_z \rceil))$, with retained fraction $\alpha_K = 0.2$ and $\epsilon_K = 10^{-6}$ in the reference. The selector uses 16 input feature channels, hidden width 32, two 3×3 GroupNorm–

SiLU blocks, and a scalar-output 1×1 convolution. Frozen during PETAL training and deployment, it supplies shared source-target support for all candidates.

Selector pretraining predicts ten-action prefixes from support-pooled source, target, and difference features plus observed gap divided by 32. A two-hidden-layer width-128 decoder uses action MSE. A straight-through mask uses a sigmoid surrogate around the detached top-k threshold at temperature 0.25. Deployment retains only the hard-mask selector, without decoder, labels, or gap input.

The reference training set contains 1,527 pairs, with 549 validation pairs and gaps of at least 20. Training uses 50 epochs, batch size 128, AdamW learning rate 3×10^{-4}, weight decay 10^{-4}, and gradient clipping at 5. Coverage cycles over $\{0.1, 0.2, 0.3, 0.4, 0.5\}$; mean validation action MSE selects the checkpoint.

Support κ^* defines a centroid translation on the normalized feature grid. Shared bilinear resampling centers source, destination, and support with zero padding and aligned corners. Normalize support by its sum for pooling and maximum for feature masking; absolute reference coordinates remain outside the encoder.

The encoder concatenates masked source, destination, difference, and support channels, followed by two 3×3 GroupNorm–SiLU blocks. Support-weighted pooling, four relative support statistics, and a Fourier duration embedding feed a two-layer LayerNorm–GELU projection. Maze uses effect dimension 96, hidden width 128, duration embedding dimension 32, and comparison-network width 256.

In the main text, $\Phi_\psi(z, y; \kappa^*) = P_\psi(\Gamma(z, y; \kappa^*), \tau_0)$ abbreviates this shared preprocessing Γ and fixed-scale effect encoding, without an additional network or adapter.

D.2 Discrepancy, anchoring, and shortlist

The comparison network maps target effect and candidate horizon to raw outputs r_m, r_γ, defining positive dimension weights and scale,

$$\begin{aligned} v_m &= \frac{\operatorname{softplus}(r_m)}{\max\left(d_e^{-1} \sum_n \operatorname{softplus}(r_n), \epsilon_D\right)}, \\ \gamma &= \operatorname{softplus}(r_\gamma) + \gamma_{\min}, \qquad D_i = \frac{\gamma}{d_e} \sum_m v_m (e^*_m - \hat{e}_{i,m})^2. \end{aligned} \tag{D1}$$

For support weights κ^*_u, the anchor averages squared endpoint errors over feature channels and the target support. Define active-set population means μ_D, μ_A and standard deviations s_D, s_A. Scale matching is

$$\begin{aligned} A_i &= \frac{\sum_u \kappa^*_u C^{-1} \sum_c (\hat{z}_{i,c,u} - w_{c,u})^2}{\max(\sum_u \kappa^*_u, \epsilon_D)}, \\ \widetilde{D}_i &= \mu_A + \max(s_A, \epsilon_D) \operatorname{clip}_{[-c,c]} \left(\frac{D_i - \mu_D}{\max(s_D, \epsilon_D)} \right). \end{aligned} \tag{D2}$$

Maze applies Equation (D2) to the lowest-anchor 128 candidates, separately per world-model member before inherited ensemble reduction. Other candidates retain anchors. Blending precedes clamping and addition of retained planner terms. Reference settings are $1 - \omega = 0.9$, $c = 2$, $\epsilon_D = 10^{-6}$, and $\gamma_{\min} = 0.05$.

The learned scale γ enters the raw discrepancy used for latent completion labels (Equation (B1)); in the planning cost, within-query standardization normalizes it away, so runtime weighting is set by ω and the anchor scale. Setting $\omega = 0$ recovers the support anchor alone.

D.3 Teacher records and the training cost

The collector takes frozen observation pairs and records the first ten-step low-level query without environment execution or the final AM-trained high-level model. Teacher goal cost sums full-feature errors over predicted steps and averages world-model members,

$$G_i^T = \sum_{b=1}^{h} \frac{1}{M_L} \sum_{m=1}^{M_L} \mathrm{MSE}(\hat{z}_{i,b}^{(m)}, w), \qquad R_i^T = J_i^T - G_i^T, \qquad J_i^{\psi,\mathrm{train}} = G_i^{\psi,\mathcal{V}} + R_i^T. \quad \text{(D3)}$$

Here J_i^T is the recorded total teacher cost. The student retains residual R_i^T, including action-perturbation correction and other active terms. Its goal score $G_i^{\psi,\mathcal{V}}$ uses the anchored blend and clamp over the entire retained pool $\mathcal{V}$, without the runtime shortlist. Thus Equation (5) supervises decisions on the training pool, whereas Equation (4) uses the deployment shortlist. Training retains 500 candidates per query; deployment refines 128.

Teacher records retain ensemble-averaged trajectory costs alongside the endpoint features used by PETAL. Candidate subsampling renormalizes teacher probabilities and recomputes updates on the retained pool. The objective therefore distills the recorded candidate decisions into an endpoint-based comparison.

For nominal sequence $\bar{\mathbf{a}}$, student probabilities and the full sequence update are

$$p_i^{\psi} = \frac{\exp(-J_i^{\psi,\mathrm{train}}/\lambda_p)}{\sum_{j\in\mathcal{V}} \exp(-J_j^{\psi,\mathrm{train}}/\lambda_p)}, \qquad \Delta\mathbf{a}^{\psi} = \sum_{i\in\mathcal{V}} p_i^{\psi}(\mathbf{a}_i - \bar{\mathbf{a}}).$$

Recorded teacher probabilities p^T define $\Delta\mathbf{a}^T$ on the same pool. Equation (5) compares full $h \times d_a$ updates, not only the first action.

D.4 Spatial MPPI distillation loss

Let $\mathcal{E}$ contain the lowest-cost teacher elites. Standardization $Z_{\mathcal{V}}(x)$ subtracts the valid-pool mean and divides by its population standard deviation with a positive floor. The auxiliary losses complementing Equation (5) are

$$\begin{aligned}
\mathcal{L}_{\mathrm{rank}} &= \operatorname*{mean}_{i\in\mathcal{E}, j\in\mathcal{V}\setminus\mathcal{E}} \mathrm{softplus}(J_i^{\psi,\mathrm{train}} - J_j^{\psi,\mathrm{train}}), \\
\mathcal{L}_{\mathrm{elite}} &= D_{\mathrm{KL}}\left(\mathrm{Unif}(\mathcal{E}) \,\middle\|\, \mathrm{softmax}\big(-Z_{\mathcal{V}}(J^{\psi,\mathrm{train}})/\lambda_e\big)\right), \\
\mathcal{L}_{\mathrm{cost}} &= \mathrm{mean}_{i\in\mathcal{V}} \left[Z_{\mathcal{V}}(G^{\psi,\mathcal{V}})_i - Z_{\mathcal{V}}(C^{\mathrm{end}})_i\right]^2, \\
C_i^{\mathrm{end}} &= \mathrm{MSE}(\hat{z}_i, w).
\end{aligned} \quad \text{(D4)}$$

Cost matching uses terminal errors, not path-summed G^T. Elite counts round the configured fraction, retaining at least one candidate; empty ranking-pair sets contribute zero. Cost standardization has a separate numerical floor from runtime scale matching.

For the valid target and candidate effects in a minibatch, let V be their sample covariance and d_e their dimension. The representation terms follow variance and covariance regularization adapted from Bardes et al. (2022),

$$\mathcal{L}_{\mathrm{var}} = \frac{1}{d_e} \sum_m [1 - \sqrt{V_{mm} + \epsilon_v}]_+, \qquad \mathcal{L}_{\mathrm{cov}} = \frac{1}{d_e(d_e-1)} \sum_{m\neq n} V_{mn}^2. \quad \text{(D5)}$$

The covariance uses the unbiased sample denominator. The spatial reference configuration uses KL and update weights 1, ranking weight 0.25, elite-distribution weight 1, and terminal-cost weight 0.1. Variance and covariance weights are 0.1 and 0.01, with $\epsilon_v = 10^{-4}$. The elite fraction is 0.1 and its standardized-cost temperature is 1. The MPPI distillation temperature is 0.0025.

The equations above describe the spatial MPPI implementation. CEM-based planners supervise PETAL through elite preferences and their induced candidate-distribution updates, as described for the robot configuration in Section 4.4.

E EXECUTOR MATCHING AND BUDGET SEMANTICS

E.1 EXECUTOR DEFINITION

An executor is identified by its encoder, predictor, PETAL, selector, action normalization, optimizer, candidate counts, precision, local objective, h, k, action repetition, initialization, warm starts, and random-number scope. AM data are further tied to the waypoint representation and completion predicate. Any change to action selection is therefore matched by corresponding AM supervision.

Related queries and repeated attempts remain grouped. Model selection and calibration use preparation groups, with final evaluation groups reserved.

Appendix D defines the spatial MPPI executor, and Appendix B defines its completion rules and score maps. The diagnostic AM in Appendix J is trained for the frozen full-latent controller without PETAL, which isolates AM's predictive quality from the effect-based objective. For Cube, the +AM and ExAbs columns use AM trained with the full-latent and PETAL-enabled executors, respectively (Appendix L). In the physical-robot configuration, the frozen PETAL-enabled controller supplies AM labels for boundary selection and planning. The three robot tasks share these learned components (Appendix H.1). Task completion rules and special control phases are specified in Appendix H; budget roles are distinguished below.

E.2 REFERENCE BUDGET AND LOCAL CONTROL

Boundary selection queries AM at the observed duration of each candidate training segment. Online selection instead scores first-waypoint completion at the fixed reference budget B_0. The local prediction horizon h, execution prefix k, and comparison scale τ_0 remain controller settings. Thus variable training boundaries and a fixed online comparison budget serve different roles within the same hierarchy.

F DOMAIN-SPECIFIC REALIZATION

Task-specific models share the roles of temporal abstraction, effect comparison, and executor-matched supervision, with adapters that preserve each backbone's observation and action interfaces. The PLDM spatial reference uses target-conditioned support and MPPI weight and action-update supervision (Appendix D). The physical-robot realization uses V-JEPA 2-AC with visual and proprioceptive inputs. Its CEM planner supervises PETAL through elite selection and candidate-distribution updates, with AM labels collected under the frozen PETAL-enabled controller. Parameters are not shared across backbone families. Appendix H identifies the remaining task backbones and completion rules, and Appendix I defines task success and interaction steps.

G ONLINE PLANNING ALGORITHM

All learned parameters remain fixed; only candidate sequences and optimizer statistics change within each control round.

Algorithm G1: Closed-loop planning with AM and PETAL.

Inputs. Trained modules, current and goal observations, causal history, H, h, k, and fixed high-level and low-level search configurations.

Control loop. Repeat until environment termination or exhaustion of the task execution budget.

1 Encode the current and goal observations, retaining causal history; initialize or warm-start high-level search.

2 At each high-level iteration, predict candidate macro-rollouts, score their first waypoints with AM, and update the plan using Equation (6).

3 Take the returned plan's first predicted state as w; compute shared support $\kappa^* = K(z_t, w)$ and effect target using Equation (3).

4 Initialize or warm-start low-level search. At each iteration, predict primitive-action outcomes, compute endpoint effects and anchors, and update actions using Equation (4).

5 Execute the returned sequence's first k actions, stopping on environment termination; update observations and history, then replan at both levels.

Both searches use their configured iterations and return optimized action sequences. AM's budget, local horizon h, and execution prefix k follow the definitions in Appendix E. PETAL's support and target effect stay fixed across candidates and are recomputed for each new waypoint. Deployment replans both levels after feedback while retaining the labeling executor's local action-selection configuration. Task-specific direct-goal phases bypass high-level waypoint selection, including Reach-Wall's final 15 environment actions (Appendix H.1).

H Evaluation protocols

H.1 Tasks and predictive backbones

The simulated comparisons use seven tasks and four backbone families. PLDM supplies the low-level model for Diverse Maze and separately adapted models for Two-room and Counterflow Navigation. LeWM supplies the PushT and Cube backbones. Reach-Wall uses JEPA-WMs, and Button–Window uses DINO-WM. Each simulated setting uses task-specific AM, PETAL, and high-level parameters.

Diverse Maze. Each method is evaluated once on each of 40 queries per difficulty group, giving 120 trials per method. Difficulty follows HWM's shortest grid-path distance D, with easy, medium, and hard groups defined by $D \in [5, 8]$, $[9, 12]$, and $[13, 16]$, respectively (Zhang et al., 2026b). All methods share the same queries, success criterion, and execution budget. Every baseline, including HWM and the offline goal-conditioned methods, is rerun from its public implementation on our offline dataset and query set, so all comparisons in Table 1 share one protocol; absolute rates therefore differ from those reported by Zhang et al. (2026b). Both hierarchies use a ten-action local horizon and executes four actions before replanning. Table 1 reports success separately for the three groups, and Table S2 lists the underlying counts.

PushT and Cube. Source-trajectory separations $d \in \{25, 50, 75\}$ define start–goal queries. Cube uses trajectory-goal queries rather than the standard OGBench goal distribution (Park et al., 2025). Success is reported as mean $\pm$ sample standard deviation over three seeds, separately for each query separation.

Reach-Wall. All configurations use the same 96 queries and a 100-step budget. Success in Tables 2 and 5 means reaching the goal at least once within budget. Flat, HWM, and ExAbs reach the goal on 31, 33, and 44 queries, giving 32.29%, 34.38%, and 45.83%, respectively. For reference, terminal success after the full budget is 21/96 (21.88%), 28/96 (29.17%), and 35/96 (36.46%), respectively. The deployment policy switches to direct-goal low-level control for the final 15 environment actions.

Two-room. In the original layout, the upper passage permits only left-to-right motion, opposite to the evaluated crossing, whereas the lower passage is bidirectional. Evaluation uses original and mirrored layouts, 64 test episodes, and a 300-step budget.

Counterflow Navigation. The agent starts below a goal in the upper region. An adverse-flow region opposes direct upward travel, while narrow side corridors permit alternative routes. Evaluation comprises 200 tasks with a 250-step budget. Success rates are 26.00%, 35.50%, and 54.50% for the flat planner, HWM, and ExAbs, respectively.

Button–Window. The task requires pressing a button to unlock a window before moving it to the target position. Each method runs once on each of 60 distinct start–goal configurations, giving 180 episodes across the three methods. Initial states, goal images, target window positions, and the 300-action budget are shared across methods. Success requires an active button-unlock state and a window-position error of at most 3 cm, maintained simultaneously for five consecutive action steps. Execution stops when this condition is first completed, and the fifth qualifying step defines the completion time. Trials that do not complete this condition within 300 actions are failures. Flat DINO-WM, HWM, and ExAbs succeed on 0, 0, and 13 of 60 trials, respectively (Table 2).

Physical robot. An AgileX PiPER six-degree-of-freedom arm receives a fixed-view RGB image, end-effector pose, and gripper state. The deployment goal comprises an RGB image and its end-effector state. Actions specify end-effector pose increments, converted to joint commands by inverse kinematics, and a gripper command, matching the end-effector action interface of V-JEPA 2-AC (Assran et al., 2025). Each task uses ten start–goal configurations and five repetitions per method. Configurations are excluded from training, AM label collection, and tuning; the physical scene is reset and methods are interleaved. Timeouts, safety stops, and human interventions count as failures.

Basket transfer succeeds when the released object lies entirely inside the basket. Drawer loading succeeds when the released object lies inside the opened drawer; closing is not required. Raised transfer succeeds when the object passes over the obstacle without contact and is released with its center within 3 cm of the planar target. The initial object position and the target lie on opposite sides of the obstacle. Camera and scene fixtures remain fixed while object, target, and arm starts vary. All three tasks share the adapted low-level predictor and, for ExAbs, the learned abstraction components and high-level model. Parameters are frozen during evaluation. Model-predicted subgoals and deployment goal images are shown in Appendix M.2.

H.2 Baseline identities

GCIQL denotes goal-conditioned IQL; HIQL is a hierarchical goal-conditioned method, and HILP learns goal-conditioned policies from Hilbert representations. HWM denotes the PLDM hierarchy in Maze. ExAbs denotes our complete method, which integrates the AM and PETAL modules into the hierarchy.

In PushT and Cube, the HWM reference is Hi-LeWM (Caselli et al., 2026), which adds an HWM-style high-level macro-action predictor to a frozen LeWM low-level planner. We evaluate the released Hi-LeWM checkpoints and planning configurations (Zenodo record 21353240, https://doi.org/10.5281/zenodo.21353240) without modification under our seeds and query sets. The four configurations in Tables 3 and S1, Hi-LeWM and Hi-LeWM-C under online and staged execution, are defined by Caselli et al. (2026). Hi-LeWM uses standard high-level CEM, whereas Hi-LeWM-C constrains the search around macro-actions encoded from training trajectories. Online replans the high level periodically, whereas staged commits to a sequence of high-level targets computed at the start of the episode; neither updates parameters during deployment. Hi-LeWM thus provides the HWM hierarchy on the same LeWM backbone as ExAbs, and all four configurations are evaluated on the same query sets and protocol as ExAbs. The physical-robot comparison shares one adapted V-JEPA 2-AC predictor. Its HWM-style reference uses randomly selected valid internal training boundaries, whereas ExAbs uses AM-selected boundaries.

Flat in Table 2 is DINO-WM for Button–Window, JEPA-WMs for Reach-Wall, and separately adapted PLDM models for the navigation tasks. Results are reported per task.

Table S1: **Complete manipulation success (%).** Mean ± sample SD over three seeds at all source-trajectory separations. Hi-LeWM and Hi-LeWM-C follow Caselli et al. (2026) (Appendix H.2). Bold marks the highest mean within each task and separation, including ties.

Method	$d = 25$	$d = 50$	$d = 75$
PushT			
Flat LeWM	82.0 ± 3.5	38.7 ± 5.0	19.3 ± 4.6
Hi-LeWM online	72.7 ± 1.2	30.7 ± 5.2	14.0 ± 3.5
Hi-LeWM-C online	74.7 ± 3.1	39.3 ± 4.2	16.7 ± 6.4
Hi-LeWM staged	82.0 ± 6.0	38.7 ± 6.1	13.3 ± 6.4
Hi-LeWM-C staged	64.0 ± 5.3	$\mathbf{44.7} \pm 6.4$	10.0 ± 2.0
ExAbs	$\mathbf{83.3} \pm 3.1$	40.7 ± 2.5	$\mathbf{24.0} \pm 3.5$
Cube			
Flat LeWM	$\mathbf{71.3} \pm 4.2$	52.0 ± 5.3	61.3 ± 9.0
Hi-LeWM online	68.0 ± 3.5	56.0 ± 3.5	68.7 ± 8.3
Hi-LeWM-C online	68.0 ± 6.9	$\mathbf{59.3} \pm 3.1$	70.0 ± 11.1
Hi-LeWM staged	67.3 ± 6.4	53.3 ± 5.0	52.7 ± 7.6
Hi-LeWM-C staged	62.0 ± 4.0	57.3 ± 8.1	48.7 ± 3.1
ExAbs	68.7 ± 3.1	$\mathbf{59.3} \pm 6.4$	$\mathbf{72.7} \pm 4.6$

H.3 Training stages

After backbone adaptation, the encoder and low-level predictor remain fixed while the abstraction components and high-level dynamics are learned. AM additionally uses preparation-time execution outcomes, separate from predictive-training trajectories and evaluation episodes. In the robot experiment, these outcomes are collected with the frozen PETAL-enabled controller. Baselines keep their own search configurations (the released ones for Hi-LeWM), and execution budgets, counted in environment actions, are shared within each task.

I Completion and action metrics

I.1 Success and repetitions

Success is the percentage of scheduled trials satisfying the task's completion rule. AM labels describe fixed-waypoint completion, while task-level success concerns the final requested goal. Reach-Wall counts a trial as successful once the goal is reached within budget. Terminal success after the full budget is recorded separately for reference in Appendix H.1.

PushT and Cube report means and sample standard deviations over three seeds separately for each query separation. Other tasks report observed rates on fixed query sets, and Table S2 gives the underlying success counts. Main-text tables and three-seed appendix tables round rates to one decimal; fixed-set appendix tables report two decimals.

Table S2: **Success counts on fixed query sets.** Entries are successes/trials with the rate (%) in parentheses.

Maze method	Easy	Medium	Hard
GCIQL	31/40 (77.50)	14/40 (35.00)	9/40 (22.50)
HIQL	34/40 (85.00)	25/40 (62.50)	17/40 (42.50)
HILP	16/40 (40.00)	5/40 (12.50)	3/40 (7.50)
PLDM	36/40 (90.00)	26/40 (65.00)	23/40 (57.50)
HWM	40/40 (100.00)	37/40 (92.50)	31/40 (77.50)
ExAbs	40/40 (100.00)	39/40 (97.50)	36/40 (90.00)
Task	**Flat**	**HWM**	**ExAbs**
Button–Window	0/60 (0.00)	0/60 (0.00)	13/60 (21.67)
Reach-Wall	31/96 (32.29)	33/96 (34.38)	44/96 (45.83)
Two-room	0/64 (0.00)	5/64 (7.81)	13/64 (20.31)
Counterflow	52/200 (26.00)	71/200 (35.50)	109/200 (54.50)
Basket transfer	2/50 (4.00)	10/50 (20.00)	14/50 (28.00)
Drawer loading	0/50 (0.00)	5/50 (10.00)	8/50 (16.00)
Raised transfer	0/50 (0.00)	12/50 (24.00)	15/50 (30.00)

I.2 Successful-trial step means

Step means average environment actions over each method's own successful trials, giving the conditional execution length given success. All scheduled trials contribute to task success rates.

Table S3: **Successful-trial step means.** Values plotted in Figure S1; each entry averages the corresponding method's own successful trials. HWM is the PLDM hierarchy in Maze and Hi-LeWM online in PushT and Cube.

Task and group	Flat	HWM	ExAbs
Maze Easy	405.56	103.43	86.79
Maze Medium	462.65	166.30	135.95
Maze Hard	482.00	244.19	219.89
PushT	24.32	31.13	20.38
Cube	6.99	12.05	6.09

Figure S1 divides each mean by the HWM mean in the same setting, which displays relative differences while keeping each task on its own step scale. Flat denotes PLDM for Maze and LeWM for the manipulation tasks. ExAbs uses fewer steps than both references in all five settings, including Maze Easy, where ExAbs and HWM succeed on the same 40 queries.

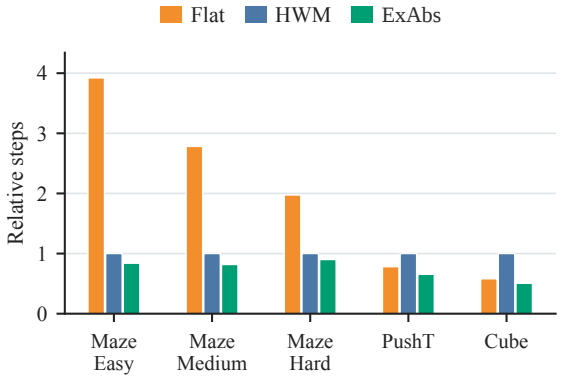

Figure S1: **Successful-execution steps.** Mean environment steps over each method's own successful trials, normalized by the HWM mean in the same setting. Raw values are in Table S3.

I.3 Figure and table conventions

Step figures display the conditional means defined above. Table bolding marks the largest observed mean, including ties. Qualitative comparisons use matched starts and goals and normalized display time. The waypoint columns contain representative predictions, with timing conventions specified in Appendix M. Both successful and unsuccessful executions are included.

J AM execution diagnostics

J.1 Fixed-waypoint protocol and probability readouts

Figure 7a–b evaluates one frozen AM checkpoint, trained for the full-latent controller described below, on 400 observed source–waypoint transitions across six Maze layouts that are disjoint from the 25 layouts used to train AM. Candidates were observed forward transitions with recorded trajectory gaps of 8–32 actions, selected independently of execution outcomes and AM scores. Source states already satisfying the completion criterion were excluded. The six layouts contributed 67, 67, 67, 67, 66, and 66 attempts.

The frozen full-latent baseline controller, without PETAL, held each requested waypoint fixed while replanning from feedback. Its MPPI planner used a ten-action horizon, 500 candidates, one refinement, temperature 0.0025, and noise scale 5. Actions were replanned after every four environment control actions. Attempts were observed for at most 32 actions. Completion was the first post-action state with planar distance at most 0.5 from the waypoint or positive environment reward. The prespecified primary budget was ten actions, with 274 completed and 126 uncompleted requests.

AM supplied raw budget-dependent probabilities, without the held-out calibration used during planning. The distance comparator used negative latent endpoint mean-squared error as a ranking score. The constant prior was estimated independently from 18,473 validation records across four equally weighted layouts. Its ten-action probability was 0.642. Brier scores compare AM with the prior, while AUROC compares AM with latent distance (Table S4).

Overall statistics use an equal-layout mixture, with each query weighted inversely by its layout's query count. Uncertainty uses 2,000 percentile bootstrap resamples of the six layout clusters, preserving method pairing and shared attempts across budgets. A probability-bin analysis with bins of width 0.2 shows that the raw scores are well ordered but compressed: queries with mean predicted probability 0.538 complete at a rate of 0.270, whereas those with mean prediction 0.672 complete at a rate of 0.796. Planning therefore uses the per-budget calibrated scores of Appendix B.3.

Table S4: **Overall AM diagnosis at ten actions.** Estimates weight layouts equally. Brackets are 95% layout-cluster bootstrap intervals. A dash denotes an inapplicable readout.

Scoring rule	AUROC ↑	Brier ↓
AM	0.793 [0.725, 0.860]	0.191 [0.181, 0.200]
Latent distance	0.547 [0.508, 0.592]	—
Validation prior	—	0.218 [0.208, 0.230]

The paired AM-minus-distance AUROC difference is 0.246 [0.153, 0.326]; the prior-minus-AM Brier reduction is 0.027 [0.019, 0.037].

J.2 Within-source candidate comparisons

The same 400 attempts formed 153 source groups. Each group shared its layout, source episode, timestep, and restored physical source state. The candidate analysis uses all 111 multi-candidate groups for selected completion and their 46 mixed-outcome groups for concordance; the 42 singleton groups do not admit within-source selection. Groups with identical outcomes remain in selection estimates but do not define success–failure rankings.

Concordance is the fraction of completed–uncompleted candidate pairs correctly ordered within a group, with half credit for tied scores. Selected completion is the observed completion indicator of the highest-scoring candidate, averaged uniformly over ties. Group summaries are averaged equally within each layout and then across layouts. The five scoring rules use the same eligible groups and paired layout resamples. Uniform selection averages the completion indicators of all candidates within each group, giving its expected completion rate. Figure 7b displays this reference alongside latent-distance and AM selection. Physical distance and shorter recorded trajectory gap use information unavailable to the visual planner; Table S5 reports both controls with their intervals.

Table S5: **Within-source diagnosis.** Concordance uses 46 mixed-outcome groups; selected completion uses all 111 multi-candidate groups. Brackets are 95% percentile intervals from 2,000 paired layout-cluster bootstrap resamples.

Scoring rule	Concordance ↑	Selected completion (%) ↑
Uniform	0.500 [0.500, 0.500]	67.10 [62.05, 71.73]
Latent distance	0.689 [0.596, 0.796]	71.49 [67.84, 75.96]
AM	0.863 [0.768, 0.953]	82.25 [77.32, 87.10]
Physical distance[†]	0.988 [0.970, 1.000]	86.82 [82.60, 90.70]
Shorter trajectory gap[†]	0.992 [0.977, 1.000]	86.82 [82.60, 90.70]

[†]Privileged diagnostic controls. The paired AM-minus-latent differences are 0.174 [0.030, 0.311] in concordance and 10.77 [4.60, 17.25] percentage points in selected completion, computed before rounding. Selected completion is evaluated retrospectively from the recorded fixed-waypoint attempts.

AM ranks completion more accurately and selects more frequently completed waypoints than the backbone's latent matching rule, closing most of the gap between latent distance and the privileged references using only the planner's latent inputs.

J.3 Budget-dependent prediction

Table S6 reports all evaluated budgets from the same 400 attempts. AM has higher AUROC than latent distance and lower Brier error than the validation prior at every evaluated budget, with the largest AUROC gains (0.30–0.37) at budgets of 16–32 actions. Each budget specifies a completion event and queried CDF value, while the local horizon and execution procedure remain fixed. Budget-specific priors and completion counts accompany the scores to show changes in completion prevalence.

Figure S2 complements the marginal intervals with paired differences across budgets.

Table S6: **Budget-dependent AM diagnosis.** All rows use the same 400 attempts and equal-layout weighting. Parenthetical completion counts are unweighted. Brackets denote 95% layout-cluster bootstrap intervals.

Budget (completed)	AM AUROC	Latent distance AUROC	AM Brier	Prior Brier
8 (215)	0.733 [0.658, 0.811]	0.526 [0.485, 0.562]	0.240 [0.233, 0.246]	0.251 [0.250, 0.251]
10 (274)	0.793 [0.725, 0.860]	0.547 [0.508, 0.592]	0.191 [0.181, 0.200]	0.218 [0.208, 0.230]
16 (339)	0.871 [0.807, 0.926]	0.506 [0.452, 0.557]	0.109 [0.092, 0.123]	0.130 [0.110, 0.150]
24 (364)	0.908 [0.874, 0.937]	0.570 [0.495, 0.626]	0.069 [0.051, 0.090]	0.083 [0.062, 0.110]
32 (372)	0.879 [0.819, 0.925]	0.580 [0.437, 0.662]	0.056 [0.034, 0.081]	0.067 [0.043, 0.097]

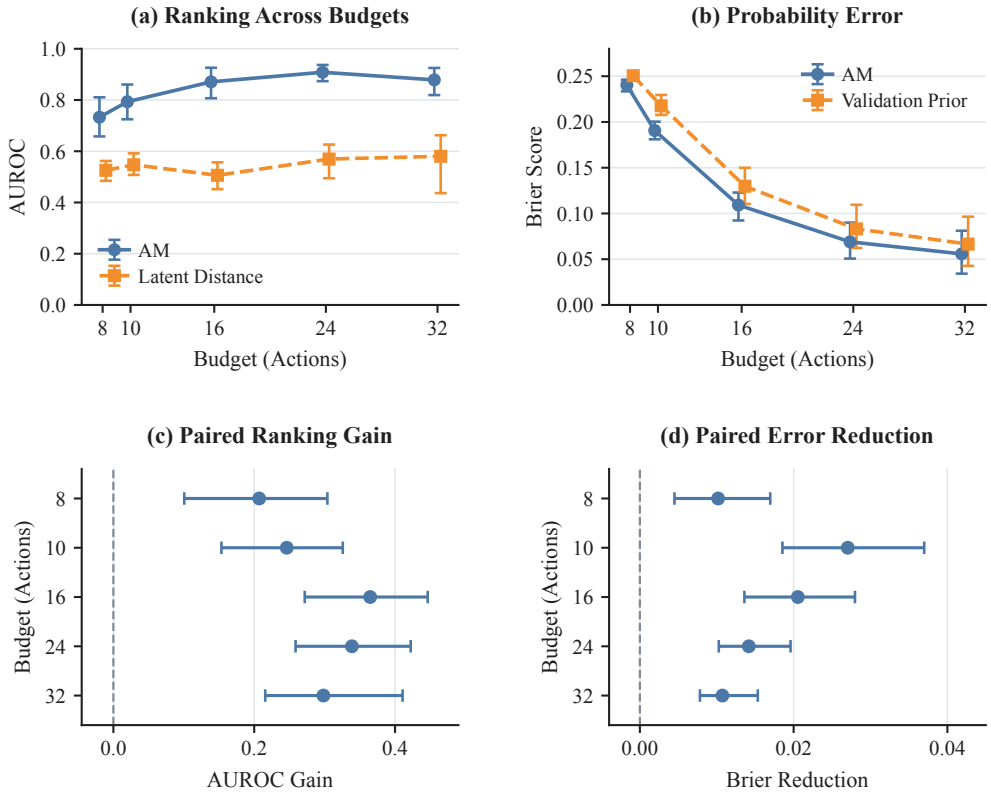

Figure S2: **Budget-dependent prediction quality and paired improvements.** (a) AUROC for AM and negative latent distance. (b) Brier scores for AM and independent budget-specific validation priors. (c) AUROC gain, AM minus latent distance. (d) Brier reduction, validation prior minus AM. Positive values in the lower panels favor AM; dashed lines mark zero difference. Points are layout-balanced estimates; bars are 95% percentile intervals from the same 2,000 layout-cluster resamples over six layouts. Upper-panel series are offset horizontally by 0.23 actions for readability. Budgets are 8, 10, 16, 24, and 32 actions from the same 400 attempts.

K PETAL REPRESENTATION DIAGNOSTICS

K.1 FROZEN-MODEL READOUT PROTOCOL

Frozen-model diagnostics quantify motion readout, target-specific responses, and sensitivity to controlled appearance changes. All control models are fixed, three probe fits are averaged before resampling, and intervals resample each cohort's independent units.

The Reach-Wall readout uses 16,384 training windows, 1,019 calibration windows, and 1,029 diagnostic windows from 15 source episodes. Source episodes are the independent resampling units, with overlapping windows kept together. Each predicted transition is produced by the retrained high-level predictor (L2) from an executed action segment. PETAL and matched random use the same feature dimension, preprocessing, probe architecture, and fitting budget. The Full-state features reference retains the complete source-endpoint features and gives the probe a larger input. The two proprioception references read out displacement from the predicted low-dimensional proprioceptive state, a channel that directly encodes the target quantity. Table S7 reports all five readouts.

Table S7: **Predicted-transition readout on Reach-Wall.** Mean Euclidean XYZ error is in millimeters; brackets give 95% source-episode bootstrap intervals. Parameter counts refer only to the probe.

Readout	Probe parameters	Error [95% interval]
PETAL	6,916	17.53 [15.98, 19.19]
Matched random	6,916	23.91 [20.89, 27.11]
Full-state features	26,215,172	51.26 [48.32, 54.09]
Predicted proprioception, MLP	1,796	17.57 [16.01, 19.23]
Direct predicted displacement	0	17.59 [16.04, 19.23]

PETAL's mean error is lower than matched random in 13 of the 15 episode averages, an overall reduction of 26.7%. It also matches the proprioception references (17.53 vs. 17.57 and 17.59 mm), although PETAL is trained without displacement supervision. The table uses a 10,000-draw percentile bootstrap summary; the main figure uses a separate 5,000-draw summary of the same observations.

K.2 PAIRED APPEARANCE INTERVENTION

The appearance cohort comprises 96 executed 15-step action prefixes, separate from the preceding readout cohort. Native, cool, and warm material colors share physical transitions, endpoints, and camera geometry. Frozen representations are recomputed under each appearance without refitting the probes; predicted-transition probes receive model-predicted endpoints rather than true future states. The 288 renderings form three paired conditions over the physical prefixes. Native proprioceptive inputs remain available in every condition.

Table S8: **Displacement error under recoloring.** Absolute errors in millimeters. Means and 95% paired query-bootstrap intervals use the same 96 physical prefixes.

Readout	Native	Cool	Warm
PETAL	28.80 [25.74, 31.97]	25.07 [22.93, 27.26]	28.31 [25.93, 30.74]
Matched random	91.17 [85.84, 96.86]	88.46 [83.19, 94.03]	109.94 [103.78, 116.47]
Full-state features	65.66 [63.42, 67.96]	68.24 [64.50, 71.98]	63.98 [59.73, 68.37]
Predicted proprioception, MLP	24.44 [21.93, 27.10]	20.80 [18.95, 22.73]	26.21 [23.84, 28.67]
Direct predicted displacement	24.20 [21.49, 27.05]	20.26 [18.30, 22.29]	26.08 [23.55, 28.73]

Under both recolorings, PETAL retains lower error and drift than matched random and Full-state features, reducing drift by 63.0% and 73.2% relative to Full-state features. Its drift approaches that of the proprioception references (Tables S8 and S9), indicating that PETAL largely discounts appearance changes that leave the physical transition unchanged.

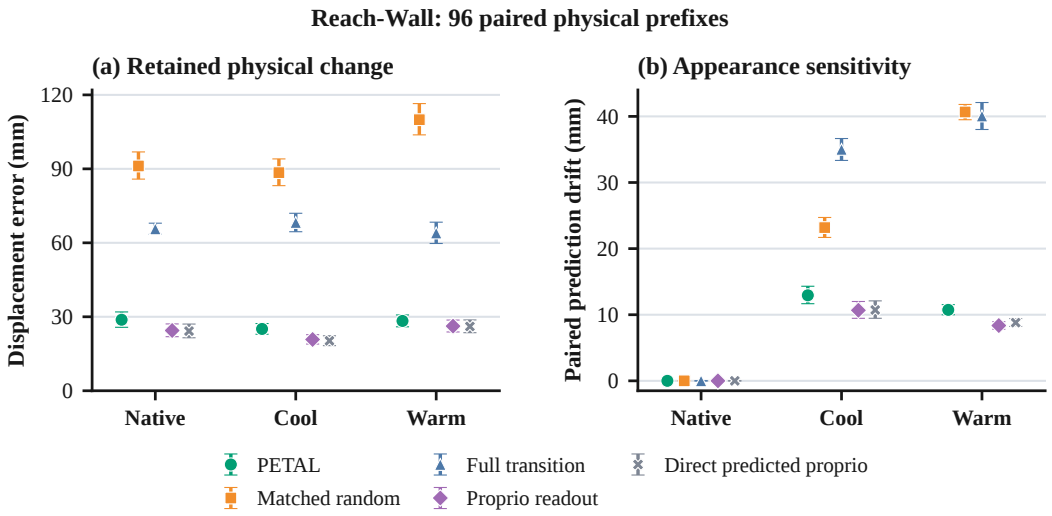

Figure S3: **Appearance controls on matched physical transitions.** (a) Displacement readout error and (b) drift from the native-color readout for all five representations. Points show means and bars show 95% paired query-bootstrap intervals over 96 physical prefixes, after averaging the three probe fits. Plot labels Full transition, Proprio readout, and Direct predicted proprio denote the Full-state features, Predicted proprioception (MLP), and Direct predicted displacement references. Native drift is zero by definition. Figure 7d summarizes the same cohort.

Table S9: **Recoloring-induced readout drift.** Same 96 prefixes as Table S8. Values are means in millimeters with 95% paired query-bootstrap intervals. Drift is the Euclidean distance from the native-color readout.

Readout	Cool	Warm
PETAL	12.94 [11.67, 14.31]	10.74 [10.00, 11.53]
Matched random	23.18 [21.69, 24.71]	40.67 [39.50, 41.80]
Full-state features	35.01 [33.34, 36.65]	40.06 [38.02, 42.10]
Predicted proprioception, MLP	10.67 [9.46, 12.00]	8.36 [7.81, 8.95]
Direct predicted displacement	10.72 [9.46, 12.10]	8.81 [8.23, 9.40]

K.3 Same-start target responses in Maze

The Maze cohort contains eight maps and 32 starting states, each paired with two physically executed targets over the same 40-primitive-step window. It contains 32 target pairs and 64 transitions. Let Δd denote the difference between the two executed displacements and $\Delta\hat{d}$ the corresponding difference between probe readouts. Positive alignment and signed response gain are defined as

$$\mathbf{1}\left[\langle \Delta\hat{d}, \Delta d\rangle > 0\right], \qquad g = \frac{\langle \Delta\hat{d}, \Delta d\rangle}{\|\Delta d\|_2^2}.$$

An exact target-induced displacement difference has gain one, while positive alignment measures agreement in direction. Table S10 reports direction and magnitude together, averaging across three probe fits. Source only reads out from source-state features alone, so both targets receive the same readout.

Table S10: **Target-specific responses in Maze.** Direction and magnitude over 32 pairs from eight maps. Alignment is the percentage of target pairs with a positive inner product between readout and executed displacement differences.

Readout	Positive alignment (%)	Signed response gain
PETAL	100.00	0.302
Matched random	81.25	0.217
Full-state features	100.00	1.137
Source only	0.00	0.000

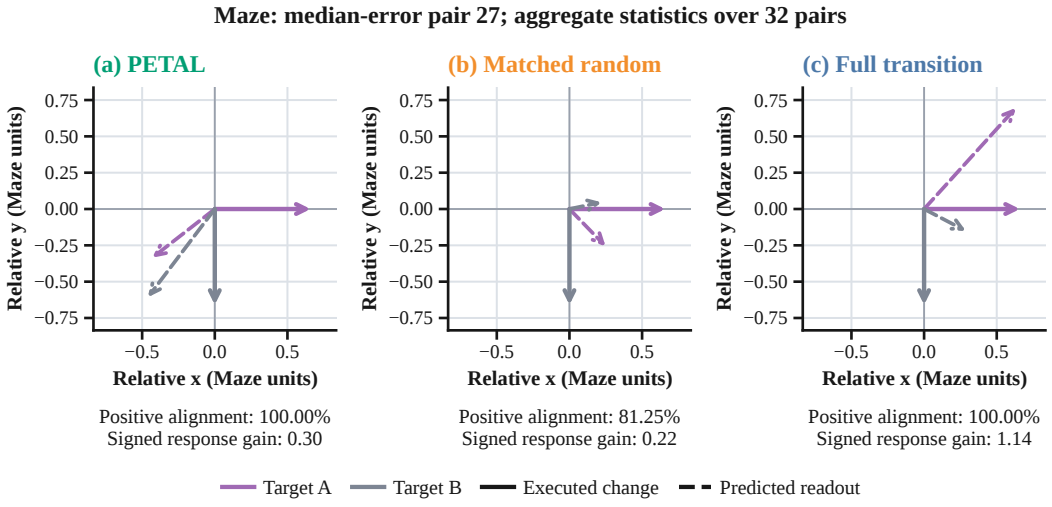

Figure S4: **Target-specific motion readouts in Maze.** Solid arrows show executed displacements; dashed arrows show probe readouts for two targets sharing a starting state. Pair 27 has the upper-median PETAL pair error among the 32 pairs. The plot label Full transition denotes the Full-state features reference. Readout arrows are shown without bias removal or rescaling. Footer statistics summarize all 32 pairs, averaged over three fits.

L Module ablations

L.1 Configurations and complete success summaries

Five configurations cover six tasks and eleven settings. L2 is the high-level model retrained on AM-selected trajectory segments, evaluated without AM or PETAL scoring during deployment. Every augmented configuration uses L2 as the hierarchy's high-level world model and adds AM, PETAL, or both; the last is ExAbs. HWM is task specific and denotes Hi-LeWM online in PushT and Cube. Tables S11 and S12 give the summaries underlying Table 5, and Appendix I reports completion steps.

Table S11: **Three-seed ablations.** Success rates (%) are mean ± sample SD. All augmented columns use L2 as the high-level world model. Bold marks the highest mean, including ties.

Setting	HWM	L2	+AM	+PETAL	ExAbs
PushT $d = 25$	72.7 ± 1.2	81.3 ± 1.2	82.7 ± 3.1	81.3 ± 1.2	$\mathbf{83.3} \pm 3.1$
PushT $d = 50$	30.7 ± 5.2	28.7 ± 1.9	28.0 ± 3.3	32.7 ± 3.4	$\mathbf{40.7} \pm 2.5$
PushT $d = 75$	14.0 ± 3.5	19.3 ± 1.2	21.3 ± 8.3	22.0 ± 3.5	$\mathbf{24.0} \pm 3.5$
Cube $d = 25$	68.0 ± 3.5	64.7 ± 5.0	66.7 ± 6.4	66.7 ± 3.1	$\mathbf{68.7} \pm 3.1$
Cube $d = 50$	56.0 ± 3.5	58.7 ± 4.2	$\mathbf{59.3} \pm 6.4$	58.7 ± 4.2	$\mathbf{59.3} \pm 6.4$
Cube $d = 75$	68.7 ± 8.3	69.3 ± 9.5	68.7 ± 10.3	68.7 ± 11.0	$\mathbf{72.7} \pm 4.6$

Table S12: **Fixed-set ablations.** Values are success rates (%) from one evaluation per configuration. All augmented columns use L2 as the high-level world model. Bold marks the largest rate, including ties.

Setting	HWM	L2	+AM	+PETAL	ExAbs
Maze Medium	92.50	95.00	**97.50**	92.50	**97.50**
Maze Hard	77.50	82.50	82.50	85.00	**90.00**
Reach-Wall	34.38	36.46	37.50	**47.92**	45.83
Counterflow	35.50	37.50	44.50	**56.50**	54.50
Two-room	7.81	9.38	15.63	15.63	**20.31**

ExAbs improves success over HWM in all eleven settings and attains the highest or tied-highest success in nine of them. Retraining on AM-selected segments (L2) already improves over HWM in nine settings, and completion-aware selection and effect-based control add complementary, task-dependent gains on top of it.

L.2 Protocol distinctions

Reach-Wall. All ablation rates use goal attainment within the same 100-step evaluation budget. The HWM reference is 33/96 (34.38%), as in Table 2. The L2, +AM, +PETAL, and ExAbs counts are 35, 36, 46, and 44 out of 96. Appendix H.1 specifies the completion rule and final direct-goal control phase.

Cube. +AM uses AM trained with the full-latent executor, and ExAbs uses AM trained with the PETAL-enabled executor, so each column pairs AM with the local-control objective that produced its labels.

Two-room. The module comparison uses the same 64 test episodes and 300-step budget as Table 2.

L.3 Additional robustness checks

Counterflow: stronger baselines and sensitivity. Table S13 reruns Counterflow on the same 200 tasks, 250-step budget, checkpoints, and planner seed as Table 2, changing one factor per row. Doubling HWM's candidates at both levels or lengthening its local horizon raises its success by at most 6.0 points, and both variants remain below ExAbs, so the gain is not explained by additional search. Across seven single-factor changes to ExAbs's AM weight λ_{AM}, reference budget B_0, and threshold weight λ_{bar} (Equation (6)), success stays within 50.5–56.5%, and every paired interval against the default includes zero. All seven settings exceed every HWM variant.

Table S13: **Counterflow robustness.** Success counts (rates, %) over the same 200 tasks, one evaluation per row. Differences are relative to default ExAbs in percentage points, with 95% paired bootstrap intervals over tasks. Multipliers scale the default value.

Method	Change	Success	Δ vs. ExAbs
ExAbs	Default	109/200 (54.50)	–
HWM	Default	71/200 (35.50)	$-19.0\ [-28.0, -9.5]$
HWM	$2\times$ candidates, both levels	72/200 (36.00)	$-18.5\ [-27.5, -9.5]$
HWM	Local horizon $h{+}5$	83/200 (41.50)	$-13.0\ [-22.5, -3.5]$
ExAbs	$\lambda_{\mathrm{AM}} \times 2$	110/200 (55.00)	$+0.5\ [-5.5, 6.5]$
ExAbs	$\lambda_{\mathrm{AM}} \times 0.5$	106/200 (53.00)	$-1.5\ [-7.5, 4.5]$
ExAbs	$B_0 \times 1.5$	106/200 (53.00)	$-1.5\ [-7.5, 4.5]$
ExAbs	$B_0 \times 0.5$	101/200 (50.50)	$-4.0\ [-10.0, 2.0]$
ExAbs	$\lambda_{\mathrm{bar}} \times 2$	101/200 (50.50)	$-4.0\ [-10.5, 2.5]$
ExAbs	$\lambda_{\mathrm{bar}} \times 0.5$	113/200 (56.50)	$+2.0\ [-4.0, 8.0]$
ExAbs	$\lambda_{\mathrm{bar}} = 0$	112/200 (56.00)	$+1.5\ [-5.0, 8.0]$

Two-room: additional planner seeds. Table S14 repeats the Two-room comparison with two new planner seeds, keeping the frozen models, 64 test episodes, and 300-step budget fixed. ExAbs

succeeds in 12 and 9 episodes, against 1 and 0 for HWM; averaged over the two seeds, the gain is 15.6 points (95% interval $[7.8, 24.2]$, resampling test episodes).

Table S14: **Two-room with additional planner seeds.** Success counts (rates, %) over 64 test episodes. p is the two-sided exact McNemar test on paired episodes.

Planner seed	HWM	ExAbs	p
Seed A	1/64 (1.56)	**12/64 (18.75)**	0.003
Seed B	0/64 (0.00)	**9/64 (14.06)**	0.004
Mean	0.78	**16.41**	–

M Qualitative execution sequences

M.1 Reading the simulation sequences

Figures S6–S19 show selected executions from the seven simulated tasks. Each task has one block containing three selected successes and one failure of ExAbs, followed by two task-matched comparisons with ExAbs above the hierarchical baseline. Each row contains nine observed states or route snapshots, two reconstructions of high-level waypoints issued to the controller, and the task goal. Red dashed frames (columns labeled Subgoal) identify waypoints, and red solid frames identify goals. Waypoint images decode full-state latents with a frozen, visualization-only decoder shared within each task.

Display time is normalized to each shown window. The two methods in each pair use the same original observation times over a common window ending at the first success of ExAbs. Baseline failure is determined over the complete original evaluation budget. Waypoint columns show representative planner outputs placed for readability; intermediate replanning outputs are not shown. Figure S5 collects the paired Button–Window, Two-room, and Counterflow executions discussed in Section 4.2.



Figure S5: **Execution under sequential and navigation constraints.** Button–Window, Two-room, and Counterflow, from top to bottom. Each pair shows ExAbs above HWM from matched starts and goals. Red dashed columns are reconstructed subgoals; other frames are observations. Time is normalized to the displayed window.

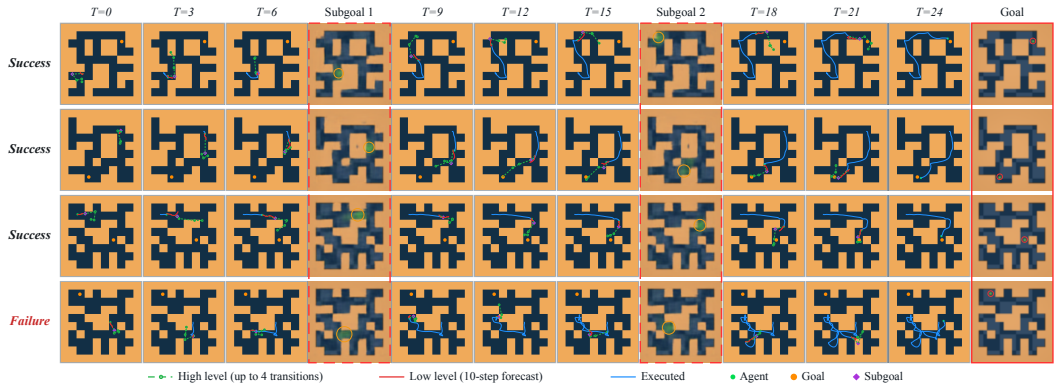

Figure S6: **Maze successes and failure.** Green dashed routes show up to four predicted macro-transitions, red routes show ten-step low-level forecasts, and blue routes show execution. Purple diamonds mark issued waypoints. Green and orange dots mark the agent and goal. Forecast positions use a shared trained readout. Straight lines connect predicted macro-states in temporal order.

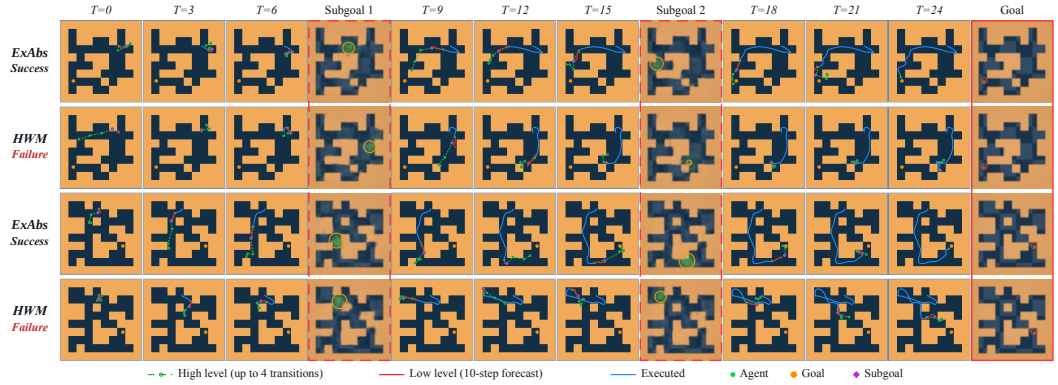

Figure S7: **Maze paired comparisons.** ExAbs follows the intervening passages to the goal, while the paired HWM executions remain unsuccessful within budget. Route colors follow Figure S6. The common display windows end at the first success of ExAbs.



Figure S8: **Button–Window successes and failure.** Successful sequences progress from pressing the unlocking button to moving the window toward its target. The failed sequence predicts states near the relevant objects but does not complete the operation. Outcomes follow the simulator's lock and window states.

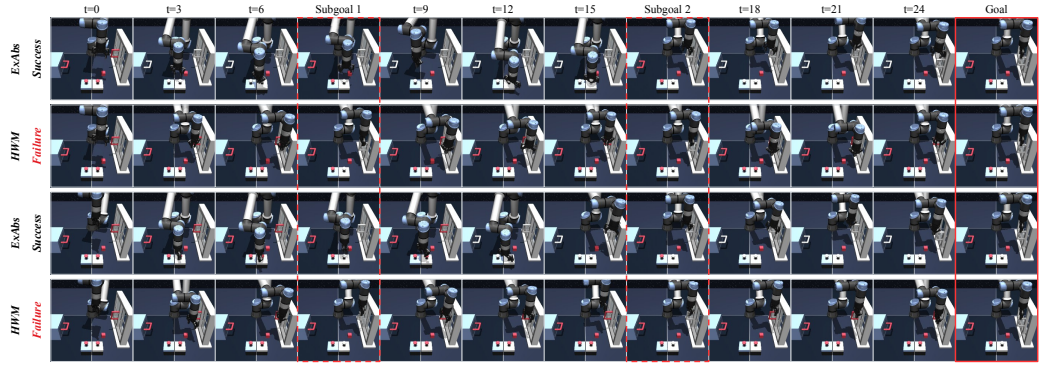

Figure S9: **Button–Window paired comparisons.** ExAbs completes the prerequisite button interaction and the subsequent window movement, while HWM remains unsuccessful over the complete budget. Each row includes its method's predicted waypoints. Paired rows share initial states, goals, and observed time points.

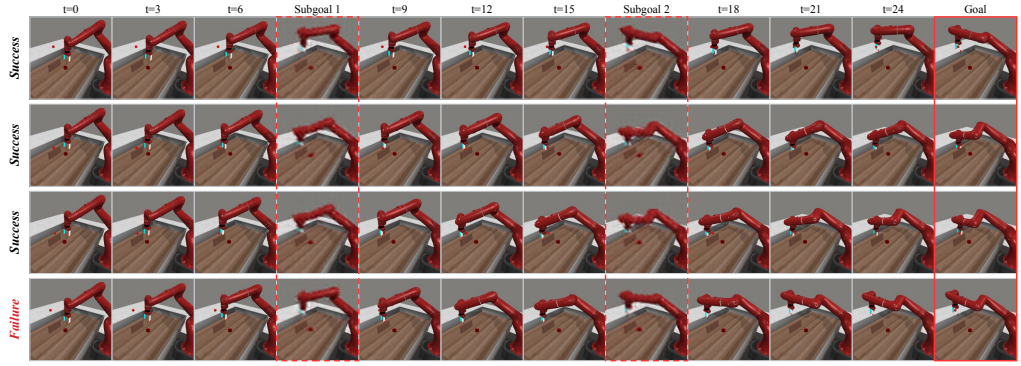

Figure S10: **Reach-Wall successes and failure.** Successful executions move the end effector across the wall into the target region. Success denotes the first satisfaction of the goal condition within budget. Decoded waypoints show requested states, and observed frames in successful rows show the end effector's passage across the wall.

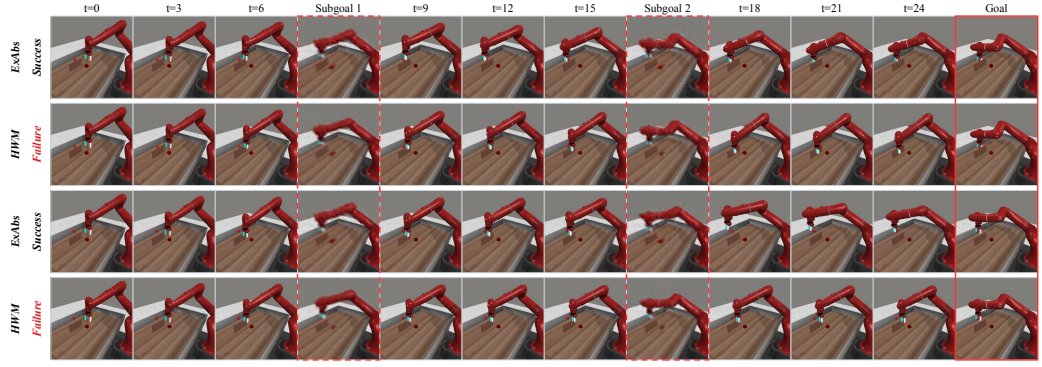

Figure S11: **Reach-Wall paired comparisons.** Both ExAbs executions cross the wall and reach the target, whereas neither HWM execution crosses the wall within budget. Success follows the first-hit criterion of Appendix H.1.

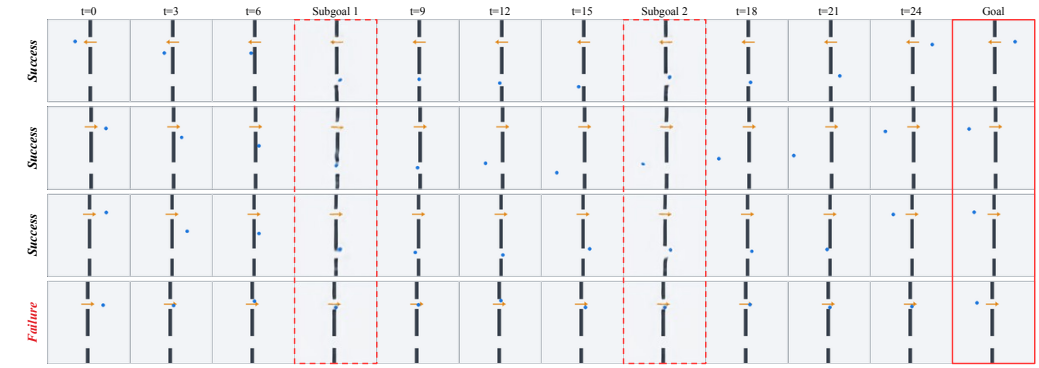

Figure S12: **Two-room successes and failure.** ExAbs traverses the accessible passage to reach the other room. The final phase of the task budget uses direct-goal low-level control. The failed sequence remains incomplete within budget.

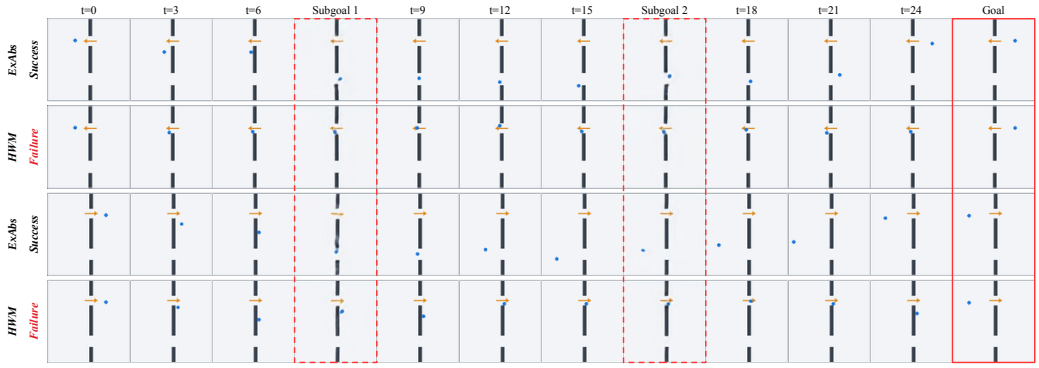

Figure S13: **Two-room paired comparisons.** ExAbs uses the accessible passage, while HWM remains near the wall and one-way-door region. Execution switches to direct-goal low-level control in the final phase of the task budget.

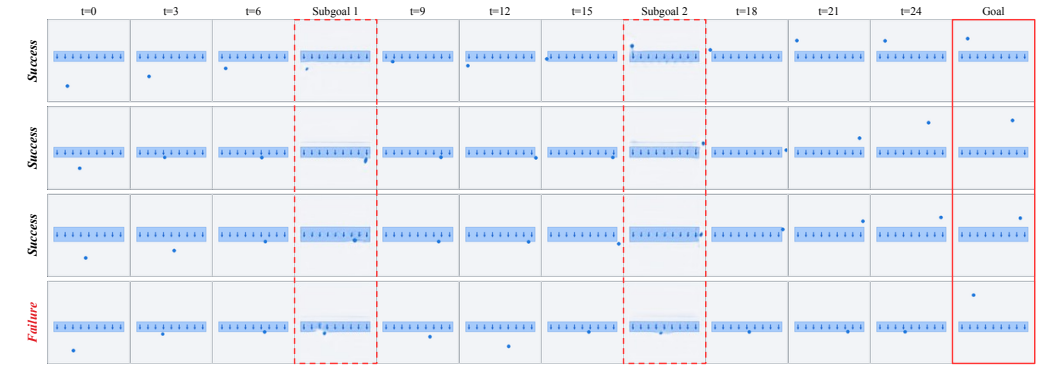

Figure S14: **Counterflow successes and failure.** The current acts inside the blue band, while narrow side passages permit progress toward the goal. The successful ExAbs sequences approach and traverse these passages. The failed sequence remains incomplete at the end of its budget.

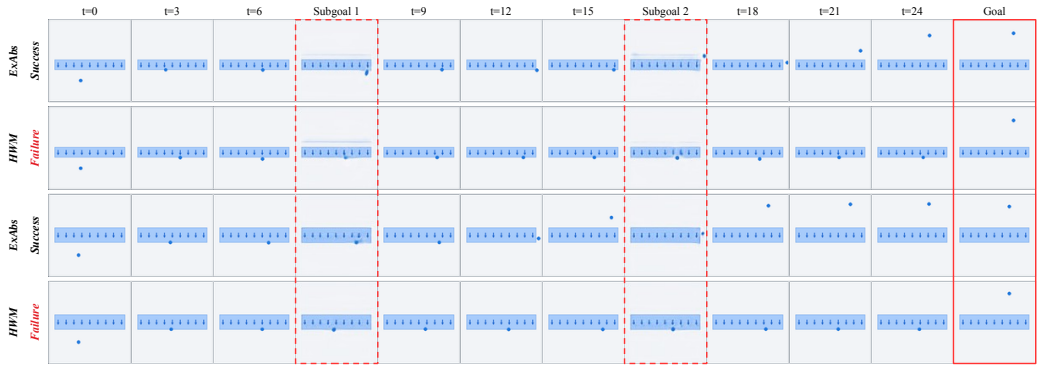

Figure S15: **Counterflow paired comparisons.** ExAbs progresses through the side passages, whereas HWM remains near the lower edge of the current band. The final phase uses direct-goal low-level control.

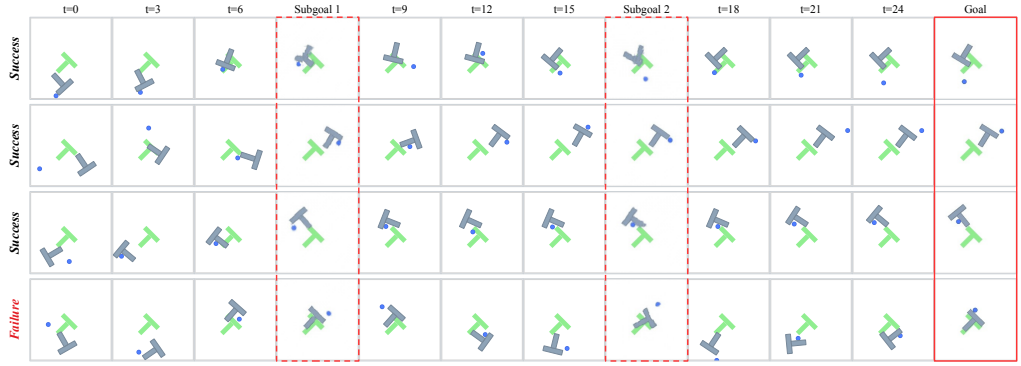

Figure S16: **PushT successes and failure.** The pusher establishes contact and translates or rotates the object toward the task goal. The fixed green silhouette is a scene reference, while the final goal column specifies the requested state.

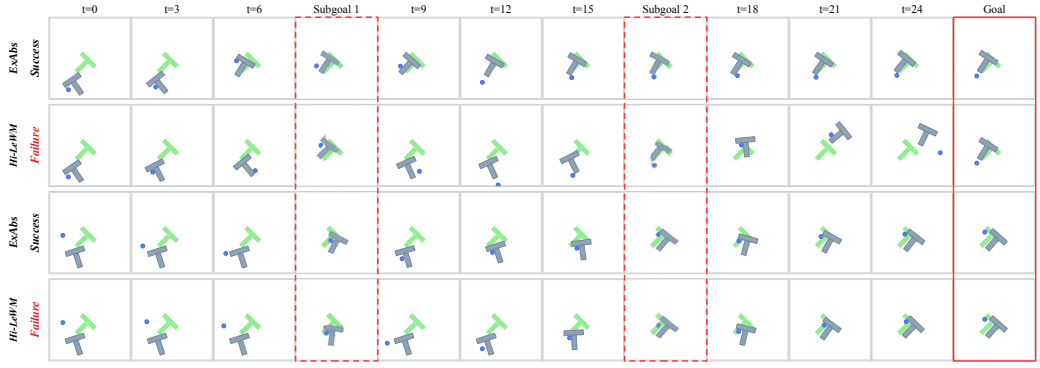

Figure S17: **PushT paired comparisons.** Hi-LeWM is the HWM-style hierarchy over LeWM. In the second pair, Hi-LeWM approaches the goal but remains outside the success tolerance. Observations and goal images are rendered from recorded physical states, and the waypoints are decoded predictions.

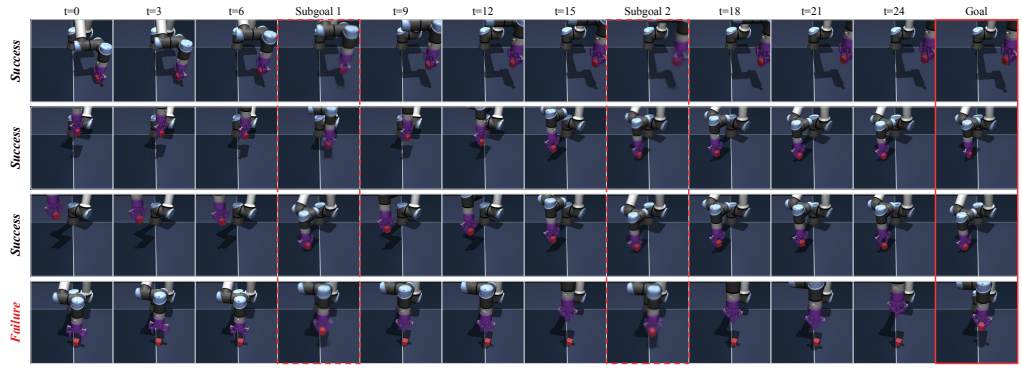

Figure S18: **Cube successes and failure.** These examples start with the red cube already held. Successful executions transport it to the target. In the failed example, object transport does not follow the gripper motion through to completion.

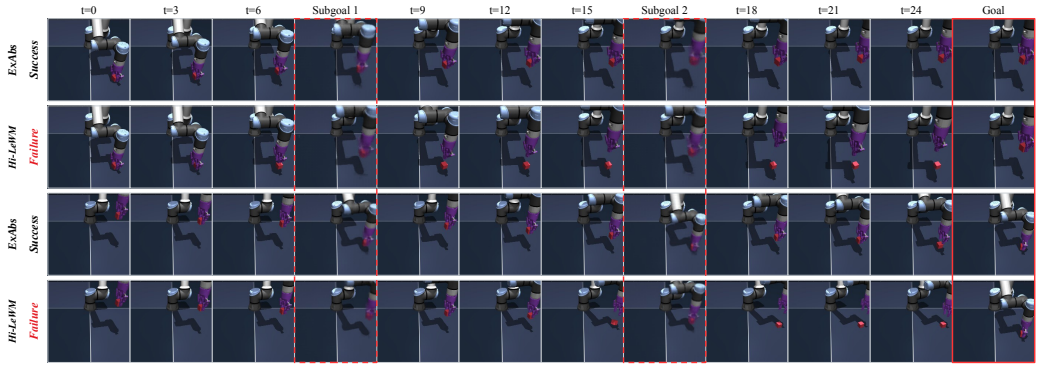

Figure S19: **Cube paired comparisons.** ExAbs completes the held-object transfers, while the Hi-LeWM counterparts remain unsuccessful within the full budget. Hi-LeWM is the HWM-style hierarchy over LeWM. Both methods use the same frozen decoder for the full-state waypoints.

M.2 PHYSICAL-ROBOT SEQUENCES

Figures S20–S22 show selected executions complementing the aggregate results in Table 4. Red dashed frames mark model-predicted subgoals reconstructed by a frozen decoder used only for visualization; red solid frames contain deployment goal images. Other frames are observations. Normalized time labels refer to the first row; other rows align by action order and stage rather than elapsed time.

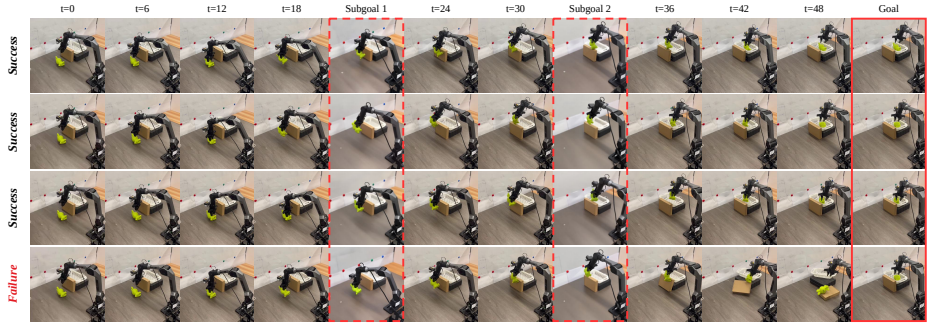

Figure S20: **Basket transfer.** Three selected successes and one failure.

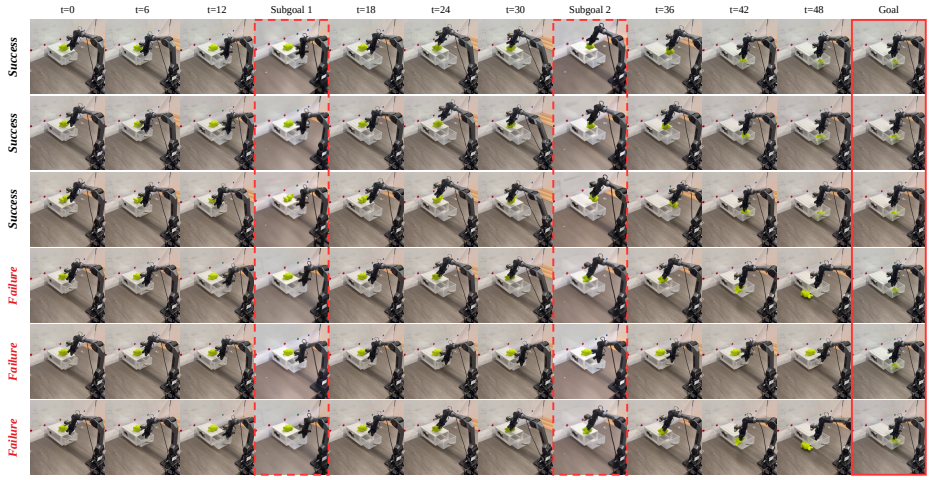

Figure S21: **Drawer loading.** Three selected successes and three failures.

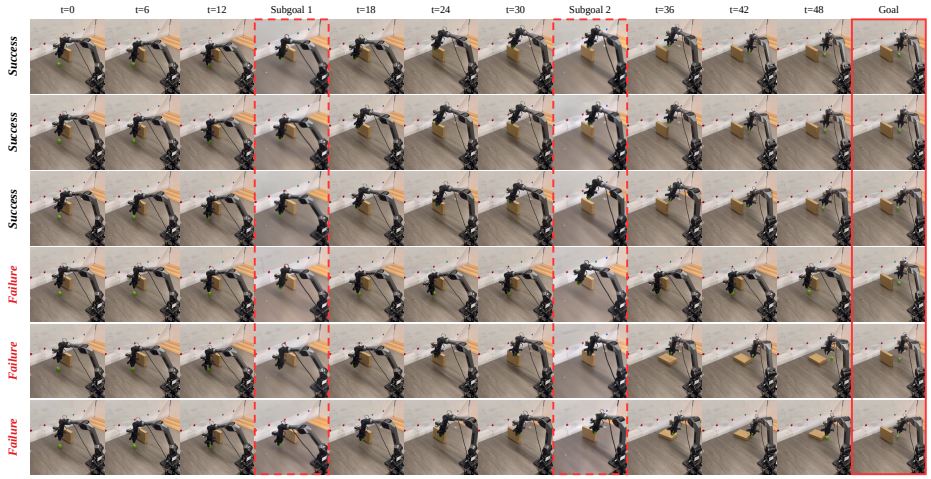

Figure S22: **Raised transfer.** Three selected successes and three failures.