WORLD ACTION PLANNER: GENERALIZABLE ROBOT DECISION-MAKING WITH ACTION-CONDITIONED WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Building generalizable robot agents for diverse applications remains a fundamental challenge. While imitation learning-based policies can perform well in familiar training environments, they often struggle to generalize to novel scenes, layouts, and task compositions. To this end, we present **World Action Planner**, an agentic robot planning system in which the agent searches for and composes executable action plans through imagination with an action-conditioned world model. The search proceeds in a coarse-to-fine manner. First, the agent performs global action optimization by reasoning over imagined world-model rollouts to identify potential failures and refine the proposed action plan. It then performs local action search, comparing the imagined future outcomes of neighboring candidates to select the best action for execution. Across compositional long-horizon tasks, novel object layouts, and real-robot planning on novel tasks without expert demonstrations, World Action Planner consistently outperforms state-of-the-art end-to-end generalist policy models and VLM planners, demonstrating the effectiveness of world-model-based action search for generalizable robot decision making. Qualitative results and videos are provided in the supplementary material.

1 INTRODUCTION

Developing generalizable robotic agents capable of performing diverse tasks across heterogeneous environments remains a fundamental challenge in AI. Recent progress has been driven by end-to-end (E2E) imitation learning-based policies that leverage pretrained Vision-Language Models (VLMs) (Intelligence et al., 2025) or video generation models (Ye et al., 2026) as backbones. However, these approaches are often fundamentally limited by the scope of the demonstration data available during training. For instance, a policy trained only on individual pick-and-place trajectories may fail to navigate between objects when executing composed, long-horizon tasks. Furthermore, E2E models can overfit to spurious motion patterns in demonstrations; if the training data consistently depicts grasping at a specific coordinate, the policy may continue targeting that position even after the object has moved. Consequently, these models struggle in novel scenarios that require systematic reasoning or flexible planning beyond the trajectory distributions observed during training.

In this work, we depart from the end-to-end (E2E) imitation learning paradigm and instead propose an agentic planning framework centered on a generalizable world model that can faithfully imagine the outcomes of diverse robot actions. When facing a new task, the agent can synthesize novel action plans through search and composition within world-model imagination, enabling behaviors that remain challenging for end-to-end policies tightly coupled to previous expert trajectories. Instead of requiring demonstrations from the same task and environment, we only need to learn reusable dynamics with a world model, potentially from data collected across different tasks and environments, and then search directly for executable robot actions through imagination under the task objective in the current environment. This decouples decision making from the expert demonstrations seen during training and enables systematic generalization to novel states, tasks, and environments.

To this end, we propose **World Action Planner** (Fig. 1), a world-model-based robot planning system designed to solve new tasks in novel scenes and layouts. We train a generalizable video world model conditioned on robot pose images (Fig. 2), enabling precise conditioning on robot actions

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

and high-fidelity prediction of physical interactions. By imagining future action rollouts, the world model reveals their physical consequences before execution, allowing an out-of-the-box VLM to evaluate predicted outcomes and provide feedback for action optimization. To efficiently search the action space, we introduce a coarse-to-fine action search hierarchy that combines agent-driven global action optimization with physically grounded local action candidate search, both performed by reasoning over imagined world-model rollouts (Fig. 3).

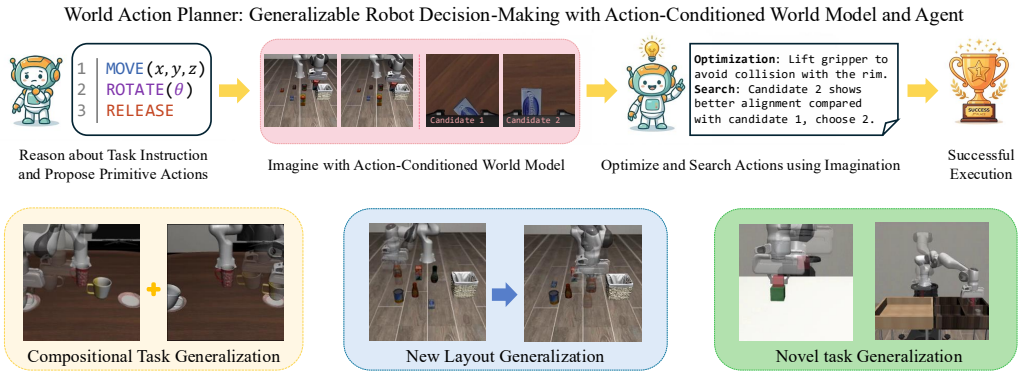


Figure 1: We introduce **World Action Planner**, a planning system in which the agent reasons about and refines its action plans via world model imagination, generalizing to novel scenarios and solving new tasks at test time.

We demonstrate the generalization capability of our method across compositional tasks, novel layouts, and novel task scenarios without expert demonstration in both simulation and real-world environments. In compositional long-horizon tasks, our method successfully handles transitions between subtasks by proposing and refining maneuvers through world-model imagination, whereas end-to-end models such as VLAs often stagnate after completing the first subtask. In spatial generalization settings with modified object positions, our method identifies the updated target locations and completes the task, while imitation learning policies tend to reach toward object coordinates observed during training. By reasoning over imagined world-model rollouts, global optimization identifies and avoids physical risks such as collisions, significantly improving safety and success rates in real-robot settings. Local action search further evaluates neighboring candidates to identify precise actions for fine-grained manipulation tasks such as mug grasping and cube stacking. Together, these components form a coarse-to-fine action search strategy that achieves substantially greater planning efficiency than conventional action-sampling methods such as MPPI (Hansen et al., 2023) and Best-of-N (Cui et al., 2026).

We summarize our main contributions as follows: (i) we propose World Action Planner, an agentic planning system that proposes and refines robot actions through a physically grounded coarse-to-fine search over imagination from a generalizable robot pose-image conditioned world model; (ii) we provide theoretical insights into the multi-task generalization advantages of model-based planning over imitation learning; and (iii) we demonstrate that our method significantly outperforms SOTA policies and VLM planners across compositional tasks, novel layouts, and novel task generalization scenarios.

2 Related Works

Here, we review the most closely related works on planning with world models and VLMs. For an extended discussion on controllable world models and generalist policies, see Appendix A.

To use world-model imagination for decision making, prior MPC-based methods (Hansen et al., 2022; Assran et al., 2025; Jain et al., 2025) typically rely on sampling-based optimization such as MPPI, while recent works (Sharma et al., 2026; Jia et al., 2026) instead train policies directly in imagination. To leverage the semantic knowledge and reasoning capabilities of VLMs, many methods use them for high-level planning while relying on separate grounding or control modules to produce executable actions (Liu et al., 2024a; Duan et al., 2024; Feng et al., 2025), whereas our method searches directly over executable robot controls grounded by the world model. More recent approaches (Zhao et al., 2024; Ning et al., 2025) combine predictive models or digital twins with VLM-based evaluation, but still rely on inefficient CEM sampling. Moreover, Zhao et al. (2024) conditions its world model on low-dimensional actions rather than explicit robot pose images, potentially limiting generalization, while Ning et al. (2025) depends on VLM-based real-to-sim reconstruction whose accuracy may constrain planning. Concurrent work Cui et al. (2026) improves a base VLA by imagining and ranking K policy-sampled action candidates. In contrast, our method enables direct synthesis of actions beyond the policy proposal distribution while generalizing to novel tasks and environments.

3 METHODS

Here we describe our world model with robot pose-image conditioning in Section 3.1 and the world action planner with our agentic action search in Section 3.2.

3.1 POSE-IMAGE CONDITIONING FOR ACTION-CONTROLLED ROBOT WORLD MODELING

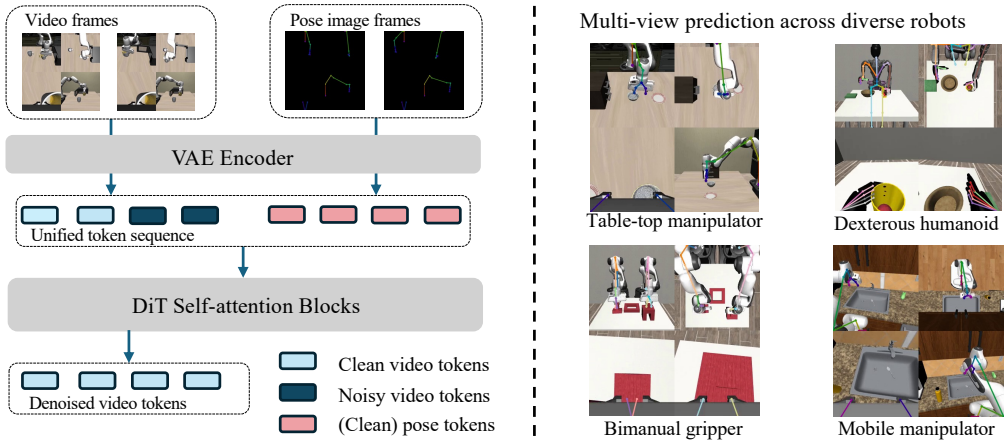


Figure 2: **Illustration of our pose-image conditioned world model**

Prior robot action-conditioned video diffusion models typically condition on low-dimensional action vectors through AdaLN-Zero modulation (Quevedo et al., 2025; Zhu et al., 2025) or cross-attention (Guo et al., 2025). However, such low-dimensional action representations can entangle robot kinematics with environment dynamics, making generalization to novel actions difficult. In our approach, we first compute the future robot joint configurations induced by the actions and render them as pose-skeleton images from the corresponding camera viewpoints. This representation makes the robot's future motion explicit in the visual input, allowing the world model to focus on predicting how the environment responds to that motion. As a result, the model can reuse learned object-interaction dynamics while generalizing to novel robot motions that were not observed during training. More details about pose-image conditioning are provided in Appendix C.1.1.

Inspired by Tan et al. (2025), we encode the pose image frames using the VAE of the video model and concatenate the resulting pose image tokens with the video tokens into a unified sequence. We adopt multi-view prediction by concatenating third-person and wrist-view camera feeds into a grid for each video and pose-image frame (Guo et al., 2025), so that the model can infer the relative 3D position of each robot joint from multi-view images. We train the model using diffusion-forcing (Song et al., 2025; Chen et al., 2025) with the flow-matching objective (Lipman et al., 2022), where we add independent random noise to the history video tokens and uniform noise to the future video tokens, while keeping the pose-image tokens noise-free for conditioning.

3.2 WORLD ACTION PLANNER

To achieve generalizable robotic decision-making, we develop World Action Planner (WAP), an agentic planning framework that generalizes to new tasks and environments. Rather than end-to-end imitation learning, WAP combines agent-driven global trajectory optimization with physically grounded local refinement to search and compose robot actions. We introduce the agentic action proposal and policy prior composition in Section 3.2.1, and focus on our core action search strategy in Section 3.2.2. The complete procedure is given in Alg. 1, with additional details in Appendix C.2.

3.2.1 ACTION PROPOSALS AND POLICY PRIORS

When generalizing to new tasks and environments, we leverage the broad world knowledge of foundation VLMs to propose action primitives in settings where demonstration data is unavailable. The VLM generates a sequence of primitive actions, such as `MOVE`, `ROTATE`, `GRASP`, and `RELEASE`. For each `MOVE` action, the VLM identifies the target position across *multi-view* images. We then triangulate the corresponding 2D pixel coordinates across views to recover the target position in 3D space, eliminating the need for explicit depth information. We employ a low-dimensional

robot controller to convert the target EEF pose specified by each goal into a sequence of executable low-level actions, where the goals are being optimized and refined in the subsequent search process.

When policy priors are available for local manipulation, such as grasping, the agent flexibly composes them in states where the policy remains in distribution. For example, when an object is located at a novel position, executing the grasping policy end-to-end may fail. Instead, the agent first navigates to the vicinity of the target object and then searches for states from which the local grasping policy is likely to succeed by imagining future policy rollouts with the world model. Treating learned policies as modular action priors and composing them within the planning process therefore provides a generalization pathway for extending their capabilities beyond the expert demonstrations.

3.2.2 Agentic Action Search with World Model

Here we discuss our action search strategy. Searching for successful robot actions in manipulation is challenging for two main reasons: the action space is high-dimensional and continuous, while task success is often sparse and concentrated around a small set of feasible poses, providing little continuous guidance for optimization. Generic sampling methods such as Best-of-N and CEM are therefore inefficient in this setting: they explore large action spaces through weakly structured proposal distributions, causing many world-model evaluations to be spent on physically implausible candidates, redundant variations, or actions that repeat the same underlying failure. Instead, we organize search into a coarse-to-fine hierarchy that operates in a low-dimensional goal space. Guided global optimization first uses agentic feedback to correct large, semantically meaningful errors in the proposed trajectory, while physically grounded local search subsequently concentrates computation on a compact neighborhood of plausible actions for fine-grained refinement. This procedure is illustrated in Figure 3.

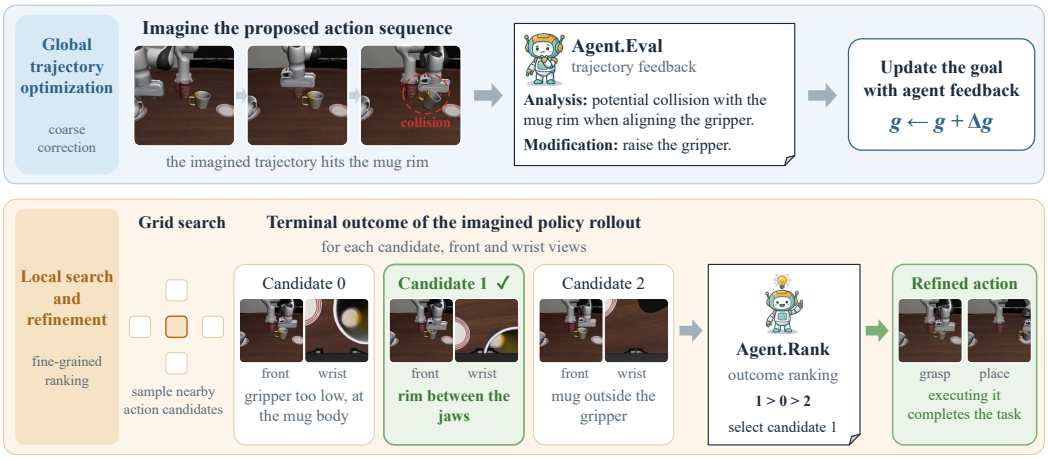


Figure 3: **Coarse-to-fine action optimization in WAP.** Global trajectory optimization imagines the proposed action sequence with the action-conditioned world model, while the VLM provides feedback to improve safety and task success. Local search then performs a grid search over neighboring actions, imagines the future outcome of each candidate, and selects the optimal action.

Global Trajectory Optimization. The initially proposed action sequence may contain large-scale errors that are difficult to avoid from the current observation alone, such as violating physical constraints or failing to account for obstacles in a novel environment. For instance, a naive `MOVE` command between two objects may cause the gripper to collide with an intervening obstacle. Although such failures may be difficult to anticipate without task-specific motion priors, they often become immediately apparent once the proposed trajectory is visualized through world-model imagination. Rather than exploring random perturbations around an erroneous proposal, the agent directly diagnoses the failure from the imagined visual rollout and applies a targeted modification to the corresponding goal action. For example, if the rollout reveals a potential collision between the gripper and a basket rim, the agent may raise the target pose to provide sufficient clearance. Similarly, if an object is predicted to be dropped behind a box, the agent may shift the goal position forward. This targeted refinement allows the planner to use a single imagined trajectory to correct major trajectory-level errors.

Local Search and Refinement. Global optimization is effective at correcting conspicuous trajectory-level errors, but precise manipulation often depends on subtle geometric differences among nearby candidate poses, which can be difficult to evaluate directly because informative rewards may emerge

Algorithm 1 World Action Planner (WAP)

Require: VLM `Agent`, robot controller ϕ, world model `WM`, optional policy π
Input: Task description ℓ, initial state s, environment `env`
while not `done` **do**
  ▷ **Agent Action Proposal**
  Propose primitive action goals using the VLM agent: $g = \texttt{Agent.Proposal}(s, \ell)$
  Translate the proposed goal into robot actions: $a = \phi(g, s_{\text{proprio}})$
  ▷ **Global Optimization**
  Imagine the proposed action using the world model: $\hat{s}_{\texttt{next}} = \texttt{WM}(s, a)$
  Obtain agent feedback: $\Delta g = \texttt{Agent.Eval}(\hat{s}_{\texttt{next}}, \ell)$
  Update the goal and robot action: $g \leftarrow g + \Delta g$, $a = \phi(g, s_{\text{proprio}})$
  ▷ **Local Search and Refinement**
  Generate local goal candidates: $g_1, g_2, \ldots, g_N = \texttt{GridSearch}(g)$
  Translate goal candidates into robot actions: $a_i = \phi(g_i, s_{\text{proprio}})$, $i = 1, 2, \ldots, N$
  Imagine the candidate actions: $\hat{s}_i = \texttt{WM}(s, a_i)$, $i = 1, 2, \ldots, N$
  if use policy **then**
    Imagine the policy rollout after each candidate action: $\hat{s}_i = \texttt{WM}\left(\hat{s}_i, \pi(\hat{s}_i)\right)$, $i = 1, 2, \ldots, N$
  end if
  Select the best candidate using the VLM agent: $i^* = \texttt{Agent.Rank}(\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_N, \ell)$
  ▷ **Execute the Actions**
  Execute the selected action: $s, \texttt{done} = \texttt{env.Step}(s, a_{i^*})$
  if use policy **then**
    Roll out the policy following the selected action: $s, \texttt{done} = \texttt{env.Step}(s, \pi(s))$
  end if
end while

only after subsequent execution. We therefore assess each candidate through its world-model-imagined downstream outcome. Specifically, we perform grid search over neighboring actions and states in the low-dimensional goal space and use the action-conditioned world model to generate a *future policy rollout* from each candidate. These rollouts reveal the downstream consequences of the local candidates, allowing the VLM to rank them according to their imagined task outcomes rather than ambiguous intermediate states. For example, when refining a pre-grasp pose, we evaluate neighboring positions by imagining the subsequent grasping policy from each candidate and select the pose whose final imagined frame most clearly indicates a successful grasp. This outcome-based local search focuses world-model evaluations on a small set of physically meaningful alternatives and enables fine-grained action refinement without requiring dense rewards or reliable intermediate value estimates.

4 Theoretical Insights

We ask when learned-model planning transfers across tasks more effectively than imitation, and what determines the quality of WAP's selected action. We state the main results informally here and defer the full settings, assumptions, and proofs to Appendix B.

4.1 When Does Model-based Planning Outperform Imitation Learning

Setting. Tasks form a contextual MDP (Hallak et al., 2015), indexed by $c \in \mathcal{C}$. They share the state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, initial distribution ρ, and transition kernel $\mathbb{P}$, but have different rewards r_c. Each learner receives K training episodes and, as is standard when assembling a multi-task dataset, fixes how many go to each task before collecting any of them. The imitation learner observes expert actions, but not rewards, and fits a task-conditioned policy. The model-based learner chooses its own exploration policy, estimates the shared dynamics and, when needed, the reward model, then plans for the test task in the estimated MDP. We measure suboptimality per task and on average across tasks.

Tabular Setting. Let $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{C}$ be finite, with unknown dynamics and a known reward for each task. This resembles WAP: the world model must learn the dynamics, while a VLM can judge success from a terminal state.

Theorem 1 (Tabular, informal)**.** *With K episodes, **(i)** a model-based planner attains $\tilde{\mathcal{O}}(K^{-1/2})$ suboptimality on every task, independent of the number of tasks $|\mathcal{C}|$; yet **(ii)** for any imitation learner, there exists a contextual MDP instance where the average suboptimality $\Omega\big(\min\{H,\ H|\mathcal{S}||\mathcal{C}|/K\}\big)$.*

The $|\mathcal{C}|$ factor reflects what each sample reveals: an explored transition constrains the shared kernel $\mathbb{P}$ and so informs planning for every task, while an expert action labels only its own, leaving roughly $K/|\mathcal{C}|$ demos per task. Two limits apply. The gain is in the task count rather than the rate, so planning wins only once $|\mathcal{C}| \gtrsim H^4|\mathcal{S}||\mathcal{A}|/\epsilon$; and it needs rewards evaluable on unseen tasks, the VLM's role in WAP. A planner forced to estimate each r_c separately pays the same penalty (Appendix B).

Linear Function Approximation. In a linear MDP (Jin et al., 2023), known features $\phi(s,a) \in \mathbb{R}^d$ satisfy $\mathbb{P}(\cdot \mid s,a) = \langle \phi(s,a), \boldsymbol{\mu}(\cdot) \rangle$. We also assume known task features $\psi(c) \in \mathbb{R}^{d'}$ and unknown matrices Θ_h, with $\mathbb{E}[r_h(s,a;c)] = \phi(s,a)^\top \Theta_h \psi(c)$. Here neither learner knows the rewards, so both must transfer through a learned representation. For imitation, we consider classifier policies $\pi_f(s;c) = \operatorname{argmax}_a f(\phi(s,a), \psi(c))$ (Rajaraman et al., 2021); their ability to transfer depends on the complexity of the shared score f as a function of the task feature.

Theorem 2 (Linear, informal)**.** ***(i)*** *A planner that learns the shared dynamics and bilinear reward model attains $\tilde{\mathcal{O}}((\kappa\sqrt{K})^{-1})$ suboptimality for every task, where κ measures exploration coverage of the feature directions.* ***(ii)*** *Yet some such instance has an optimal policy that no classifier policy can express when f has degree below $|\mathcal{A}|$ in the task feature, regardless of K.*

In the construction for **(ii)**, the reward remains bilinear, but the optimal action switches $2|\mathcal{A}|$ times as task changes; a polynomial score of degree below $|\mathcal{A}|$ cannot change sign that often. This is a representational gap, not a sample-size gap. The planner learns a description of the dynamics and reward that does not grow with $|\mathcal{A}|$, then solves the argmax at test time; a policy must encode that argmax directly (Appendix 3).

4.2 Error Analysis for WAP

For one planning step, let a^* maximize the action value Q, and let $\hat{a}$ denote WAP's selected action. Under local regularity of Q and a covering condition on the candidates, Appendix B.5 gives

$$Q(a^*) - Q(\hat{a}) \lesssim \underbrace{h^{\text{local}} N^{-1/d_a}}_{\text{local resolution}} + \underbrace{\left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}\right]_+}_{\text{proposal out of reach}} + \underbrace{\epsilon_{\texttt{Agent.Eval}} + \epsilon_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}}_{\text{component errors}},$$

where h^{global} and h^{local} are the global-correction and local-search radii, and the N candidates occupy a d_a-dimensional goal space. The ϵ terms denote component errors, and this is an error decomposition rather than a performance guarantee; Appendix B.5 shows that no term can be dropped from a general bound over the assumption class.

Three design consequences follow. First, resolution improves only as N^{-1/d_a}, motivating WAP's search in a low-dimensional goal space rather than over raw action chunks. Second, the proposal term vanishes once $\epsilon_{\texttt{Agent.Proposal}} \le h^{\text{global}} + h^{\text{local}}$ so the VLM proposal need only lie within reach of the two search stages. Third, N does not enter the component errors, so the bound stops improving once they dominate, which is a statement about the bound rather than a proven error floor for the deployed system. Appendix D.1.2 accordingly tests a necessary condition, whether imagined rollouts preserve the agent's candidate ranking, rather than estimating those errors.

5 Experiments

Here, we provide comprehensive experiments of both action-conditioned world modeling with our robot pose-image conditioning in Section 5.1, and generalizable robot planning with our World Action Planner in Section 5.2. Experiment details and qualitative examples are in Appendix D.

5.1 Action-Conditioned World Modeling

We evaluate our pose-image conditioned world model by addressing three primary questions: **Q1:** Can we accurately generate multi-view future images conditioned on robot pose-images? **Q2:** Does pose-image conditioning improve generalization compared with low-dimensional action conditioning approaches? **Q3:** Can the model learn dynamics across diverse robot embodiments and action spaces?

Setup. We validate our approach across four simulation suites: LIBERO (Liu et al., 2023), Robocasa (Nasiriany et al., 2024a), MimicGen (Mandlekar et al., 2023), and DexMimicGen (Jiang et al., 2025).

Table 1: **Results for single-embodiment action-conditioned world modeling.** Each cell reports PSNR↑ / LPIPS↓ averaged across the predicted frames. Methods are compared on two in-distribution suites (LIBERO-90, DexMimicGen) and three generalization settings: zero-shot transfer (LIBERO-Long, LIBERO-Spatial) and few-shot finetuning on new hardware (MimicGen-Robot). Best in **bold**, second best underlined.

Method	LIBERO-90		DexMimicGen		LIBERO-Long		LIBERO-Spatial		MimicGen-Robot	
	wrist	third	wrist	third	wrist	third	wrist	third	wrist	third
WPE	15.01/.339	19.05/.121	15.52/.298	19.38/.142	14.05/.364	17.97/.157	14.72/.379	19.50/.156	16.38/.193	17.10/.130
IRA-Sim	14.94/.343	19.73/.122	15.43/.293	18.42/.142	14.00/.366	17.94/.160	14.76/.384	19.52/.153	16.37/.195	17.15/.130
Ctrl-World vel.[1]	15.54/.318	20.10/.109	15.56/.276	19.19/.135	14.63/.345	18.26/.139	15.50/.361	19.76/.137	16.80/.171	17.43/.124
Ctrl-World pos.	15.56/.317	20.13/.108	15.55/.277	19.18/.136	14.61/.348	18.27/.138	14.23/.403	19.28/.154	16.82/.169	17.44/.126
Ours	**17.02/.286**	**23.13/.085**	**16.33/.266**	**21.18/.112**	**15.98/.320**	**22.14/.093**	**16.75/.322**	**22.79/.097**	**18.02/.144**	**21.40/.095**

To generalize to exploratory actions, we perturb demonstration trajectories with Gaussian noise following the MPPI algorithm (Williams et al., 2017), a standard for world-model-based control (Hansen et al., 2023; Jain et al., 2025). Each task uses 100 training trajectories (including perturbed versions) and 10 held-out evaluation trajectories. We employ `Wan`-T2V-1.3B (Wan et al., 2025) as the backbone, finetuning for 10K steps with a global batch size of 64. Models process 21 history frames at 7 FPS to predict 20 future frames at 20 FPS. Input consists of four camera views concatenated into a 2×2 grid (224px per view). Details in Appendix D.1.1.

Baselines and Evaluation. We compare against state-of-the-art diffusion-based world models: WPE (Quevedo et al., 2025) and IRA-Sim (Zhu et al., 2025), which utilize AdaLN-Zero modulation (Peebles & Xie, 2023) for the action embeddings, and Ctrl-World (Guo et al., 2025) employing cross-attention with low-dimensional action tokens. For cross-embodiment methods, we adapt VLA architectures including unified action spaces (Liu et al., 2024b), embodiment-aware encoders (Bjorck et al., 2025), and soft prompting (Zheng et al., 2025). All baselines use the same `Wan`-T2V-1.3B backbone and model configurations, but are trained for 20K steps (twice our method's duration) to ensure a competitive comparison. Evaluation metrics include LPIPS (Zhang et al., 2018) and PSNR (Hore & Ziou, 2010), as they best align with human preferences for controllable world modeling (Zhu et al., 2025).

We first evaluate in-distribution prediction (**Q1**) with a 7-DoF Franka-Panda (LIBERO-90) and a bi-manual dexterous robot (DexMimicGen), achieving a consistent average 11.3% relative improvement in PSNR and LPIPS over the strongest baseline in each column (Table 1). To assess generalization (**Q2**), we conduct zero-shot evaluation on LIBERO-Long and LIBERO-Spatial tasks, alongside few-shot finetuning on distinct hardware (Sawyer, IIWA, and UR5e) within the MimicGen-Robot, using the LIBERO-90 model. Across these novel trajectories, our pose-image conditioning yields a 16.8% average relative improvement on the same measure, confirming its superior generalization capability. Qualitative results in Appendix D.1.2.

Cross-Embodiment Modeling (Q3). To evaluate cross-embodiment world modeling, we train on a heterogeneous mixture of RoboCasa, MimicGen, and DexMimicGen, spanning action dimensions of 7 (fixed arm), 12 (mobile Panda), 14 (bi-manual parallel grippers), and 24 (bi-manual dexterous hands). As shown in Table 2, our method outperforms prior approaches for cross-embodiment training, suggesting that representing actions via robot pose images provides an embodiment-agnostic interface that may be of broader utility for future robot models.

Table 2: **Results for cross-embodiment modeling.** We use a mixture of trajectories from multiple robots with distinct action spaces. Our method consistently outperforms existing approaches. Each cell represents PSNR↑ / LPIPS↓.

Method	wrist-view	third-view
Unified action space	14.60 / 0.323	17.80 / 0.164
Embodiment-aware encoder	14.68 / 0.321	17.52 / 0.168
Soft prompt	14.66 / 0.323	17.83 / 0.162
Ours	**15.11 / 0.308**	**19.44 / 0.131**

5.2 World Action Planner

In this section, we evaluate our World Action Planner in three generalization scenarios: *compositional task generalization*, *new layout generalization*, and *novel task generalization*. Details including baseline training and in-distribution performance report, method details, additional results and visualizations, and failure analysis are provided in Appendix D.2.

[1]The original implementation in Ctrl-World (Guo et al., 2025) uses end-effector position control, and all other baselines use end-effector velocity control.

Table 3: **Average success rate (%) across generalization scenarios.** Results are averaged over 5 tasks in compositional task generalization, 6 tasks in new-layout generalization, and 2 tasks in novel task generalization, with 50 trials per task, and over 2 real robot novel tasks with 20 trials per task.

Method	Compositional task	New layout	Novel task	Real robot novel task
$\pi_{0.5}$	9.6	1.7	10.0	0.0
cosmos-policy	2.4	0.0	2.0	0.0
SAILOR	16.4	3.7	6.0	0.0
GPC-RANK	12.0	2.7	8.0	0.0
DREAMSTEER	20.0	10.7	18.0	5.0
Vision-language planner	42.0	37.7	40.0	10.0
World action planner (ours)	**72.0**	**82.0**	**78.0**	**65.0**

Setup and Baselines. For our method, We employ Gemini 3.0 Flash (Pichai et al., 2025) as our default VLM agent, and we train our optional diffusion policy with the same or substantially less expert data compared with baseline methods, detailed in each scenario below. For imitation learning baselines, we use SOTA VLA $\pi_{0.5}$ (Intelligence et al., 2025) and WAM cosmos-policy (Kim et al., 2026), where we tune the baseline setups to enable *superior in-distribution performance*, as detailed in Appendix D.2. We also compare against prior world-model-based MPC methods that sample actions from SOTA VLAs and optimize them. Specifically, we adapt SAILOR (Jain et al., 2025) and GPC-RANK (Qi et al., 2025) using the same world model and VLM-based evaluation as WAP, and for DREAMSTEER (Cui et al., 2026), we use its original VLAC critic, with action proposals sampled from the finetuned SOTA VLAs as the generalist policy baselines. To isolate the impact of planning with the action-conditioned world model, we compare against a vision-language planner baseline that is identical to WAP with the same VLM agent and policy with additional reasoning without imagination, but executes the actions directly without optimization and search over imagined rollouts.

Compositional Task Generalization. In real-world applications, robot policies are often required to generalize to compositional, long-horizon tasks. For example, a robot may be trained to pick and place individual objects, while a task such as tidying a room requires it to execute multiple such placements in a continuous sequence. We select five tasks from LIBERO-Long composed by pairs of tasks from the LIBERO-90 suite, and train our diffusion policy and world model exclusively on the corresponding atomic LIBERO-90 tasks, following the same training setting as the $\pi_{0.5}$ and Cosmos-Policy baselines. When transferring to these newly composed tasks, we observe a distinct failure mode in end-to-end VLA models: although they often complete the first sub-task successfully, they stagnate immediately afterward and produce near no-op actions. This behavior arises because the policy must transition from the terminal state of the first sub-task to the initial configuration required by the next, yet such transitions are absent from the training demonstrations. Our World Action Planner overcomes this limitation by using the world model to plan transition motions that move the robot into states from which the policy can successfully complete the subsequent sub-task. Additional details are provided in Appendix D.2.1, with per-task results in Table 11 and qualitative examples in Fig. 7.

New Layout Generalization. A generalizable robot must adapt to novel object configurations at test time despite training on a limited set of layouts. We evaluate this capability on six LIBERO-Object tasks with substantial test-time shifts in both target and distractor positions. Using the official $\pi_{0.5}$ and Cosmos-Policy checkpoints as baselines, we observe that generalist VLAs and WAMs often fail to ground instructions in the modified scene and instead fall back to motion priors from the training layout. This leads to failures such as grasping distractors or repeatedly attempting to grasp empty space near the original target coordinates (Fig. 10, 11). For WAP, we finetune the diffusion policy and world model using demonstrations from the *original layout only*. To highlight sample efficiency, we train the policy with just *5 expert demonstrations* per task, compared with more than 40 used by the baseline checkpoints. Although this few-shot policy fails end-to-end, it remains effective for local grasping under modified layouts. WAP uses the world-model planner to navigate to suitable manipulation states and plan grasping and placing motions, decoupling local manipulation from global motion planning. This enables successful task completion with substantially less expert data. Additional details are provided in Appendix D.2.2, with per-task results in Table 12 and qualitative examples in Fig. 8.

Novel Task Generalization. We evaluate WAP without a learned task policy or expert demonstrations on the `PickPlaceCan` and `StackCube` tasks from Robosuite (Zhu et al., 2020), using only hard-coded gripper-closing and release primitives. At test time, we train the world model on 50 exploratory trajectories collected by the VLM in the target environment. We evaluate policy-based

baselines both without target-environment finetuning and after finetuning on the same 50 trajectories; Table 3 reports the later, while Table 13 reports both regimes. WAP performs best in both, with the largest margin on the precision-intensive `StackCube` task, showing the benefit of refining coarse VLM proposals through action-conditioned imagination. Additional details are provided in Appendix D.2.3, with qualitative examples in Fig. 12.

Real Robot Novel Task Generalization. We evaluate WAP on a real Franka Panda robot for placing either a green or purple can into a white basket. Following the novel task setup above, we train the world model at test time using 10 *failed* explorations of placing the green can using the VLM and use them to finetune our world model. The resulting real-robot execution is shown in Fig. 4. As shown in Fig. 16, our world model is able to accurately imagine O.O.D successful placement maneuvers never seen in training. Additional details are provided in Appendix D.2.4, with per-task results in Table 14.

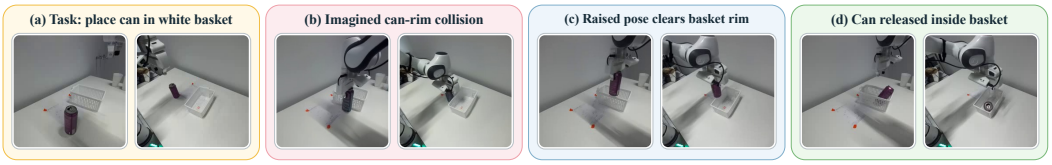


Figure 4: **Real-robot novel task generalization.** To place the can into the white basket, WAP uses world-model imagination to anticipate a potential collision between the can and basket rim. Global action optimization then lifts the gripper to create sufficient clearance, enabling successful placement.

5.3 Planning Efficiency

We directly compare the planning sample efficiency of our coarse-to-fine action search framework, which combines global optimization with local grid search, against prior MPC method VLMPC (Zhao et al., 2024). To isolate the efficiency of the planning protocol from world-model prediction errors, we provide the baselines with *ground-truth* simulator rollouts instead of the world model imagined trajectories, and we use the same VLM agent to provide the reward. We use the same rollout length and same number of VLM calls for the baseline so runtime is determined by the total number of samples. As shown in Table 4, our method achieves high success rates while evaluating substantially fewer candidate rollouts, demonstrating the efficiency of our structured and physically grounded planning pipeline. As illustrated in Fig. 17, global optimization can identify and avoid potential physical failures from a single imagined trajectory, whereas conventional sampling-based methods often repeat errors across multiple samples. Additional details are provided in Appendix D.3.

Table 4: **Planning sample efficiency against MPC.** Success rates (%) are averaged within each setting.

Method	Compositional task	New layout	novel task
VLMPC (8 samples)	36.0	38.3	64.0
VLMPC (16 samples)	46.0	50.0	76.0
WAP w/ global optimization only (1 sample)	58.0	66.0	65.0
WAP (6 samples)	**72.0**	**82.0**	**78.0**

5.4 Ablation Experiments

We ablate the stage-wise contributions of global optimization, local search, and policy rollout imagination in Appendix D.4.1. The results are reported in Table 16, where each stage provides gains as task complexity increases. The PnP cream cheese box task requires precise gripper rotations that are difficult to specify analytically, but can be accurately identified through local search by reasoning over imagined frames (Fig. 18). For the PnP mug task, imagining future policy rollouts is critical for identifying states from which the policy can successfully complete the grasp (Fig. 19). We also ablate the choice of the VLM agent in Appendix D.4.2 and the choice of hyperparameters in Appendix D.4.3, demonstrating the robustness of our method.

6 Conclusion and Limitations

In this work, we propose World Action Planner, an agentic robot planning system with world model to address novel tasks in generalization scenarios. Currently the planning stages are manually designed, and we left self-evolving agentic planning for future work. While world model imagination is time consuming, with wall clock time reported in Appendix E, inference acceleration and few step sampling techniques could be applied to further increase planning speed and real-time efficiency.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for assist in the writing of proofs. We have not used generative AI tools for Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims,propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, and assist with translation are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images. We have reviewed all AI-assisted work. We checked the proof line by line and read the LLM-generated code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide full training details of our method, including detailed data recipe, model architecture, hyper-parameters and environment setups. We include detailed setups on how to compute the joint pose image using simulator with robot model, and detailed setup of the diffusion video model design. For our world action planner, we disclose the prompts used and specific hyper-parameter choices, and for the baseline policies, we disclose the training setup of both VLA and WAM. Thus we ensure our method and baseline results are reproducible.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011. 22, 29

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 15

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025. 15

Hadi Alzayer, Wenlong Huang, Haonan Chen, Christopher Luey, Lvmin Zhang, Maneesh Agrawala, Gordon Wetzstein, Li Fei-Fei, Yilun Du, Jiajun Wu, and Jia-Bin Huang. Masked visual actions for unified world modeling, 2026. URL https://arxiv.org/abs/2607.19343. 15

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 2, 15

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. 7

Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025. 3, 15, 16

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 15, 45

Hanchen Cui, Sergio Arnaud, Arjun Majumdar, Daniel Dugas, Elie Aljalbout, Karthik Desingh, Krishna Murthy Jatavallabhula, and Franziska Meier. Dreamsteer: Latent world models can steer vla policies during deployment without any finetuning. *arXiv preprint arXiv:2607.02865*, 2026. 2, 8, 16, 44, 47, 49, 52

Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. *arXiv preprint arXiv:2310.10625*, 2023. 15

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024. 2, 15

Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. *arXiv preprint arXiv:2502.16707*, 2025. 2, 15

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in neural information processing systems*, 34:25502–25515, 2021. 16

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025. 3, 7, 15, 41, 42

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015. 5, 16

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022. 2, 15

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023. 2, 7, 15

Xingyi He, Hao Yu, Sida Peng, Dongli Tan, Zehong Shen, Hujun Bao, and Xiaowei Zhou. Matchanything: Universal cross-modality image matching with large-scale pre-training. *arXiv preprint arXiv:2501.07556*, 2025. 15

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010. 7

Jiachen Hu, Xiaoyu Chen, Chi Jin, Lihong Li, and Liwei Wang. Near-optimal representation learning for linear bandits and linear rl. In *International Conference on Machine Learning*, pp. 4349–4358. PMLR, 2021. 16

Pihe Hu, Yu Chen, and Longbo Huang. Nearly minimax optimal reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pp. 8971–9019. PMLR, 2022. 18

Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023. 15

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 15

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. 1, 8, 15, 16

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 57

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search. *arXiv preprint arXiv:2506.05294*, 2025. 2, 7, 8, 15, 41, 44, 47, 49, 50, 51

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025. 15

Emily Yue-Ting Jia, Weiduo Yuan, Tianheng Shi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Dreamplan: Efficient reinforcement fine-tuning of vision-language planners via video world models. *arXiv preprint arXiv:2603.16860*, 2026. 2, 15

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930. IEEE, 2025. 6

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pp. 4870–4879. PMLR, 2020. 18

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. *Mathematics of Operations Research*, 48(3):1496–1521, 2023. 6, 22, 29

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 15, 46

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. 8, 15, 45

Fang Kong, Xiangcheng Zhang, Baoxiang Wang, and Shuai Li. Improved regret bounds for linear adversarial mdps via linear optimization. *arXiv preprint arXiv:2302.06834*, 2023. 23

Jeongyeol Kwon, Yonathan Efroni, Constantine Caramanis, and Shie Mannor. Reward-mixing mdps with few latent contexts are learnable. In *International Conference on Machine Learning*, pp. 18057–18082. PMLR, 2023. 16

Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Sample complexity of asynchronous q-learning: Sharper analysis and variance reduction. *Advances in neural information processing systems*, 33:7031–7043, 2020. 23

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. 15

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024. 57

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 3

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. 6, 45

Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. Moka: Open-world robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024a. 2, 15

Haowen Liu, Shaoxiong Yao, Haonan Chen, Jiawei Gao, Jiayuan Mao, Jia-Bin Huang, and Yilun Du. Simpact: Simulation-enabled action planning using vision-language models. *arXiv preprint arXiv:2512.05955*, 2025. 16

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024b. 7

Rui Lu, Gao Huang, and Simon S Du. On the power of multitask representation learning in linear mdp. *arXiv preprint arXiv:2106.08053*, 2021. 16

Dhruv Malik, Yuanzhi Li, and Pradeep Ravikumar. When is generalizable reinforcement learning tractable? *Advances in Neural Information Processing Systems*, 34:8032–8045, 2021. 16

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023. 6

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024a. 6

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024b. 15

Chuanruo Ning, Kuan Fang, and Wei-Chiu Ma. Prompting with the future: Open-world model predictive control with interactive digital twins. *arXiv preprint arXiv:2506.13761*, 2025. 2, 16, 54

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023. 7

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. A new era of intelligence with gemini 3. *Google Blog, November*, 18, 2025. 8

Han Qi, Haocheng Yin, Yilun Du, and Heng Yang. Strengthening generative robot policies through predictive world modeling. *arXiv e-prints*, pp. arXiv–2502, 2025. 8, 44, 47, 49, 50, 51

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation. *arXiv preprint arXiv:2506.00613*, 2025. 3, 7, 15

Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33:2914–2924, 2020. 16, 19, 20

Nived Rajaraman, Yanjun Han, Lin Yang, Jingbo Liu, Jiantao Jiao, and Kannan Ramchandran. On the value of interaction and function approximation in imitation learning. *Advances in Neural Information Processing Systems*, 34:1325–1336, 2021. 6, 16, 27

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. World-gymnast: Training robots with reinforcement learning in a world model. *arXiv preprint arXiv:2602.02454*, 2026. 2, 15

Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025. 3, 40

Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on learning theory*, pp. 2898–2933. PMLR, 2019. 16

Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. Ominicontrol: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14940–14950, 2025. 3

Gemini Robotics Team, Krzysztof Choromanski, Coline Devin, Yilun Du, Debidatta Dwibedi, Ruiqi Gao, Abhishek Jindal, Thomas Kipf, Sean Kirmani, Isabel Leal, et al. Evaluating gemini robotics policies in a veo world simulator. *arXiv preprint arXiv:2512.10675*, 2025. 15

Andrew J Wagenmaker, Yifang Chen, Max Simchowitz, Simon Du, and Kevin Jamieson. Reward-free rl is no harder than reward-aware rl in linear markov decision processes. In *International Conference on Machine Learning*, pp. 22430–22456. PMLR, 2022. 23

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 7

Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12713–12724, 2025. 15

Max Weltevrede, Caroline Horsch, Matthijs TJ Spaan, and Wendelin Böhmer. Exploration implies data augmentation: Reachability and generalisation in contextual mdps. *arXiv preprint arXiv:2410.03565*, 2024. 16

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025. 40

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017. 7

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models. *Advances in Neural Information Processing Systems*, 37:68082–68119, 2024. 15

Haotian Ye, Xiaoyu Chen, Liwei Wang, and Simon Shaolei Du. On the power of pre-training for generalization in rl: provable benefits and hardness. In *International Conference on Machine Learning*, pp. 39770–39800. PMLR, 2023. 16

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. 1, 15

Kenny Young, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. The benefits of model-based generalization in reinforcement learning. *arXiv preprint arXiv:2211.02222*, 2022. 16

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023. 15

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018. 7

Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. *arXiv preprint arXiv:2505.23614*, 2025. 41

Wentao Zhao, Jiaming Chen, Ziyu Meng, Donghui Mao, Ran Song, and Wei Zhang. Vlmpc: Vision-language model predictive control for robotic manipulation. *arXiv preprint arXiv:2407.09829*, 2024. 2, 9

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025. 7

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9834–9844, 2025. 3, 7, 15

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. 8, 50

Appendix Overview. First, we provide a comprehensive review of related works in Appendix A. Proof of theoretical results is provided in Appendix B. Then we provide the details of our method in Appendix C and details of the experiments in Appendix D. We report the wall clock time of our world model in Appendix E.

A Additional Related Works

Controllable World Models. To generate and predict future states given control inputs, foundational video generative models typically condition on textual and visual descriptions (Ali et al., 2025; Agarwal et al., 2025). To apply such controllable generation to robot manipulation, most existing works condition on low-dimensional action vectors through AdaLN-Zero modulation (Quevedo et al., 2025; Zhu et al., 2025) in diffusion models, or interleave low-dimensional action tokens and visual state tokens in auto-regressive models (Wu et al., 2024; Assran et al., 2025). Recently, Guo et al. (2025) proposed cross-attention between frames and action tokens for improved control, while adopting multi-view prediction to enhance consistency. Furthermore, Team et al. (2025) demonstrated that pose-conditioned robot world models can successfully simulate OOD scenarios and unsafe executions, thereby serving as a tool to evaluate robot policies. Wang et al. (2025) also adopted pose-image conditioning via the ControlNet (Zhang et al., 2023) mechanism; however, they extract the ground truth robot poses from future frames using image matching (He et al., 2025), while we compute the poses using forward dynamics to imagine different actions during test time planning. Concurrently, Jia et al. (2026); Alzayer et al. (2026) proposed rendering future arm images corresponding to actions as direct inputs to the video diffusion model.

World Models for Decision Making. Recent works (Sharma et al., 2026; Jia et al., 2026) have conducted reinforcement learning (RL) on generalist VLAs and VLM planners, respectively, demonstrating that RL within world models can enhance performance on physical robots. To plan using world models, MPC-based methods (Hansen et al., 2022; 2023; Jain et al., 2025) utilize the MPPI algorithm for exploration and update the sampling distribution based on Q-functions computed via the dynamics model. On the other hand, video planners (Du et al., 2023; Jang et al., 2025; Chen et al., 2025) generate synthetic task completion videos using video generation models, and then extract the control actions using vision-based inverse-dynamics models or pose estimation techniques.

VLMs for Robot Decision Making. To leverage the extensive world knowledge and reasoning capabilities of VLMs, the majority of existing works utilize these models for high-level task planning (Duan et al., 2024; Liu et al., 2024a; Huang et al., 2023; Hu et al., 2023). However, to ground executable actions in the physical world, these methods typically rely on manually engineered or sampling-based manipulation techniques to generate low-level actions—such as grasp synthesis (Liu et al., 2024a; Duan et al., 2024), volumetric value maps (Huang et al., 2023), motion arrows (Nasiriany et al., 2024b), or teleoperation and kinesthetic teaching (Hu et al., 2023). In contrast, our method leverages highly-expressive generative policies (Chi et al., 2025) as modular tools to execute atomic skills. By grounding the low-level actions proposed by the VLM through our fine-grained, pose-image conditioned world model, we enable the VLM to synthesize and reason directly over low-level executable actions.

VLMs and World Models for Imitation Learning Policies. Recent attempts have adopted pretrained VLMs or video world models as backbones for imitation learning policies, such as vision-language-action models (VLA) (Intelligence et al., 2025; Kim et al., 2024) and world-action-models (WAM) (Ye et al., 2026; Kim et al., 2026; Li et al., 2026). These methods finetune action heads on top of the VLM or video generation model backbone to generate future actions, where OpenVLA (Kim et al., 2024) adopts auto-regressive action generation and $\pi_{0.5}$ (Intelligence et al., 2025) further incorporates diffusion-based action expert that denoises the action chunk. To adapt video generation models for policies, Ye et al. (2026); Kim et al. (2026); Li et al. (2026) first generates future frames and then generates the actions auto-regressively with the action head. However, these approaches often necessitate substantial quantities of high-quality action-labeled demonstration data, and their generalization performance can remain constrained, as observed in our experimental evaluation.

MPC with VLMs or World Models. More closely related to our work, we discuss methods that employ MPC for general robot decision-making with foundation models. Feng et al. (2025) iteratively improves VLM action proposals using imagination from a diffusion generative model, but requires in-domain finetuning of both the VLM and the world model and operates on abstract semantic textual

actions rather than precise robot control, which makes it impossible to directly operate robots so we do not include a direct comparison. Ning et al. (2025) uses digital twins built with physical simulators, rather than learned world models, for imagination and adopts CEM-based action sampling, which has lower sample efficiency than our physically grounded search framework. Liu et al. (2025) constructs a simulation of the task environment using the VLM and iteratively optimizes the action inside simulation, however, there precision is limited by the simulation construction capabilities of the VLM. Concurrent work by Cui et al. (2026) improves a base VLA policy through imagination with an action-conditioned world model by selecting among K action candidates sampled from the VLA. In contrast, we employ a more efficient search strategy and enable generalization to new environments beyond the training distribution of the base VLA.

B Theoretical Insights

Here we provide the theoretical results which demonstrates the benefits of model-based learning with exploration over imitation learning (Rajaraman et al., 2020; 2021) in multi-task generalization settings, where different tasks share the same dynamics but have different rewards (Kwon et al., 2023). We focus on tabular MDPs and representation learning with linear function approximation. High level speaking, we show that model-based exploration can leverage data collected across all tasks, while imitation learning can only learn from the corresponding task expert, thus suffering from sub-optimal sample complexity.

B.1 Literature Overview

B.1.1 Separation Between Model-Based and Model-Free Learning

Prior research has demonstrated that model-free methods can exhibit exponentially worse sample complexity compared to model-based methods (Sun et al., 2019) for specific classes of MDPs. However, these findings typically rely on strong structural assumptions regarding the model class; in the standard tabular setting, there is no information-theoretic gap between model-based and model-free approaches (Sun et al., 2019; Young et al., 2022). In our analysis, we show that within the tabular setting and under linear function approximation, multi-task imitation learning can be significantly more challenging than model-based planning. This stands in contrast to single-task settings, where imitation learning is generally more sample-efficient.

B.1.2 Generalization in Reinforcement Learning

In task-agnostic settings, where the agent lacks knowledge of the test environment's context and is evaluated on its *average performance* across a family of MDPs, Malik et al. (2021) established that MDPs must share certain similarities for a policy to generalize effectively. Furthermore, Ye et al. (2023) derived average optimality gaps comparable to those in supervised learning without online interaction, and demonstrating that pretraining can significantly improve sample complexity when subsequent online interactions are permitted in the test environment. Also, Ghosh et al. (2021); Malik et al. (2021); Ye et al. (2023) have shown that such task agnostic policies can not achieve instance optimality in new environment without further interaction or history dependence. In our framework, we assume the context of each environment is provided to the learner, and we aim for *instance optimality* without the need for online interactions during testing. Regarding multi-task representation learning, Lu et al. (2021); Hu et al. (2021) showed that learning a shared representation across tasks can improve sample complexity. We further assume that task-dependent context features are known to the learner and use them to construct context-dependent policies. Our setting is more aligned with modern robotic practices where task instructions serve as inputs to a generalist policy, and we derive representations for the tasks using pretrained models (Chen et al., 2025; Intelligence et al., 2025).

B.2 Setup

We consider multi-task learning as contextual MDPs (Hallak et al., 2015; Weltevrede et al., 2024) where MDPs with different contexts have shared dynamics and different rewards. Each MDP is described by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \rho, H)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and

H is the horizon. The initial state s_1 is drawn from the initial state distribution $\rho \in \Delta(\mathcal{S})$, and the state transition is governed by the transition probability $\mathbb{P}$. At each time step $h \in [H]$, the agent chooses action $a_h \in \mathcal{A}$ in state s_h according to policy $\pi(s_h) \in \Delta(\mathcal{A})$, receives reward $r_h(s_h, a_h)$ and transitions to the next state according to the transition kernel $\mathbb{P}_h(s_{h+1}|s_h, a_h)$. The episode terminates after H steps. Define the state-action value function (Q-function) and the state value function as the expected cumulative reward of π:

$$Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \,\middle|\, s_h = s, a_h = a, \pi\right],$$
$$V_h^\pi(s) = \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}) \,\middle|\, s_h = s, \pi\right],$$

with the convention $V_{H+1}^\pi \equiv 0$. Both are maps into $[0, H]$ and satisfy the Bellman equations

$$Q_h^\pi(s,a) = r_h(s,a) + \sum_{s' \in \mathcal{S}} \mathbb{P}_h(s'|s,a)\, V_{h+1}^\pi(s'), \quad V_h^\pi(s) = \mathbb{E}_{a \sim \pi_h(s)}\left[Q_h^\pi(s,a)\right].$$

Both sides of each identity are real numbers: Q_h^π and V_h^π are value functions, not actions. The goal of the agent is to achieve maximum expected cumulative reward $\mathbb{E}_{s_1 \sim \rho} V_1^\pi(s_1)$. Since the action space and episode length is finite, there always exist an optimal Q-function $Q_h^*(s,a) = \sup_\pi Q_h^\pi(s,a)$ and value function $V_h^*(s) = \sup_\pi V_h^\pi(s)$ for all s, a that satisfies the Bellman optimality equation:

$$Q_h^*(s,a) = r_h(s,a) + \sum_{s' \in \mathcal{S}} \mathbb{P}_h(s'|s,a) V_{h+1}^*(s'), \quad V_h^*(s) = \max_{a \in \mathcal{A}} Q_h^*(s,a), \quad V_{H+1}^* \equiv 0.$$

Note that the second identity takes the *maximum* rather than the argmax: $V_h^*(s) \in [0, H]$ is a value, whereas $\operatorname{argmax}_a Q_h^*(s,a) \in \mathcal{A}$ is an action. The two are related through the greedy policy: there exists a deterministic policy $\pi_h^*(s) = \delta\left(\operatorname{argmax}_a Q_h^*(s,a)\right)$ that is optimal with $Q_h^{\pi^*} = Q_h^*$ and $V_h^{\pi^*} = V_h^*$. When the policy is deterministic, we abuse the notation and denote $a = \pi(s)$.

In the multi-task setting, we have a set of contexts (tasks) $\mathcal{C}$, with different reward functions $\{r_c : \mathcal{S} \times \mathcal{A} \to [0,1], c \in \mathcal{C}\}$, and the MDP associated with context c is $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r_c, \rho, H)$. We assume that the context c is determined before the episode and revealed to the agent. We then denote the policy and value functions given context c as $\pi_c(s)$, $Q_h^\pi(s,a;c)$ and $V_h^\pi(s;c)$ respectively. Since the context is known to the agent, we aim to obtain a context dependent policy $\pi_c(s)$ that maximizes $\mathbb{E}_{s_1 \sim \rho} V_1^\pi(s_1; c)$ for all context $c \in \mathcal{C}$. When rewards for different contexts share the same function approximation, we abuse notation and write $r(s,a;c) = r_c(s,a)$; similarly when policies for different contexts are from the same class, we write $\pi(s;c) = \pi_c(s)$.

B.2.1 Interaction Protocol

In this work we investigate the generalization task, where the agent is able to explore the environment and collect data during training time, across contexts of its own choosing. At test time, the agent is evaluated by the performance of its policy under a randomly selected context. For the imitation learning setup, we further assume the existence of expert policies to collect demonstrations for any given context. Denote the total rounds of episode during the training phase as K.

Task Schedule. Before data collection begins, an agent commits to a *task schedule*: non-negative integers $\{n_c\}_{c \in \mathcal{C}}$ with $\sum_{c \in \mathcal{C}} n_c = K$, giving the number of episodes it spends on each context. The schedule may depend on $\mathcal{C}$, on the feature maps where those are available, and on the agent's own randomness, but not on the data, which has not been observed yet. This is how multi-task datasets are assembled in practice, either as a fixed number of demonstrations per task or by i.i.d. task sampling.

We impose the schedule on *both* agents, so neither is advantaged by it, and it costs the model-based agent nothing in either setting we study: the tabular algorithm explores reward-free and therefore never consults the context, and the linear algorithm fixes its contexts through a G-optimal design computed from the known task features (Theorem 4). Within its schedule the model-based agent still chooses each exploration policy π_k adaptively from the data collected so far. We specify the two protocols below.

Imitation Learning Agent. At each episode $k \in [K]$, for the context $c_k \in \mathcal{C}$ assigned by its schedule, the imitation learning agent queries the deterministic expert policy $\pi^*_{c_k}$ to rollout in the environment $\mathcal{M}_{c_k}$ and collect data $\tau_{k,c_k} = \left\{(s_h^k, a_h^k)\right\}_{h=1}^H$, where $a_h^k = \pi^*_{c_k}(s_h^k)$. We denote the total data collected as $\mathcal{D} = \{\tau_{k,c_k}\}_{k=1}^K$. After the data collection phase, the learner outputs a context dependent policy $\widehat{\pi} = \{\widehat{\pi}_c, c \in \mathcal{C}\}$ minimizing the empirical risk of π with respect to $\mathcal{D}$ in a policy class Π:

$$\widehat{\pi} = \operatorname*{argmin}_{\pi \in \Pi} \Pr_{(s,a,c)\sim\mathcal{D}} \left(\pi_c(s) \neq a\right) ,$$

where we break ties arbitrarily.

Model-based Agent. At each episode $k \in [K]$, for the context $c_k \in \mathcal{C}$ assigned by its schedule, the agent may explore with any policy π_k, itself chosen from the data collected so far, to collect the trajectory $\tau_{k,c_k} = \left\{(s_h^k, a_h^k, r_h^k)\right\}_{h=1}^H$ in $\mathcal{M}_{c_k}$, where $a_h^k \sim \pi_k(s_h^k)$ and $r_h^k \sim r_c(s_h^k, a_h^k)$. After collecting the dataset $\mathcal{D} = \{\tau_{k,c_k}\}_{k=1}^K$, the learner outputs an estimate of the model $\widehat{\mathbb{P}}$, and if the reward is unknown, an estimate of the reward $\widehat{r} = \{\widehat{r}_c, c \in \mathcal{C}\}$. At test time, given the random context c, the agent adopts a planner that outputs the optimal policy $\widehat{\pi}_c$ with estimated MDP $\widehat{\mathcal{M}}_c = \left(\mathcal{S}, \mathcal{A}, \widehat{\mathbb{P}}, \widehat{r}_c, \rho, H\right)$.

Information Available to Each Learner. The two protocols above are not two algorithms fed the same observations. They are two different *interfaces* onto the environment, and what we compare is the interfaces. Spelling out what each learner gets to see therefore keeps the source of the separation in plain sight.

- The imitation learner observes state–action pairs (s_h^k, a_h^k) with $a_h^k = \pi^*_{c_k}(s_h^k)$. It never observes a reward, and it is not given the reward functions $\{r_c\}$. This is the defining feature of imitation learning: the task is specified only through expert behavior.
- The model-based learner observes transitions $(s_h^k, a_h^k, s_{h+1}^k)$ and reward samples r_h^k, and it can evaluate the reward of *any* task at planning time — in the tabular setting of Section B.3 because $\{r_c\}$ is given, and in the linear setting of Section B.4 because the shared reward representation $\{\Theta_h\}$ is estimated from data. It never observes an expert action.

Neither interface contains the other. The imitation learner is handed optimal action labels the model-based learner never sees, and the model-based learner is handed reward information the imitation learner never sees. Our claim is that the second is the more valuable of the two once the tasks are many, for the simple reason that the ability to score a reward carries over to a new task while an expert label does not. Remark 2 pins down which half of the asymmetry is responsible for which result, and shows the tabular separation surviving even after the dynamics information is leveled between the two.

Suboptimality Gap. We define the suboptimality gap of the context dependent policy class $\pi = \{\pi_c, c \in \mathcal{C}\}$ returned by the agent as:

$$\mathbb{E}_{c\sim\mathrm{Unif}(\mathcal{C}),\ s_1\sim\rho}\left[V^*(s_1; c) - V^{\pi_c}(s_1; c)\right] . \tag{1}$$

B.3 Tabular MDPs

Here we consider tabular MDPs with finite state and action spaces and finite context set. We assume the reward function for each context is known to the agent and the transition is unknown (Hu et al., 2022). We denote the number of states, actions and contexts as $|\mathcal{S}|$, $|\mathcal{A}|$ and $|\mathcal{C}|$.

B.3.1 Model-based Planning Agent

Here we prove Part **(i)** of Theorem 1. The result for the model-based exploration agent when generalizing to new rewards comes immediately with the reward-free exploration result of (Jin et al., 2020).

Theorem 3 (Theorem 3.1 of Jin et al. (2020))**.** *Fix $\epsilon \in (0, H]$ and $\delta \in (0, 1)$. There exists a reward-free model-based exploration algorithm which, with probability at least $1 - \delta$, outputs a model estimate $\widehat{\mathbb{P}}$ such that* simultaneously *for an arbitrary collection of known reward functions*

$\{r_c : \mathcal{S} \times \mathcal{A} \to [0,1]\}$, *the optimal policy* $\widehat{\pi}_c$ *of the estimated MDP* $\widehat{\mathcal{M}}_c = \left(\mathcal{S}, \mathcal{A}, \widehat{\mathbb{P}}, r_c, \rho, H\right)$ *satisfies* $V^*_{\mathcal{M}_c} - V^{\widehat{\pi}_c}_{\mathcal{M}_c} \le \epsilon$. *The number of exploration episodes is* $\tilde{\mathcal{O}}\left(\frac{H^5|\mathcal{S}|^2|\mathcal{A}|}{\epsilon^2}\right)$, *where* $\tilde{\mathcal{O}}$ *hides logarithmic factors in* $|\mathcal{S}|, |\mathcal{A}|, H$ *and* $1/\delta$.

Inverting the sample complexity of Theorem 3 gives Part **(i)**: after K exploration episodes the model-based planner attains, for *every* task simultaneously,

$$V^*_{\mathcal{M}_c} - V^{\widehat{\pi}_c}_{\mathcal{M}_c} \le \tilde{\mathcal{O}}\left(\min\left\{H, \sqrt{\frac{H^5|\mathcal{S}|^2|\mathcal{A}|}{K}}\right\}\right), \quad \forall c \in \mathcal{C}\,, \tag{2}$$

where the min with H records that no bound can be informative below the trivial range of the value function, so the rate $K^{-1/2}$ is only meaningful once $K \gtrsim H^3|\mathcal{S}|^2|\mathcal{A}|$. The bound is independent of the number of tasks $|\mathcal{C}|$, because the exploration phase is reward-free and its guarantee holds uniformly over reward functions.

B.3.2 Imitation Learning Agent

Here we provide the hardness result for imitation learning in Part **(ii)** of Theorem 1. High level speaking, since the expert policies differ across tasks, the average number of demonstrations available for each task is $\frac{K}{|\mathcal{C}|}$, and the suboptimality gap inherits the $\Omega\left(1/N\right)$ lower bound of Rajaraman et al. (2020) with $N = \frac{K}{|\mathcal{C}|}$. Stated that baldly the result is not yet correct, for two reasons that also happen to determine when the separation matters.

Regime and Clipping. The first is a ceiling. No policy can be worse than H, so no lower bound may exceed H either, and the bound has to saturate somewhere. The second concerns the construction. The hard instance of Rajaraman et al. (2020) spreads mass ξ over $|\mathcal{S}|-1$ "light" initial states, which needs $(|\mathcal{S}|-1)\xi \le 1$. Setting $\xi = \frac{1}{\bar{N}+1}$ outright breaks this as soon as $\bar{N} < |\mathcal{S}|-2$, leaving a signed measure in place of a distribution, and it breaks it precisely in the data-scarce regime the bound is meant to describe. We therefore define

$$\xi = \min\left\{\frac{1}{\bar{N}+1}, \frac{1}{|\mathcal{S}|-1}\right\}, \qquad \bar{N} = \frac{K}{|\mathcal{C}|}\,, \tag{3}$$

which is a valid construction in both regimes and yields a lower bound that is clipped at H, as it must be.

We design the hard MDP following (Rajaraman et al., 2020, Sec. A.4.1). For each state $s \in \mathcal{S}$, it is an absorbing state where $\mathbb{P}(\cdot|s,a) = \delta(s)$, $\forall a \in \mathcal{A}$. With ξ defined in (3), we define the initial state distribution $\rho = \{\xi, \xi, \cdots, \xi, 1-(|\mathcal{S}|-1)\xi\}$, which is a probability distribution because $(|\mathcal{S}|-1)\xi \le 1$.

We define the class of all context dependent deterministic policies as Π_{det}. We first sample the expert policy for each context uniformly and independently from Π_{det}, that is for each $s \in \mathcal{S}$, $c \in \mathcal{C}$, the optimal action $\pi^*_c(s)$ is uniformly sampled from $\mathcal{A}$. Given π^*_c, we define the reward function $r_c(\pi^*_c)$ where at each state, choosing the expert's action $\pi^*_c(s)$ receives reward 1 and otherwise 0. We denote the MDP with the transition and initial state distribution constructed above and the reward function $r_c(\pi^*_c)$ as $\mathcal{M}_c(\pi^*_c)$. It is obvious from the construction that the value of π^*_c in $\mathcal{M}_c(\pi^*_c)$ is H for any initial state s_1, since it receices a reward of 1 at every step.

Define the set of deterministic policies $\pi = \{\pi_c, c \in \mathcal{C}\}$ that agrees with the expert policy in dataset D as:

$$\Pi(\mathcal{D}) = \left\{\pi \in \Pi_{\text{det}}^{|\mathcal{C}|} : \ \pi_{c_k}(s^k_h) = a^k_h, \ \forall k \in [K], h \in [H]\right\}.$$

where we recall $a^k_h = \pi^*_{c_k}(s^k_h)$ in the expert dataset. We also denote

$$\mathcal{K}_c = \{k = 1, 2, \cdots, K : \ c_k = c\}\,, \quad \mathcal{D}_c = \{\tau_{c_k,k}, \ k \in \mathcal{K}_c\}\,.$$

as the subset of episode indices and episodes the context of which is c. We then denote the class of deterministic policies conditioning on context c that agrees with the expert π^*_c in dataset D_c as:

$$\Pi_c(\mathcal{D}_c) = \left\{\pi \in \Pi_{\text{det}} : \ \pi_c(s^k_h) = a^k_h\,, k \in \mathcal{K}_c\right\}.$$

We have the conditional distribution of $\pi^* = \{\pi_c^*, c \in \mathcal{C}\}$ conditioned on $\mathcal{D}$ is uniform in $\Pi(\mathcal{D})$, with the marginal conditional distribution of π_c^* being uniform in $\Pi_c(\mathcal{D}_c)$. Thus, the Bayes suboptimality gap of the output policy $\widehat{\pi} = \{\widehat{\pi}_c, c \in \mathcal{C}\}$ under the constructed MDP class $\mathcal{M}(\pi^*)$ can be written as:

$$\begin{aligned}&\mathbb{E}_{\pi^* \sim \mathrm{Unif}(\Pi(\mathcal{D}))}\left[\mathbb{E}_{c\in\mathrm{Unif}(\mathcal{C})}\left[\mathbb{E}_{s_1\sim\rho}\left[V^*_{\mathcal{M}_c(\pi_c^*)}(s_1;c) - V^{\widehat{\pi}_c}_{\mathcal{M}_c(\pi_c^*)}(s_1;c)\right]\right]\right]\\ =&\mathbb{E}_{c\in\mathrm{Unif}(\mathcal{C})}\left[\mathbb{E}_{\pi_c^*\sim\mathrm{Unif}(\Pi_c(\mathcal{D}_c))}\left[\mathbb{E}_{s_1\sim\rho}\left[H - V^{\widehat{\pi}_c}_{\mathcal{M}_c(\pi_c^*)}(s_1;c)\right]\right]\right] \qquad (4)\end{aligned}$$

which is because the value of π_c^* in our constructed MDP $\mathcal{M}_c(\pi_c^*)$ is H; and each entry π_c^* of π^* is independently uniform distributed in $\Pi_c(\mathcal{D}_c)$ so we apply Fubini's Theorem to exchange the expectations.

We denote the set of initial states visited in episodes with context c as

$$\mathcal{S}_1(\mathcal{D}_c) = \left\{s \in \mathcal{S} : \exists k \in \mathcal{K}_c, s_1^k = s\right\} .$$

Recall that in our MDP design each state is an absorbing state, so $\mathcal{S}_1(\mathcal{D}_c)$ is also the set of total states visited under context c in $\mathcal{D}$. Intuitively speaking, in context c, if $s_1 \sim \rho$ is from a unvisited state outside of $\mathcal{S}_1(\mathcal{D}_c)$, then the the estimated policy $\widehat{\pi}$ will suffer from constant errors. This is because the potential distribution of optimal action $\pi_c^*(s_1)$ given $\mathcal{D}$ is distributed uniformly in $\mathcal{A}$, independent of the data from other contexts, thus the agent can not guess the right action with high probability. Given context $c \in \mathcal{C}$, we have the following Lemma from Rajaraman et al. (2020).

Lemma 1 (Lemma A.19 of Rajaraman et al. (2020))**.** *Given context $c \in \mathcal{C}$ and dataset $\mathcal{D}$, we have the Bayes suboptimality of the imitation policy $\widehat{\pi}_c$ lower bounded as:*

$$\mathbb{E}_{\pi_c^*\sim\mathrm{Unif}(\Pi_c(\mathcal{D}_c))}\left[\mathbb{E}_{s_1\sim\rho}\left[H - V^{\widehat{\pi}_c}_{\mathcal{M}_c(\pi_c^*)}(s_1;c)\right]\right] \geq H\left(1 - \frac{1}{|\mathcal{A}|}\right)\left(1 - \rho\left(\mathcal{S}_1(\mathcal{D}_c)\right)\right)$$

where $\rho(\mathcal{S}_1(\mathcal{D}_c)) = \sum_{s\in\mathcal{S}_1(\mathcal{D}_c)} \rho(s)$ is the total measure of all initial states visited in $\mathcal{D}_c$.

Next, we lower bound the average uncovered initial-state mass $\mathbb{E}_c\left[1 - \rho(\mathcal{S}_1(\mathcal{D}_c))\right]$. Even with the task schedule fixed, $\mathcal{S}_1(\mathcal{D}_c)$ is a random set, because the initial states are drawn from ρ; the statement below is therefore an expectation over the draw of $\mathcal{D}$ rather than a deterministic statement about one dataset.

Lemma 2. *Let $\{n_c\}_{c\in\mathcal{C}}$ be the agent's task schedule (Section B.2.1), so that $\sum_{c\in\mathcal{C}} n_c = K$ and $\mathcal{D}$ contains n_c expert episodes for each context c. Then, with ξ as in (3),*

$$\mathbb{E}_{\mathcal{D}}\mathbb{E}_{c\sim\mathrm{Unif}(\mathcal{C})}\left[1 - \rho(\mathcal{S}_1(\mathcal{D}_c))\right] \geq (|\mathcal{S}| - 1)\,\xi\,(1-\xi)^{K/|\mathcal{C}|} \geq \frac{1}{e}\min\left\{1, \frac{|\mathcal{S}| - 1}{\frac{K}{|\mathcal{C}|} + 1}\right\} .$$

Proof. Recall $\rho = \{\xi, \xi, \cdots, \xi, 1 - (|\mathcal{S}| - 1)\xi\}$ with ξ from (3) and $\bar{N} = \frac{K}{|\mathcal{C}|}$. Since for every episode index $k \in \mathcal{K}_c$ the initial state s_1^k is drawn independently from ρ, and the schedule fixes n_c before those draws,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[1 - \rho(\mathcal{S}_1(\mathcal{D}_c))\right] =& \sum_{s\in\mathcal{S}} \rho(s)(1-\rho(s))^{n_c}\\ \geq& (|\mathcal{S}| - 1)\,\xi\,(1-\xi)^{n_c} ,\end{aligned}$$

where we kept only the $|\mathcal{S}| - 1$ light states and dropped the non-negative contribution of the heavy state. The map $t \mapsto (1-\xi)^t$ is convex, so Jensen's inequality and $\frac{1}{|\mathcal{C}|}\sum_c n_c = \bar{N}$ give

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\mathbb{E}_{c\sim\mathrm{Unif}(\mathcal{C})}\left[1 - \rho(\mathcal{S}_1(\mathcal{D}_c))\right] \geq& (|\mathcal{S}| - 1)\,\xi\,\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(1-\xi)^{n_c}\\ \geq& (|\mathcal{S}| - 1)\,\xi\,(1-\xi)^{\bar{N}} .\end{aligned}$$

It remains to evaluate the two regimes of (3).

Data-rich regime, $\bar{N} \geq |\mathcal{S}| - 2$. Here $\xi = \frac{1}{\bar{N}+1}$, and using $\left(1 - \frac{1}{m}\right)^{m-1} \geq e^{-1}$ with $m = \bar{N} + 1$,

$$(|\mathcal{S}| - 1)\,\xi\,(1-\xi)^{\bar{N}} \geq \frac{|\mathcal{S}| - 1}{e\left(\bar{N} + 1\right)} = \frac{1}{e}\min\left\{1, \frac{|\mathcal{S}| - 1}{\bar{N} + 1}\right\} ,$$

where the last equality uses $|\mathcal{S}| - 1 \le \bar{N} + 1$ in this regime.

Data-scarce regime, $\bar{N} < |\mathcal{S}| - 2$. Here $\xi = \frac{1}{|\mathcal{S}|-1}$, so $(|\mathcal{S}| - 1)\xi = 1$ and

$$(|\mathcal{S}| - 1)\,\xi\,(1-\xi)^{\bar{N}} = \left(1 - \frac{1}{|\mathcal{S}| - 1}\right)^{\bar{N}} \ge \left(1 - \frac{1}{|\mathcal{S}| - 1}\right)^{|\mathcal{S}|-2} \ge \frac{1}{e} = \frac{1}{e}\min\left\{1, \frac{|\mathcal{S}| - 1}{\bar{N} + 1}\right\},$$

where the last equality uses $\bar{N} + 1 < |\mathcal{S}| - 1$ in this regime. Combining the two regimes proves the claim. □

The second regime is exactly the term that a bound of the form $\Omega\left(\frac{|\mathcal{C}|}{K}\right)$ omits: when each task receives fewer demonstrations than there are states, the uncovered mass is a constant and no rate in K is available at all.

Finally, we prove the main theorem.

*Proof of Part **(ii)**, Theorem 1.* To bound the worst-case suboptimality of $\widehat{\pi}$ it suffices to bound the Bayes suboptimality under our joint distribution of expert policies $\pi^* \sim \mathrm{Unif}\,(\Pi(\mathcal{D}))$ and constructed MDP instances $\mathcal{M}(\pi^*)$. Combining (4) with Lemma 1 and then taking the expectation over $\mathcal{D}$ and applying Lemma 2, we obtain

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\mathbb{E}_{\pi^*\sim\mathrm{Unif}(\Pi(\mathcal{D}))}\left[\mathbb{E}_{c\in\mathrm{Unif}(\mathcal{C})}\left[\mathbb{E}_{s_1\sim\rho}\left[V^*_{\mathcal{M}_c(\pi^*_c)}(s_1;c) - V^{\widehat{\pi}_c}_{\mathcal{M}_c(\pi^*_c)}(s_1;c)\right]\right]\right] \\
&\ge H\left(1 - \frac{1}{|\mathcal{A}|}\right)\mathbb{E}_{\mathcal{D}}\mathbb{E}_{c\sim\mathrm{Unif}(\mathcal{C})}\left[1 - \rho(\mathcal{S}_1(\mathcal{D}_c))\right] \\
&\ge \frac{H}{e}\left(1 - \frac{1}{|\mathcal{A}|}\right)\min\left\{1, \frac{|\mathcal{S}| - 1}{\frac{K}{|\mathcal{C}|} + 1}\right\} = \Omega\left(\min\left\{H, \frac{H|\mathcal{S}||\mathcal{C}|}{K}\right\}\right).
\end{aligned}$$

Thus there exists a worst-case instance $\pi^* = \{\pi^*_c, c \in \mathcal{C}\}$ and MDP family $\mathcal{M}(\pi^*) = \{\mathcal{M}_c(\pi^*_c), c \in \mathcal{C}\}$ under which the average suboptimality of the imitation learner is at least $\Omega\left(\min\left\{H, \frac{H|\mathcal{S}||\mathcal{C}|}{K}\right\}\right)$. □

Equivalently, in terms of sample complexity, the imitation learner needs $K = \Omega\left(\frac{H|\mathcal{S}||\mathcal{C}|}{\epsilon}\right)$ episodes to reach average suboptimality $\epsilon \in (0, H)$, whereas by Theorem 3 the model-based planner needs $K = \tilde{\mathcal{O}}\left(\frac{H^5|\mathcal{S}|^2|\mathcal{A}|}{\epsilon^2}\right)$ episodes and is independent of $|\mathcal{C}|$.

Remark 1 (When the separation is real)**.** *Quoting the comparison as "$K^{-1/2}$ versus $|\mathcal{C}|/K$" invites the wrong reading, because in K alone the imitation rate K^{-1} is the* faster *of the two and the bounds are not ordered for every choice of parameters. Everything hinges on the $|\mathcal{C}|$ factor. The model-based sample complexity is the smaller one exactly when*

$$|\mathcal{C}| \gtrsim \frac{H^4|\mathcal{S}||\mathcal{A}|}{\epsilon}, \tag{5}$$

that is, once the tasks are many relative to the size of any one of them and to the accuracy sought. Our experiments sit in this regime: one manipulation workspace, compactly described, with an open-ended list of language-specified jobs to do in it. Outside (5), with few tasks and a large state space, imitation learning is the better protocol, which is the familiar single-task picture.

Remark 2 (What information each bound actually uses)**.** *The two halves of Theorem 1 are proved under the two different interfaces of Section B.2.1. Since the interfaces differ, it matters which part of the difference is carrying the result.*

(a) The gap does not come from the imitation learner's ignorance of the dynamics. *Every state in the hard instance is absorbing, so one trivial transition kernel serves the entire instance family. We can therefore give the imitation learner $\mathbb{P}$ outright and lose nothing: $\mathbb{P}$ is a constant of the family, so a learner reading $(\mathcal{D}, \mathbb{P})$ is still a learner reading $\mathcal{D}$, and the proof runs unchanged. Theorem 1**(ii)** accordingly holds against imitation learners that know the dynamics exactly, which disposes of the most obvious objection to the comparison.*

(b) It comes from reward evaluability, and that assumption is load-bearing. *The one thing the model-based learner has and the imitation learner lacks is the ability to score any task's reward at planning time. Take it away and the advantage goes with it. Suppose the model-based learner also had to estimate r_c from reward samples, with no structure shared across tasks, as the tabular setting allows. In this instance r_c encodes the expert action at every state, so pinning down r_c at a state means visiting that state under context c; Lemma 1 and Lemma 2 then apply to it word for word, and it too pays $\Omega\left(\min\left\{H, H|\mathcal{S}||\mathcal{C}|/K\right\}\right)$. Read the tabular theorem, then, as measuring what a transferable success detector is worth alongside shared-dynamics data. In our system that detector is a VLM judging an unseen task from a terminal frame. The theorem does not say that planning beats behavior cloning when the two see the same observations.*

(c) The linear-MDP results of Section B.4 are the matched-information counterpart. *There the reward is withheld from both: the model-based learner estimates $\{\Theta_h\}$ from noisy reward samples much as the imitation learner fits a classifier to expert labels. Each sees K episodes and each must reach unseen tasks through a learned representation, so whatever separates them there is about representation rather than about access (Section B.4.2).*

B.4 Linear Function Approximation

In this section, we provide the results under linear function approximation. Unlike the tabular setting, here the reward is unknown to *both* learners: the model-based agent must estimate the shared reward representation from noisy reward samples, while the imitation agent must fit a policy to expert labels. The separation established below is therefore representational rather than informational (cf. Remark 2**(c)**). First, we provide the definition of linear MDP following Jin et al. (2023):

Definition 1 (Linear MDP (Jin et al., 2023))**.** *MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \rho, H)$ is called a linear MDP with feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, if there exist d (unknown) signed measures $\boldsymbol{\mu}_h = \left(\mu_h^{(1)}, \mu_h^{(2)}, \cdots, \mu_h^{(d)}\right)$ over $\mathcal{S}$ such that the transition probability can be represented as linear functions*

$$\mathbb{P}_h(\cdot|s,a) = \langle \phi(s,a), \boldsymbol{\mu}_h(\cdot) \rangle \ ,$$

and (unknown) vectors $\theta_h \in \mathbb{R}^d$ such that the expectation of the random reward can be represented as a linear function

$$\mathbb{E}\left[r_h(s,a)\right] = \langle \phi(s,a), \theta_h \rangle \ , \ \ 0 \le r_h(s,a) \le 1,$$

for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $h \in [H]$. Without loss of generality, we assume $\|\phi(s,a)\| \le 1$ for all (s,a) and $\max\left\{\|\boldsymbol{\mu}_h(\mathcal{S})\|, \|\theta_h\|\right\} \le \sqrt{d}$ for all h.

Given the above definition, we further explore the setting of learning the reward function for generalizing to new tasks with task features. Specifically, we assume that the per-task reward vectors $\theta_{h,c} \in \mathbb{R}^d$ share a common task feature map $\psi(c) \in \mathbb{R}^{d'}$ known to the agent, with $\theta_{h,c} = \Theta_h \psi(c)$. Thus, by learning the transformations $\{\Theta_h\}_{h\in[H]}$ using reward samples from source tasks, we can generalize to any new task using the representation ψ. We point out that learning such a reward function with linear structure is non-trivial due to the random noise in the reward received (Abbasi-Yadkori et al., 2011).

Assumption 1 (Reward representation)**.** *There exists a feature map $\psi : \mathcal{C} \to \mathbb{R}^{d'}$ and (unknown) matrices $\Theta_h \in \mathbb{R}^{d \times d'}$, $h \in [H]$, such that the random reward for context c satisfies $\mathbb{E}\left[r_{h,c}(s,a)\right] = \phi(s,a)^\top \Theta_h \psi(c)$ with $0 \le r_{h,c}(s,a) \le 1$ and $\|\psi(c)\| \le 1$, for all $(s,a,c,h) \in \mathcal{S} \times \mathcal{A} \times \mathcal{C} \times [H]$. We assume the set $\{\psi(c), c \in \mathcal{C}\} \subset \mathbb{R}^{d'}$ is compact and span$(\{\psi(c), c \in \mathcal{C}\}) = \mathbb{R}^{d'}$. We further assume the normalization $\|\Theta_h\|_F \le \sqrt{dd'}$ for all $h \in [H]$, equivalently $\|\mathrm{vec}(\Theta_h)\| \le \sqrt{dd'}$, which matches the normalization $\|\theta_h\| \le \sqrt{d}$ placed on the single-task reward vector in Definition 1. Allowing Θ_h to depend on h is what makes the class closed under step-dependent rewards; a single shared Θ cannot, for instance, express a reward that is identically zero at $h = 1$ and non-trivial at $h = 2$, a construction we use in Section B.4.2.*

The normalization of Θ_h has to be assumed; it is not implied by the rest of the assumption. Bounded rewards and bounded task features leave the parameter free to blow up whenever the task features are

small: in the scalar case $d = d' = 1$, taking $\phi \equiv 1$, $\psi(c) \equiv \eta$ and $\Theta = 1/\eta$ gives reward 1 for every $\eta > 0$, while $|\Theta| \to \infty$ as $\eta \to 0$. The span condition rules out $\psi \equiv 0$ but not this sequence. Since the ridge estimator of Lemma 5 pays a regularization bias proportional to $\|\vartheta_h\|$, the bound enters the analysis and we state it up front. A general bound $\|\Theta_h\|_F \le B_\Theta$ works equally well and enters the reward error of Lemma 5, and hence the rate of Theorem 2, linearly in B_Θ; we fix $B_\Theta = \sqrt{dd'}$ only to keep the $\mathrm{poly}(d, d')$ bookkeeping in one place. To explore the feature set, we define the G-optimal design of the feature set as:

Theorem 4 (Kiefer–Wolfowitz). *There exists a distribution $p \in \Delta(\mathcal{C})$ such that $|\mathrm{Supp}(p)| \le \frac{d'(d'+1)}{2}$ and $\max_{c\in\mathcal{C}} \|\psi(c)\|^2_{W^{-1}} = d'$, i.e. $\max_{c\in\mathcal{C}} \|\psi(c)\|_{W^{-1}} = \sqrt{d'}$, where W is the covariance matrix under p:*

$$W = \sum_{c\in\mathcal{C}} p(c)\psi(c)\psi(c)^\top .$$

We write $p_c = p(c)$ below. Note that the design guarantee is on the *squared* norm; carrying it as $\|\psi(c)\|_{W^{-1}} \le d'$ would overstate the dimension dependence by a square.

We also assume that the MDP has well-conditioned covariates and there exists a *unknown* policy that can explore the h layer of the MDP, following standard practice (Kong et al., 2023; Li et al., 2020)

Assumption 2 (Exploratory assumption). *There exists $\kappa > 0$, such that*

$$\min_{h\in[H]} \sup_{\pi} \lambda_{\min}\left(\mathbb{E}_\pi\left[\phi(s_h, a_h)\phi(s_h, a_h)^\top\right]\right) \ge \kappa .$$

B.4.1 Model-based Planning Agent

In this section, we prove the suboptimality gap for the model-based planning agent under our linear function approximation. We first restate the result from Wagenmaker et al. (2022) that enables sufficient exploration of the feature space, without knowledge of any exploratory policies:

Theorem 5 (Theorem 4, (Wagenmaker et al., 2022)). *Fix $\gamma \in [0, 1]$, under Assumption 2. There exists an algorithm that collects observations $\mathcal{D} = \left\{(s_1^k, a_1^k), \cdots, (s_H^k, a_H^k)\right\}_{k=1}^K$, such that with probability $1 - \delta$, we have for any h:*

$$\lambda_{\min}\left(\sum_{k=1}^K \phi(s_h^k, a_h^k)\phi(s_h^k, a_h^k)^\top\right) \ge \frac{\kappa}{\gamma^2} .$$

after running at most

$$K = \mathcal{O}\left(\frac{dH}{\kappa\gamma^2}\right)$$

where $\mathcal{O}$ hides polynomial dependencies on H and $\log\frac{1}{\delta}$.

Data Collection Protocol. We jointly explore the state-action feature space $\phi(s, a)$ and the context feature space $\psi(c)$ by combining our exploration algorithm with a G-optimal design. Specifically, for each context $c \in \mathrm{Supp}(p_c)$, we execute the exploration procedure described in Theorem 5 for K_c episodes. We denote the dataset collected under context c as $\mathcal{D}_c$, and the corresponding episode indices as $\mathcal{K}_c = \{k \in [K] : c_k = c\}$. The total dataset across all contexts is defined as $\mathcal{D} = \cup_{c\in\mathcal{C}}\mathcal{D}_c$. To estimate the model parameters, we define the context-specific covariance matrices at step h as $\Lambda_h^c = \sum_{k\in\mathcal{K}_c} \phi(s_h^k, a_h^k)\phi(s_h^k, a_h^k)^\top$, and the aggregate covariance across all episodes as $\Lambda_h = \sum_{k=1}^K \phi(s_h^k, a_h^k)\phi(s_h^k, a_h^k)^\top + \boldsymbol{I}$.

Building on the high probability event in Theorem 5, we have the total number of episodes be bounded as $\mathcal{O}\left(\frac{\mathrm{poly}(d,d',H)}{\kappa\gamma^2}\right)$:

Lemma 3. *For any $\delta > 0$, with probability $1 - \delta$, we have for each $c \in \mathrm{Supp}(p_c)$ and $h \in [H]$, $\lambda_{\min}\left(\Lambda_h^c\right) \ge \frac{\kappa}{\gamma^2}$, and the total number of episodes is bounded as $K = \tilde{\mathcal{O}}\left(\frac{d(d')^2H}{\kappa\gamma^2}\right)$.*

Proof. From Theorem 4, we have $|\mathrm{Supp}(p_c)| = \mathcal{O}\left((d')^2\right)$. Thus, the procedure in Theorem 5 is run for at most $|\mathrm{Supp}(p_c)|$ rounds, each using $\mathcal{O}\left(\frac{dH}{\kappa\gamma^2}\right)$ episodes, and the guarantee follows by a union bound of the event in Theorem 5 over all rounds (with failure probability $\delta/|\mathrm{Supp}(p_c)|$ per round). □

Next, we bound the error of the estimated model:

Lemma 4. *We estimate the transition model as:*

$$\widehat{\boldsymbol{\mu}}_h = \Lambda_h^{-1} \sum_{k=1}^{K} \phi(s_h^k, a_h^k) \delta\left(s_{h+1}^k\right)^\top .$$

We have the following error bound for any function $V : \mathcal{S} \to \mathbb{R}$ that is the optimal value function for a linear MDP: with probability at least $1-\delta$, we have

$$\left|\left(\widehat{\mathbb{P}}_h - \mathbb{P}_h\right) V(s,a)\right| \leq \tilde{\mathcal{O}}\left(\mathrm{poly}\left(d, H\right) \sqrt{\frac{\gamma^2}{\kappa}}\right) ,$$

for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, $h \in [H]$ and $0 < \delta < 1$.

Proof. We decompose the LHS as:

$$\begin{aligned}\left|\left(\widehat{\mathbb{P}}_h - \mathbb{P}_h\right) V(s,a)\right| &= \left|\left\langle \phi(s,a), \left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right) V \right\rangle\right| \\ &\leq \|\phi(s,a)\|_{\Lambda_h^{-1}} \left\|\left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right) V\right\|_{\Lambda_h} ,\end{aligned}$$

where $\left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right) V = \sum_{s' \in \mathcal{S}} \left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right)(s') V(s') \in \mathbb{R}^d$ is a vector that does not depend on (s,a). According to our least squares estimator, we have:

$$\begin{aligned}\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h =& \Lambda_h^{-1} \sum_{k=1}^{K} \phi(s_h^k, a_h^k) \delta\left(s_{h+1}^k\right)^\top - \Lambda_h^{-1} \left(\sum_{k=1}^{K} \phi(s_h^k, a_h^k) \mathbb{P}(\cdot | s_h^k, a_h^k) + \boldsymbol{\mu}_h\right) \\ =& \Lambda_h^{-1} \sum_{k=1}^{K} \phi(s_h^k, a_h^k) \left(\delta\left(s_{h+1}^k\right)^\top - \mathbb{P}(\cdot | s_h^k, a_h^k)\right) - \Lambda_h^{-1} \boldsymbol{\mu}_h ,\end{aligned}$$

which is because of our linear transition function:

$$\sum_{k=1}^{K} \phi(s_h^k, a_h^k) \mathbb{P}(\cdot | s_h^k, a_h^k) + \boldsymbol{\mu}_h = \sum_{k=1}^{K} \phi(s_h^k, a_h^k) \phi(s_h^k, a_h^k)^\top \boldsymbol{\mu}_h + \boldsymbol{\mu}_h = \Lambda_h \boldsymbol{\mu}_h .$$

Thus, we have:

$$\left\|\left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right) V\right\|_{\Lambda_h} \leq \left\|\sum_{k=1}^{K} \phi(s_h^k, a_h^k) \left(V(s_{h+1}^k) - \mathbb{E}\left[V(s_{h+1}^k) | s_h^k, a_h^k\right]\right)\right\|_{\Lambda_h^{-1}} + \sqrt{d} .$$

Applying Lemma 7 and Lemma 8 with the covering parameter $\epsilon = 1/K$, we have:

$$\left\|\left(\widehat{\boldsymbol{\mu}}_h - \boldsymbol{\mu}_h\right) V\right\|_{\Lambda_h} \leq \mathrm{poly}\left(d, H, \log K, \log \frac{1}{\delta}\right) .$$

On the other hand, based the event in on Lemma 3, we have $\Lambda_h \succeq \sup_{c \in \mathrm{Supp}(p_c)} \Lambda_h^c \succeq \frac{\kappa}{\gamma^2} \boldsymbol{I}$, thus we have $\|\phi(s,a)\|_{\Lambda_h^{-1}} \leq \sqrt{\frac{\gamma^2}{\kappa}}$. Also, we have $K = \tilde{\mathcal{O}}\left(\frac{d(d')^2 H}{\kappa\gamma^2}\right)$ by Lemma 3. Putting these together proves the lemma. □

Now, we move on to estimate the reward parameters Θ_h. Note that the reward mean can be written as:

$$\mathbb{E}[r_h(s,a;c)] = \phi(s,a)^\top \Theta_h \psi(c) = \left(\psi(c)^\top \otimes \phi(s,a)^\top\right) \mathrm{vec}\left(\Theta_h\right)$$

Denote $\varphi(s,a,c)$ as $\psi(c) \otimes \phi(s,a)$ and $\mathrm{vec}\left(\Theta_h\right) = \vartheta_h$, we have $\phi(s,a)^\top \Theta_h \psi(c) = \varphi(s,a,c)^\top \vartheta_h$. We have the following results for the reward estimate $\widehat{r}$:

Lemma 5. *Denote, for each $h \in [H]$,*

$$\Sigma_h = \left(\boldsymbol{I} + \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) \varphi(s_h^k, a_h^k, c_k)^\top \right) \in \mathbb{R}^{dd' \times dd'} ,$$

then we estimate $\widehat{\vartheta}_h$, the vectorization of $\widehat{\Theta}_h$, using linear regression:

$$\widehat{\vartheta}_h = \Sigma_h^{-1} \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) r_h(s_h^k, a_h^k, c_k) .$$

We have with probability at least $1-\delta$ for $0 < \delta < 1$, for all $(s, a, c) \in \mathcal{S} \times \mathcal{A} \times \mathcal{C}$,

$$|\widehat{r}_h(s, a; c) - r_h(s, a; c)| \leq \tilde{\mathcal{O}} \left(\text{poly}\,(d, d') \sqrt{\frac{\gamma^2}{\kappa}} \right) .$$

Proof. Using the linear regression estimator, we have:

$$\widehat{\vartheta}_h - \vartheta_h = \Sigma_h^{-1} \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) \left(r_h(s_h^k, a_h^k, c_k) - \mathbb{E}\left[r_h(s_h^k, a_h^k, c_k) \right] \right) - \Sigma_h^{-1} \vartheta_h .$$

Thus, we can bound the estimation error of as:

$$\begin{aligned}
& |\widehat{r}_h(s_h, a_h, c) - r_h(s_h, a_h, c)| \\
=& \left| \left\langle \varphi(s_h, a_h, c), \widehat{\vartheta}_h - \vartheta_h \right\rangle \right| \\
\leq& \|\varphi(s_h, a_h, c)\|_{\Sigma_h^{-1}} \left\| -\vartheta_h + \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) \eta_h^k \right\|_{\Sigma_h^{-1}} \\
\leq& \tilde{\mathcal{O}} \left(\sqrt{dd'} + \sqrt{dd' + \log \frac{1}{\delta}} \right) \|\varphi(s_h, a_h, c)\|_{\Sigma_h^{-1}}
\end{aligned}$$

where η_h^k is the noise of the random reward:

$$\eta_h^k = \left(r_h(s_h^k, a_h^k, c_k) - \mathbb{E}\left[r_h(s_h^k, a_h^k, c_k) \right] \right) ,$$

which is bounded in $[-1, 1]$ and hence 1-subGaussian, and we use Lemma 6 to bound $\left\| \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) \eta_h^k \right\|_{\Sigma_h^{-1}}$; the term $\|\vartheta_h\|_{\Sigma_h^{-1}} \leq \|\vartheta_h\| \leq \sqrt{dd'}$ uses $\Sigma_h \succeq \boldsymbol{I}$ together with the normalization $\|\Theta_h\|_F \leq \sqrt{dd'}$ of Assumption 1.

We now process the term $\|\varphi(s_h, a_h, c)\|_{\Sigma_h^{-1}}$. First, we lower bound Σ_h:

$$\begin{aligned}
& \sum_{k=1}^{K} \varphi(s_h^k, a_h^k, c_k) \varphi(s_h^k, a_h^k, c_k)^\top \\
=& \sum_{k=1}^{K} \left(\psi(c_k) \otimes \phi(s_h^k, a_h^k) \right) \left(\psi(c_k)^\top \otimes \phi(s_h^k, a_h^k)^\top \right) \\
=& \sum_{k=1}^{K} \left(\psi(c_k) \psi(c_k)^\top \right) \otimes \left(\phi(s_h^k, a_h^k) \phi(s_h^k, a_h^k)^\top \right) \\
=& \sum_{c \in \text{Supp}(p_c)} \left(\psi(c) \psi(c)^\top \right) \otimes \left(\sum_{k \in \mathcal{K}_c} \left(\phi(s_h^k, a_h^k) \phi(s_h^k, a_h^k)^\top \right) \right) \\
\succeq& \left(\sum_{c \in \text{Supp}(p_c)} \left(\psi(c) \psi(c)^\top \right) \right) \otimes \frac{\kappa}{\gamma^2} \boldsymbol{I}
\end{aligned}$$

where the last step uses Lemma 3 and the fact that $A \otimes B \succeq A \otimes B'$ whenever $A \succeq 0$ and $B \succeq B'$. Denote $G = \sum_{c \in \mathrm{Supp}(p_c)} \left(\psi(c)\psi(c)^\top\right)$. Since $p_c \in [0,1]$ we have $G \succeq \sum_{c \in \mathrm{Supp}(p_c)} p_c \left(\psi(c)\psi(c)^\top\right) = W$, so by the property of the G-optimal design in Theorem 4, for any c,

$$\|\psi(c)\|_{G^{-1}} \leq \|\psi(c)\|_{W^{-1}} \leq \sqrt{d'}\,.$$

As a result, we have:

$$\begin{aligned}
&\|\varphi(s_h, a_h, c)\|_{\Sigma_h^{-1}} \\
=&\left\|\left(\psi(c) \otimes \phi(s_h, a_h)\right) \Sigma_h^{-1/2}\right\|_2 \\
\leq&\left\|\left(\psi(c) \otimes \phi(s_h, a_h)\right) \left(G \otimes \frac{\kappa}{\gamma^2}\boldsymbol{I}\right)^{-1/2}\right\| \\
\leq&\sqrt{\frac{\gamma^2}{\kappa}} \left\|\left(\psi(c)G^{-1/2}\right) \otimes \phi(s_h, a_h)\right\| \\
=&\sqrt{\frac{\gamma^2}{\kappa}} \|\psi(c)\|_{G^{-1}} \|\phi(s_h, a_h)\| \\
\leq&\sqrt{\frac{\gamma^2}{\kappa}}\sqrt{d'}
\end{aligned}$$

using $\|\phi(s_h, a_h)\| \leq 1$. Thus we have with probability at least $1-\delta$, for any (s_h, a_h, c):

$$|\widehat{r}_h(s_h, a_h, c) - r_h(s_h, a_h, c)| \leq \tilde{\mathcal{O}}\left(\sqrt{\frac{\gamma^2}{\kappa}}\sqrt{d}\,d'\right)\,.$$

□

*Proof of Part **(i)**, Theorem 2.* Finally, we bound the error of $\widehat{\pi}$ which is the optimal policy in the estimated MDP $\widehat{\mathcal{M}}$, against π^* which is the optimal policy of $\mathcal{M}$. For the value function of a policy π in $\mathcal{M}$ and $\widehat{\mathcal{M}}$, we denote them as V^π and $\widehat{V}^\pi$, respectively. We can decompose the optimality gap as:

$$V^* - V^{\widehat{\pi}} = \left(V^{\pi^*} - \widehat{V}^{\pi^*}\right) + \left(\widehat{V}^{\pi^*} - \widehat{V}^{\widehat{\pi}}\right) + \left(\widehat{V}^{\widehat{\pi}} - V^{\widehat{\pi}}\right) \leq \left(V^{\pi^*} - \widehat{V}^{\pi^*}\right) + \left(\widehat{V}^{\widehat{\pi}} - V^{\widehat{\pi}}\right)\,.$$

Using the simulation lemma, we have:

$$\begin{aligned}
V^{\pi^*} - \widehat{V}^{\pi^*} =&\mathbb{E}_{\widehat{\mathcal{M}}, \pi^*}\left[\sum_{h=1}^{H} \left(r_c(s_h, a_h) - \widehat{r}_c(s_h, a_h)\right) + \left(\mathbb{P}_h - \widehat{\mathbb{P}}_h\right) V^{\pi^*}(s_h, a_h)\right]\,, \\
\widehat{V}^{\widehat{\pi}} - V^{\widehat{\pi}} =&\mathbb{E}_{\mathcal{M}, \widehat{\pi}}\left[\sum_{h=1}^{H} \left(\widehat{r}_c(s_h, a_h) - r_c(s_h, a_h)\right) + \left(\widehat{\mathbb{P}}_h - \mathbb{P}_h\right) \widehat{V}^{\widehat{\pi}}(s_h, a_h)\right]\,.
\end{aligned}$$

Here $\widehat{V}^{\widehat{\pi}}$ is computed with the clipped backups of Remark 4, so both V^{π^*} and $\widehat{V}^{\widehat{\pi}}$ take values in $[0, H]$ and both lie in a class of the form $\mathcal{V}_R$ covered by Lemma 8. Invoking Lemma 4 and Lemma 5, we have for any $(s, a, c) \in \mathcal{S} \times \mathcal{A} \times \mathcal{C}$:

$$\begin{aligned}
&|\widehat{r}_c(s_h, a_h) - r_c(s_h, a_h)| \leq \tilde{\mathcal{O}}\left(\sqrt{\frac{\gamma^2}{\kappa}}\mathrm{poly}\left(d, d'\right)\right)\,, \\
&\left|\left(\widehat{\mathbb{P}}_h - \mathbb{P}_h\right) V(s, a)\right| \leq \tilde{\mathcal{O}}\left(\mathrm{poly}\left(d, H\right)\sqrt{\frac{\gamma^2}{\kappa}}\right)\,,\ V \in \left\{\widehat{V}^{\widehat{\pi}}, V^*\right\}\,.
\end{aligned}$$

Based on Lemma 3, we have $K = \tilde{\mathcal{O}}\left(\frac{d(d')^2 H}{\kappa\gamma^2}\right)$, i.e. $\sqrt{\frac{\gamma^2}{\kappa}} = \tilde{\mathcal{O}}\left(\frac{\sqrt{d(d')^2 H}}{\kappa\sqrt{K}}\right)$, so both $V^{\pi^*} - \widehat{V}^{\pi^*}$ and $\widehat{V}^{\widehat{\pi}} - V^{\widehat{\pi}}$ are bounded by $\tilde{\mathcal{O}}\left(\frac{\text{poly}(d,d',H)}{\kappa\sqrt{K}}\right)$. We then obtain the desired result

$$V^* - V^{\widehat{\pi}} \leq \tilde{\mathcal{O}}\left(\min\left\{H, \frac{\text{poly}\,(d, d', H)}{\kappa\sqrt{K}}\right\}\right),$$

where, as in (2), the min with H records that the bound is informative only once $K \gtrsim \text{poly}(d, d', H)/\kappa^2$; the guarantee holds simultaneously for every context $c \in \mathcal{C}$ and in particular does not depend on $|\mathcal{C}|$. □

B.4.2 Imitation Learning Agent

Here we provide proof for Part **(ii)** of Theorem 2.

Positive Result in Single Task Learning. In the single task setting, according to the properties of linear MDP, for each $h \in [H]$ there exists $w_h^* \in \mathbb{R}^d$ such that $Q_h^*(s,a) = \langle \phi(s,a), w_h^* \rangle$, so the optimal policy can be represented as a linear classifier using Q_h^*: $\pi_h^*(s) = \text{argmax}_a\, Q_h^*(s,a) = \text{argmax}_a\, \langle \phi(s,a), w_h^* \rangle$. Following the results in Rajaraman et al. (2021), we can learn the optimal linear policy with suboptimality gap $\tilde{\mathcal{O}}\left(\frac{\text{poly}(d,H)}{K}\right)$ through imitation learning with empirical risk minimization (ERM).

Next, we provide the proof of the hardness result in the multi-task setting. Let $\mathcal{F}$ be a class of score functions $f(\phi(s,a); \psi(c))$ inducing the context-conditioned classifier policies $\pi_f(s;c) = \text{argmax}_a\, f(\phi(s,a); \psi(c))$. To prove that no $f \in \mathcal{F}$ satisfies $\pi^*(s;c) = \text{argmax}_a\, f(\phi(s,a); \psi(c))$ when $\mathcal{F}$ has low polynomial degree in $\psi(c)$, we construct a set of contexts c that are necessarily roots of $f(\phi(s,a_1); \psi(c)) - f(\phi(s,a_2); \psi(c))$, by exhibiting contexts at which the optimal action changes from a_1 to a_2. By creating a large set of such roots we show that f must have high-order dependency on $\psi(c)$. Notice that we do not require $Q^* \in \mathcal{F}$ for the optimal policy to be realizable, although the optimal Q-function $Q_h^*(s,a;c)$ also depends on $\psi(c)$ in a complicated way.

Here, we provide the construction of the hard instance. We construct our MDP with two levels $H = 2$ and feature dimension $d = 3$. We denote actions as $\mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\}$. Now, we define the feature maps as follows. At the fixed initial state s_1, the actions have features $\phi(s_1, 1) = [1, 0, 0]$, $\phi(s_1, 2) = [0, 1, 0]$ and for all actions $a = 3, \cdots, |\mathcal{A}|$, we have $\phi(s_1, a) = [0, 0, 1]$. We have three states at $h = 2$, with $\boldsymbol{\mu}_1(1) = [1, 0, 0]$, $\boldsymbol{\mu}_1(2) = [0, 1, 0]$ and $\boldsymbol{\mu}_1(3) = [0, 0, 1]$. So taking action 1 will move to state 1, taking action 2 will move to state 2 and taking action $a \geq 3$ will move to state 3. We design state 3 as an empty state, such that for all actions in state 3, the feature $\phi(3, a) = \mathbf{0}$ so the reward is always 0 in state 3. As a result, the value of state 3 is always 0.

At the second level, for the actions $a = 1, 2, \cdots, |\mathcal{A}|$, we design the feature set in states $s_2 = 1, 2$ to be

$$\phi(1,a) = \frac{1}{\sqrt{2}}\left[\cos\pi\tfrac{2a}{|\mathcal{A}|}, \sin\pi\tfrac{2a}{|\mathcal{A}|}, 1\right], \qquad \phi(2,a) = \frac{1}{\sqrt{2}}\left[\cos\pi\tfrac{2a-1}{|\mathcal{A}|}, \sin\pi\tfrac{2a-1}{|\mathcal{A}|}, 1\right].$$

The factor $\frac{1}{\sqrt{2}}$ is needed: $[\cos\alpha, \sin\alpha, 1]$ has Euclidean norm $\sqrt{2}$ and would violate the normalization $\|\phi(s,a)\| \leq 1$ required by Definition 1, which the rest of the analysis (notably Lemma 6 and Lemma 8) relies on.

For the context dependent reward, we define the continuous context set as $\mathcal{C} = [0, 1)$, and the context feature as $\psi(c) = \frac{1}{\sqrt{2}}[\cos 2\pi c, \sin 2\pi c, 1]$, so that $\|\psi(c)\| = 1$. We set

$$\Theta_1 = \mathbf{0}, \qquad \Theta_2 = \boldsymbol{I},$$

which is the step-dependent reward representation of Assumption 1. The instance meets that assumption's normalization, since $\|\Theta_1\|_F = 0$ and $\|\Theta_2\|_F = \sqrt{3} \leq 3 = \sqrt{dd'}$ here. The choice $\Theta_1 = \mathbf{0}$ is what makes the reward at the first level identically 0; with a single step-independent $\Theta = \boldsymbol{I}$ one would instead get $r_1(s_1, 1; c) = \frac{1}{2}\cos 2\pi c \neq 0$, contradicting the construction. With $\Theta_1 = \mathbf{0}$ the first-level reward vanishes and the Q-function $Q_1^*(s_1, a; c)$ of all actions $a \geq 3$ equals 0, while

$Q_1^*(s_1, 1; c) = V_2^*(1; c) > 0$ and $Q_1^*(s_1, 2; c) = V_2^*(2; c) > 0$, so the optimal first-level action is always 1 or 2.

Thus, we have the reward at the second level defined as $r_2(1, a; c) = \frac{1}{2}\left(\cos\left(2\pi c - 2\pi\frac{a}{|\mathcal{A}|}\right) + 1\right) \in [0, 1]$, and $r_2(2, a; c) = \frac{1}{2}\left(\cos\left(2\pi c - \pi\frac{2a-1}{|\mathcal{A}|}\right) + 1\right) \in [0, 1]$. Suppose $2|\mathcal{A}|c \in [n_c, n_c + 1]$ where n_c is an integer, then at the second level the optimal value function for the two states are:

$$\begin{aligned}
V^*(1, c) &= \max_i \frac{1}{2}\cos\left(2\pi c - 2\pi\frac{i}{|\mathcal{A}|}\right) + \frac{1}{2} \\
&= \max_i \frac{1}{2}\cos\left(\frac{\pi}{|\mathcal{A}|}\left(2|\mathcal{A}|c - 2i\right)\right) + \frac{1}{2} \\
&= \frac{1}{2}\cos\left(2\pi c - \pi\frac{2\lfloor\frac{n_c+1}{2}\rfloor}{|\mathcal{A}|}\right) + \frac{1}{2} \\
V^*(2, c) &= \max_i \frac{1}{2}\cos\left(2\pi c - \pi\frac{2i-1}{|\mathcal{A}|}\right) + \frac{1}{2} \\
&= \max_i \frac{1}{2}\cos\left(\frac{\pi}{|\mathcal{A}|}\left(2|\mathcal{A}|c - (2i-1)\right)\right) + \frac{1}{2} \\
&= \frac{1}{2}\cos\left(2\pi c - \pi\frac{2\lfloor\frac{n_c}{2}\rfloor + 1}{|\mathcal{A}|}\right) + \frac{1}{2}
\end{aligned}$$

Thus, the optimal action $\pi(s_1; c)$ is 1 if $V^*(1, c) > V^*(2, c)$ and 2 otherwise, since taking action 1 will move to state 1 and taking action 2 will move to state 2.

Now, assume that the optimal policy can be represented by a function

$$\pi^*(s_1; c) = \operatorname*{argmax}_{a\in\mathcal{A}} f_a(\psi(c))\,,$$

where, as established above, the maximum is attained at $a \in \{1, 2\}$ for every c, and where we assume f_a can be written as a polynomial in the features $\psi(c)$ with degree n:

$$f_a(\psi(c)) = \sum_{u+v+w\le n} g_a^{u+v+w}\psi(c)_0^u\psi(c)_1^v\psi(c)_2^w = \sum_{u+v\le n} g_a^{u,v}\sin^u 2\pi c\cos^v 2\pi c\,, a = 1, 2\,.$$

Consider $f(\psi(c)) = f_1(\psi(c)) - f_2(\psi(c))$, which by the display above is a trigonometric polynomial of degree at most n in $2\pi c$. Notice that at the points $c_i = \frac{2i+1}{4|\mathcal{A}|}$, $i = 0, 1, 2, \cdots, 2|\mathcal{A}| - 1$, we have $V^*(2, c_i) = V^*(1, c_i)$; and for any small enough $\epsilon > 0$,

$$\left(V^*(2, c_i - \epsilon) - V^*(1, c_i - \epsilon)\right)\left(V^*(2, c_i + \epsilon) - V^*(1, c_i + \epsilon)\right) < 0\,.$$

That is, the optimal action $\pi^*(s_1; c)$ switches between 1 and 2 as c moves past c_i. If $\pi^*(s_1; c) = \operatorname{argmax}_a f_a(\psi(c))$ realizes the optimal policy, then f must be > 0 on one side of c_i and < 0 on the other, and since f is continuous it has a root in every neighborhood of c_i; taking $\epsilon \downarrow 0$ and using continuity again gives $f(\psi(c_i)) = 0$. In particular $f \not\equiv 0$, since a vanishing f would leave the two actions tied everywhere and could not induce a policy that switches. By Lemma 9, a non-zero trigonometric polynomial of degree n has at most $2n$ roots in $[0, 2\pi)$, and we have exhibited $2|\mathcal{A}|$ distinct roots c_i. Hence $2n \ge 2|\mathcal{A}|$, i.e. $n \ge |\mathcal{A}|$.

Remark 3 (What the two linear results do and do not compare)**.** *The two parts of Theorem 2 are not two points on one axis. Part **(i)** is an estimation rate in K; Part **(ii)** is a statement about what a class can express, and it holds for every K. What can be compared is the fixed description the planner carries against the degree the classifier is forced to. The model-based planner is described by $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_H$ and $\Theta_1, \ldots, \Theta_H$, a description that does not grow with $|\mathcal{A}|$, and it drives its error to 0 at rate $\tilde{\mathcal{O}}((\kappa\sqrt{K})^{-1})$ with that fixed budget. Part **(ii)** says the classifier's score must instead reach degree $|\mathcal{A}|$ in the task feature, and any class of lower degree carries the resulting bias however much data it is given.*

We deliberately stop at the degree and do not convert it into a count of parameters. It is tempting to say that a degree-n polynomial in $\psi(c) \in \mathbb{R}^{d'}$ has $\binom{n+d'}{d'}$ coefficients and therefore that realizability

costs $\Omega(|\mathcal{A}|^{d'})$ of them, but that counts a dense expansion and our argument does not rule out a sparse or otherwise structured score of the same degree being realized with far fewer free parameters. Our root-counting argument (Lemma 9) constrains only the degree, so that is all we claim. Neither learner is spared an estimation problem here: the model-based learner never sees Θ_h and must recover it from noisy rewards (Lemma 5), while the imitation learner sees optimal action labels the model-based learner never does. The asymmetry behind the tabular result (Remark 2) is gone, and what is left is a question of which representation the structure of the task happens to be simple in.

B.4.3 Technical Lemmas

Here we provide the concentration inequality for vector-valued martingales, which will be used in our analysis of the model-based method in the linear function approximation setting.

Lemma 6 (Hoeffding concentration inequality for vector-valued martingales, Theorem 1 in Abbasi-Yadkori et al. (2011))**.** *Let $\{\eta_\tau\}_{\tau=1}^{\infty}$ be a stochastic process with corresponding filtration $\{\mathcal{F}_\tau\}_{\tau=1}^{\infty}$. Let $\eta_\tau|\mathcal{F}_{\tau-1}$ be zero mean and σ-subGaussian. Let $\{\phi_\tau\}_{\tau=1}^{\infty}$ be a $\mathbb{R}^d$-valued stochastic process with $\phi_\tau \in \mathcal{F}_{\tau-1}$, and $\|\phi_\tau\| \leq 1$. Let $\Lambda_k = \lambda \boldsymbol{I} + \sum_{\tau=1}^{k} \phi_\tau \phi_\tau^\top$. Then for any $0 < \delta < 1$, with probability at least $1-\delta$, we have for all $k > 1$,*

$$\left\| \sum_{\tau=1}^{k} \phi_\tau \eta_\tau \right\|_{\Lambda_k^{-1}}^2 \leq \sigma^2 d \log(1 + k/(d\lambda)) + \log \frac{1}{\delta}$$

When bounding the learning error for the transition model, we need to apply the above bound with respect to the value functions $v \in \mathcal{V}$, where we have the following lemma:

Lemma 7 (Lemma D.4 of Jin et al. (2023))**.** *Let $\{x_\tau\}_{\tau=1}^{\infty}$ be a stochastic process on state space $\mathcal{S}$ with corresponding filtration $\{\mathcal{F}_\tau\}_{\tau=1}^{\infty}$. Let $\{\phi_\tau\}_{\tau=1}^{\infty}$ be a $\mathbb{R}^d$-valued stochastic process with $\phi_\tau \in \mathcal{F}_{\tau-1}$, and $\|\phi_\tau\| \leq 1$. Let $\Lambda_k = \lambda \boldsymbol{I} + \sum_{\tau=1}^{k} \phi_\tau \phi_\tau^\top$. Then for any $\delta > 0$, and any $\tau > 0$ and $v \in \mathcal{V}$ that $\sup_x |V(x)| \leq H$, we have:*

$$\left\| \sum_{\tau=1}^{k} \phi_\tau \left\{ V(x_\tau) - \mathbb{E}\left[V(x_\tau) | \mathcal{F}_{\tau-1} \right] \right\} \right\|_{\Lambda_k^{-1}}^2 \leq 4H^2 \left[\frac{d}{2} \log\left(\frac{k+\lambda}{\lambda} \right) + \log \mathcal{N}_\epsilon + \log \frac{1}{\delta} \right] + \frac{8k^2\epsilon^2}{\lambda}.$$

where $\mathcal{N}_\epsilon$ is the ϵ-covering number of $\mathcal{V}$ with respect to the distance defined as the maximum distance over all states in $x \in \mathcal{S}$: $\operatorname{dist}(V, V') = \sup_x |V(x) - V'(x)|$.

Given the linear MDP assumption, we have the following bound on the covering number of optimal value functions:

Lemma 8. *Fix $R > 0$ and let*

$$\mathcal{V}_R = \left\{ V : \mathcal{S} \to [0, H] \;\middle|\; V(s) = \min\left\{ \max_{a \in \mathcal{A}} \langle \phi(s,a), w \rangle,\ H \right\} \text{ for some } w \in \mathbb{R}^d,\ \|w\| \leq R \right\}.$$

Then the covering number of $\mathcal{V}_R$ satisfies $\log \mathcal{N}_\epsilon \leq d \log(1 + 2R/\epsilon)$. In particular, the optimal value functions of any linear MDP with feature map ϕ satisfying Definition 1 lie in $\mathcal{V}_R$ with $R = 2H\sqrt{d}$, giving $\log \mathcal{N}_\epsilon \leq d \log\left(1 + 4H\sqrt{d}/\epsilon\right)$.

Proof. Using Lemma B.1 of Jin et al. (2023), for any policy π the Q-function of a linear MDP satisfying Definition 1 can be written as $Q_h^\pi(s,a) = \langle \phi(s,a), w_h^\pi \rangle$ with $\|w_h^\pi\| \leq 2H\sqrt{d}$, and $V_h^*(s) = \max_a \langle \phi(s,a), w_h^* \rangle$, which is already in $[0, H]$ so the truncation is inactive. For two members of $\mathcal{V}_R$ with parameters w, w', since truncation and $\max_a$ are 1-Lipschitz,

$$\begin{aligned} \operatorname{dist}(V, V') &= \sup_s |V(s) - V'(s)| \\ &\leq \sup_{s,a} |\langle \phi(s,a), w - w' \rangle| \\ &\leq \|w - w'\|, \end{aligned}$$

using $\|\phi(s,a)\| \leq 1$. Thus the ϵ-covering number of $\mathcal{V}_R$ is bounded by the ϵ-covering number of the Euclidean ball of radius R, which is $\log \mathcal{N}_\epsilon \leq d \log(1 + 2R/\epsilon)$ by Lemma D.5 of Jin et al. (2023). □

Remark 4 (Why the estimated MDP needs the truncated class)**.** *The value functions that appear in the proof of Part* ***(i)****, Theorem 2, are V^*, the optimal value function of the* true *MDP $\mathcal{M}$, and $\widehat{V}^{\widehat{\pi}} = \widehat{V}^*$, the optimal value function of the* estimated *MDP $\widehat{\mathcal{M}}$. Only the first is covered by the plain statement for linear MDPs: the estimate $\widehat{\boldsymbol{\mu}}_h$ is a least-squares solution and is not guaranteed to satisfy the normalization $\|\widehat{\boldsymbol{\mu}}_h(\mathcal{S})\| \le \sqrt{d}$ of Definition 1, so $\widehat{\mathcal{M}}$ need not be a linear MDP in the normalized sense and its optimal value function need not lie in $[0, H]$. We therefore (i) define the planner to compute $\widehat{V}_h = \min\{\max_a \langle \phi(\cdot, a), \widehat{w}_h \rangle, H\}$, clipping each backup to the range of the true value function, and (ii) note that $\|\widehat{\boldsymbol{\mu}}_h(\mathcal{S})\| = \left\|\Lambda_h^{-1} \sum_k \phi(s_h^k, a_h^k)\right\| \le K$ because $\Lambda_h \succeq \boldsymbol{I}$, so $\widehat{V}_h \in \mathcal{V}_R$ with $R = \mathcal{O}(HK)$. Applying Lemma 8 with this R changes $\log \mathcal{N}_\epsilon$ from $d \log\left(1 + 4H\sqrt{d}/\epsilon\right)$ to $\mathcal{O}\left(d \log\left(HK/\epsilon\right)\right)$, which is absorbed by the $\tilde{\mathcal{O}}$ in Lemma 4. Clipping is what makes the Bellman backups in $\widehat{\mathcal{M}}$ type-correct as elements of $[0, H]^{\mathcal{S}}$; without it neither the covering argument nor the simulation lemma below is valid.*

This lemma is used to bound the degree of the optimal function in the hard instance for imitation learning.

Lemma 9. *Consider a n-degree polynomial defined on $[\sin x, \cos x]$:*

$$f(x) = \sum_{u+v \le n} g^{u,v} \sin^u x \cos^v x \,.$$

Then it has at most $2n$ roots in $x \in [0, 2\pi)$.

Proof. We can write $\sin x = \frac{z - z^{-1}}{2i}$ and $\cos x = \frac{z + z^{-1}}{2}$, where $z = \cos x + i \sin x$. Thus, $z^n f$ becomes a $2n$-degree univariate polynomial in z. Since a univariate polynomial with degree $2n$ can have at most $2n$ complex roots, we have at most $2n$ roots for $x \in [0, 2\pi)$. □

B.5 Error Bound of WAP

In this section, we analyze the error of World Action Planner (WAP) and characterize how its performance depends on the initial proposal, global evaluation and correction, the number of local candidate evaluations, world-model prediction, and local candidate ranking.

Status and Scope of this Analysis. What follows is an error *decomposition*, and it is worth being clear about the difference between that and a performance guarantee. The quantities $\epsilon_{\texttt{Agent.Proposal}}$, $\epsilon_{\texttt{Agent.Eval}}$, $\epsilon_{\texttt{Agent.Rank}}$ and $\bar{\epsilon}_{\mathrm{WM}}$ are *defined* to be the errors of the corresponding learned components (Definitions 2, 3 and 4). We neither assume they are small nor bound them by a separate argument, and no line below establishes that a VLM or a video world model is accurate; Section B.6 is where those quantities meet measurements. What the analysis does supply is the rule by which the errors combine: which of them a search radius can swallow, which of them the candidate budget never enters, how the two search stages divide the work between them, and what the candidate budget buys at a given radius. Being an upper bound, it can say that a larger N stops improving the *bound*; it cannot say that a larger N stops improving the system. Only one substantive modeling assumption enters, the local regularity of Q in Assumption 3; the rest of the work is geometry, in Assumption 5 and Lemma 10. A decomposition whose terms were all removable would say nothing, so Proposition 1 exhibits, for each term, an instance of the assumption class in which it alone is responsible for the error. That is a worst-case statement about the class, and it does not claim our tasks are such instances.

One further restriction: all of this concerns a *single* planning decision, with $\hat{a}$ the action WAP commits to at one replanning step and Q the action value there. Errors compounding around a closed loop are outside its scope. Replanning at every step is what keeps $\epsilon_{\texttt{Agent.Proposal}}$ from accumulating in practice, but we do not analyze that here.

Given state s and task description ℓ, we denote the action-value function of taking action $a \in \mathcal{A}$ as $Q(s, a; \ell)$. In the following analysis, when s and ℓ are fixed, we abuse notation and simply write $Q(a)$. We denote an optimal action at the current state as

$$a^* \in \operatorname*{argmax}_{a \in \mathcal{A}} Q(a).$$

The action space $\mathcal{A}$ is continuous. In WAP, a can be interpreted as the low-dimensional goal parameter optimized by the planner, e.g., an end-effector goal pose, which is subsequently converted into an action chunk by the low-level controller.

For simplicity, we assume that the local action space considered by WAP is convex under the distance metric d, as is the case for the Euclidean goal-position space used in our implementation.

Local Regularity of the Action Value. We first assume that actions sufficiently close to an optimal action have similar task values.

Assumption 3 (Local regularity of Q)**.** *For actions in the region considered by WAP,*

$$Q(a^*) - Q(a) \leq L_Q d(a, a^*),$$

where $L_Q > 0$ and d is a distance metric on the action space, e.g., the Euclidean distance between end-effector goal positions.

Assumption 3 is local and does not require the global action-value landscape to be convex or smooth.

Initial Proposal and Global Evaluation. Let $a_0 = \texttt{Agent.Proposal}(s, \ell)$ denote the initial action proposed by WAP. We define the proposal error as

$$\epsilon_{\texttt{Agent.Proposal}} = d(a_0, a^*). \tag{6}$$

WAP then performs a coarse global correction by evaluating the imagined outcome of a_0 using `Agent.Eval`. Let h^{global} denote the maximum correction radius of this global stage.

Define

$$a^*_{\text{global}} \in \operatorname*{arg\,min}_{a:\, d(a,a_0)\leq h^{\text{global}}} d(a, a^*), \tag{7}$$

i.e., a^*_{global} is the point within the global correction radius that is closest to the optimal action.

If the geodesic from a_0 to a^* stays inside the local action space — which holds under the convexity assumption above when a^* itself lies in that space — then the point of that geodesic at distance $\min\left\{h^{\text{global}}, d(a_0, a^*)\right\}$ from a_0 is feasible in (7), and therefore

$$d(a^*_{\text{global}}, a^*) \leq \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}}\right]_+, \tag{8}$$

where $[x]_+ = \max\{x, 0\}$. We write this as an inequality rather than an equality: equality also requires that no point of the ball be closer to a^* than the geodesic point, which holds in Euclidean goal space but not for a general metric, and only the upper bound is used below.

The global correction produced by `Agent.Eval` depends on the imagined rollout of the initial proposal. To separate the intrinsic error of `Agent.Eval` from world-model prediction error, we consider the correction that would be produced if `Agent.Eval` were given the true physical rollout.

Let a_g^{true} denote the corrected action obtained when `Agent.Eval` observes the true rollout, and let a_g denote the corrected action actually obtained from the world-model-predicted rollout.

Definition 2 (`Agent.Eval` error)**.** *Define the value-level error of `Agent.Eval` during global correction as*

$$\epsilon_{\texttt{Agent.Eval}} \coloneqq L_Q d\left(a_g^{\text{true}}, a^*_{\text{global}}\right). \tag{9}$$

Note that this is a definition rather than an assumption. Whatever gap separates the correction `Agent.Eval` produces from a faithful rollout and the best correction the global radius allows, that gap is $\epsilon_{\texttt{Agent.Eval}}$. The factor L_Q puts it in value units, so it can be added to the other terms of the final bound. Had we instead posited an inequality bounding this gap, the theorem would largely be repeating its own premise back. Nothing below asks $\epsilon_{\texttt{Agent.Eval}}$ to be small; Section B.6 discusses how one gets at it empirically.

World-Model Prediction Error. Both `Agent.Eval` and the subsequent local candidate evaluation rely on trajectories generated by the learned world model. We therefore characterize both effects using a common underlying world-model prediction error.

For an action a, let $\tau(a)$ denote its true physical rollout and let $\hat{\tau}(a)$ denote the corresponding rollout predicted by the world model.

Definition 3 (World-model prediction error)**.** *Let*

$$\bar{\epsilon}_{\mathrm{WM}} := \sup_{a} d_\tau\left(\tau(a), \hat{\tau}(a)\right), \tag{10}$$

the supremum being over the rollouts actually considered during one WAP planning step, where d_τ is a task-relevant trajectory distance.

The remaining content about the world model is a genuine assumption, namely that the downstream use of a rollout is stable in d_τ rather than merely measurable in it.

Assumption 4 (Downstream stability of world-model error)**.** *For the global correction produced by* `Agent.Eval`*, we assume*

$$d\left(a_g, a_g^{\mathrm{true}}\right) \le L_{\mathrm{WM}}^{\mathrm{global}} d_\tau\left(\tau(a_0), \hat{\tau}(a_0)\right), \tag{11}$$

where $L_{\mathrm{WM}}^{\mathrm{global}}$ characterizes how world-model prediction error perturbs the global correction.

For local candidate evaluation, let $Q_{\mathrm{WM}}(a)$ denote the ideal task value associated with the world-model-predicted rollout $\hat{\tau}(a)$. We assume

$$|Q_{\mathrm{WM}}(a) - Q(a)| \le L_{\mathrm{WM}}^{\mathrm{local}} d_\tau\left(\tau(a), \hat{\tau}(a)\right) \tag{12}$$

for every candidate evaluated by WAP.

Assumption 4 carries real content: it asks that the two consumers of imagined rollouts degrade gracefully as d_τ grows. It is also the assumption we would expect to fail first. A video world model can be faithful almost everywhere in d_τ and still hallucinate one contact event, and one contact event is enough to reverse a comparison. Hence the choice in Section B.6 to compare the *rankings* that imagined and real rollouts induce rather than a pixel metric: reordering is the failure that matters to a planner, and d_τ is not observable on its own. That comparison probes a necessary condition rather than measuring $L_{\mathrm{WM}}^{\mathrm{local}}\bar{\epsilon}_{\mathrm{WM}}$, as Section B.6 spells out.

The trajectory distance d_τ does not need to correspond to pixel-level reconstruction error. Instead, it can capture task-relevant differences in object motion, contact outcomes, robot pose, or other physical quantities that affect action selection.

By Definition 3 and Assumption 4,

$$d\left(a_g, a_g^{\mathrm{true}}\right) \le L_{\mathrm{WM}}^{\mathrm{global}}\bar{\epsilon}_{\mathrm{WM}}, \tag{13}$$

$$|Q_{\mathrm{WM}}(a) - Q(a)| \le L_{\mathrm{WM}}^{\mathrm{local}}\bar{\epsilon}_{\mathrm{WM}}. \tag{14}$$

We can now bound the residual distance after global evaluation and correction. By the triangle inequality,

$$\begin{aligned} d(a_g, a^*) &\le d\left(a_g, a_g^{\mathrm{true}}\right) + d\left(a_g^{\mathrm{true}}, a_{\mathrm{global}}^*\right) + d(a_{\mathrm{global}}^*, a^*) \\ &\le \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\mathrm{global}}\right]_+ + \frac{\epsilon_{\texttt{Agent.Eval}}}{L_Q} + L_{\mathrm{WM}}^{\mathrm{global}}\bar{\epsilon}_{\mathrm{WM}} \;=:\; \Delta_{\mathrm{global}}. \end{aligned} \tag{15}$$

Thus, the residual after the global stage depends on the initial `Agent.Proposal` error, the global correction radius, the error of `Agent.Eval`, and the world-model prediction error.

Local Candidate Search. After global correction, WAP constructs a finite candidate set

$$\mathcal{A}_N = \{a_1, \ldots, a_N\}$$

around the corrected proposal a_g, where N is the number of local candidates simulated by the world model. Let h^{local} denote the radius covered by the local candidate search.

Rather than specifying the exact geometry of the candidate set, we parameterize it by the resolution it achieves.

Assumption 5 (Local candidate coverage)**.** *The candidate set $\mathcal{A}_N$ is a ϱ_N-net of the local search region: for every action $a \in \mathcal{A}$ satisfying*

$$d(a, a_g) \le h^{\mathrm{local}},$$

there exists a candidate $\tilde{a} \in \mathcal{A}_N$ with $d(a, \tilde{a}) \le \varrho_N$. When the candidates form a grid or a uniformly random sample of the d_a-dimensional local region, the resolution obeys the standard covering rate

$$\varrho_N \le c_{d_a} h^{\text{local}} N^{-1/d_a} \tag{16}$$

for a constant $c_{d_a} > 0$ depending only on the geometry and dimension d_a of the local action space.

Assumption 5 captures the tradeoff between the number of candidate evaluations and the resolution of the local search. Increasing N gives finer coverage, while increasing h^{local} under a fixed candidate budget gives coarser coverage. We keep ϱ_N and the rate (16) apart because they are different claims, and our own implementation makes the first but not the second. WAP evaluates the five-point cross $\{(x, y), (x \pm \delta, y), (x, y \pm \delta)\}$ (Appendix C.2.3). That is a ϱ_N-net of the radius-δ region with $\varrho_N = \Theta(\delta)$, not $\Theta(\delta N^{-1/2})$: the cross probes four directions rather than tiling the disc. The rate (16) describes a dense grid, and the sampling baselines we compare against; for the cross all one can say is $\varrho_N \le h^{\text{local}}$, with refinement arriving through replanning rather than through a larger N inside a single step. Everything below is written in terms of ϱ_N, and we substitute (16) only where the dependence on N is the point at issue.

We next bound the distance between the optimal continuous action and the closest candidate generated by WAP.

Lemma 10 (Candidate coverage error)**.** *Under Assumption 5,*

$$\min_{a \in \mathcal{A}_N} d(a, a^*) \le \varrho_N + [d(a_g, a^*) - h^{\text{local}}]_+. \tag{17}$$

Consequently, with Δ_{global} as in (15),

$$\begin{aligned} \min_{a \in \mathcal{A}_N} d(a, a^*) &\le \varrho_N + \left[\Delta_{\text{global}} - h^{\text{local}}\right]_+ && (18) \\ &\le \varrho_N + \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}\right]_+ \\ &\quad + \frac{\epsilon_{\texttt{Agent.Eval}}}{L_Q} + L_{\text{WM}}^{\text{global}} \bar{\epsilon}_{\text{WM}}. && (19) \end{aligned}$$

The intermediate form (18) is the sharper of the two and the one that says something structural: the local radius absorbs the whole *global-stage residual, the* `Agent.Eval` *and world-model contributions along with the uncorrected proposal error. That information is what gets thrown away on the way to (19), which is why the additive form keeps charging $\epsilon_{\texttt{Agent.Eval}}$ in full even in cases where a generous h^{local} has already covered it.*

Proof. We consider two cases.

If

$$d(a_g, a^*) \le h^{\text{local}},$$

then a^* lies inside the local search region. By Assumption 5, there exists a candidate $a \in \mathcal{A}_N$ satisfying

$$d(a, a^*) \le \varrho_N.$$

Otherwise,

$$d(a_g, a^*) > h^{\text{local}}.$$

Since the local action space is convex, consider a point a_{local} on the line segment from a_g toward a^* such that

$$d(a_g, a_{\text{local}}) = h^{\text{local}}.$$

Then

$$d(a_{\text{local}}, a^*) = d(a_g, a^*) - h^{\text{local}}.$$

By Assumption 5, there exists a candidate $a \in \mathcal{A}_N$ satisfying

$$d(a, a_{\text{local}}) \le \varrho_N.$$

Therefore, by the triangle inequality,

$$\begin{aligned} d(a, a^*) &\le d(a, a_{\text{local}}) + d(a_{\text{local}}, a^*) \\ &\le \varrho_N + d(a_g, a^*) - h^{\text{local}}. \end{aligned}$$

Combining the two cases gives

$$\min_{a \in \mathcal{A}_N} d(a, a^*) \le \varrho_N + [d(a_g, a^*) - h^{\text{local}}]_+.$$

Substituting Eq. 15 and using that $x \mapsto [x - h^{\text{local}}]_+$ is non-decreasing gives (18):

$$\min_{a \in \mathcal{A}_N} d(a, a^*) \le \varrho_N + \left[\left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} \right]_+ - h^{\text{local}} + \frac{\epsilon_{\texttt{Agent.Eval}}}{L_Q} + L_{\text{WM}}^{\text{global}} \bar{\epsilon}_{\text{WM}} \right]_+ .$$

For the weaker additive form, use

$$[x + y]_+ \le [x]_+ + y$$

for $y \ge 0$, together with

$$\left[\left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} \right]_+ - h^{\text{local}} \right]_+ = \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}} \right]_+ ,$$

which proves Eq. 19. □

Lemma 10 reveals the complementary roles of the global and local search stages. Ignoring learned-component errors, the two stages can cover proposal error as large as

$$h^{\text{global}} + h^{\text{local}},$$

while the approximation error within the covered region is ϱ_N, which decreases as N^{-1/d_a} for a dense grid.

Let

$$a^*_{\mathcal{A}_N} \in \operatorname*{argmax}_{a \in \mathcal{A}_N} Q(a)$$

denote the best action among the N generated candidates.

Let $a_{\text{near}} \in \mathcal{A}_N$ denote a candidate closest to a^*. Since $a^*_{\mathcal{A}_N}$ is the best candidate according to the true action value,

$$Q(a^*_{\mathcal{A}_N}) \ge Q(a_{\text{near}}).$$

By Assumption 3 and Lemma 10, provided a^* and the candidates all lie in the region where Assumption 3 applies,

$$\begin{aligned} Q(a^*) - Q(a^*_{\mathcal{A}_N}) \le & L_Q \varrho_N \\ & + L_Q \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}} \right]_+ \\ & + \epsilon_{\texttt{Agent.Eval}} + L_Q L_{\text{WM}}^{\text{global}} \bar{\epsilon}_{\text{WM}}. \end{aligned} \tag{20}$$

The first term characterizes the finite-search error caused by evaluating only N local candidates, while the second term captures proposal error that remains outside the combined global and local search range.

World-Model Evaluation and `Agent.Rank`. For each of the N local candidates, WAP predicts its physical outcome using the world model and uses `Agent.Rank` to select among the predicted outcomes.

Let

$$a^*_{\text{WM}} \in \operatorname*{argmax}_{a \in \mathcal{A}_N} Q_{\text{WM}}(a)$$

denote the best candidate according to an ideal evaluation of the world-model-predicted rollouts.

Since `Agent.Rank` performs comparative ranking rather than necessarily predicting a calibrated scalar value, we characterize its error directly through ranking regret.

Definition 4 (`Agent.Rank` error)**.** *Let $\hat{a}$ denote the candidate selected by* `Agent.Rank`*. Define its ranking regret among the locally imagined candidates as*

$$\epsilon_{\texttt{Agent.Rank}} \coloneqq Q_{\mathrm{WM}}(a^*_{\mathrm{WM}}) - Q_{\mathrm{WM}}(\hat{a}) \ \geq 0. \tag{21}$$

Again a definition rather than a hypothesis. We phrase it as a regret rather than a calibration error because `Agent.Rank` only ever compares candidates against one another; it never emits a number one could be calibrated about.

We next bound the error introduced by world-model prediction and `Agent.Rank` during candidate selection.

Lemma 11 (Candidate selection error)**.** *Under Definitions 3 and 4 and Assumption 4,*

$$Q(a^*_{\mathcal{A}_N}) - Q(\hat{a}) \leq 2L^{\mathrm{local}}_{\mathrm{WM}}\bar{\epsilon}_{\mathrm{WM}} + \epsilon_{\texttt{Agent.Rank}}. \tag{22}$$

Proof. From Eq. 14,

$$Q(a^*_{\mathcal{A}_N}) \leq Q_{\mathrm{WM}}(a^*_{\mathcal{A}_N}) + L^{\mathrm{local}}_{\mathrm{WM}}\bar{\epsilon}_{\mathrm{WM}}.$$

Since a^*_{WM} maximizes Q_{WM} over $\mathcal{A}_N$,

$$Q_{\mathrm{WM}}(a^*_{\mathcal{A}_N}) \leq Q_{\mathrm{WM}}(a^*_{\mathrm{WM}}).$$

By Definition 4,

$$Q_{\mathrm{WM}}(a^*_{\mathrm{WM}}) \leq Q_{\mathrm{WM}}(\hat{a}) + \epsilon_{\texttt{Agent.Rank}}.$$

Applying Eq. 14 again gives

$$Q_{\mathrm{WM}}(\hat{a}) \leq Q(\hat{a}) + L^{\mathrm{local}}_{\mathrm{WM}}\bar{\epsilon}_{\mathrm{WM}}.$$

Combining the above inequalities gives

$$Q(a^*_{\mathcal{A}_N}) - Q(\hat{a}) \leq 2L^{\mathrm{local}}_{\mathrm{WM}}\bar{\epsilon}_{\mathrm{WM}} + \epsilon_{\texttt{Agent.Rank}},$$

which proves the result. □

To simplify the final result, we define the total value-level error caused by world-model prediction as

$$\epsilon_{\mathrm{WM}} = \left(L_Q L^{\mathrm{global}}_{\mathrm{WM}} + 2L^{\mathrm{local}}_{\mathrm{WM}}\right)\bar{\epsilon}_{\mathrm{WM}}. \tag{23}$$

This quantity combines the two ways in which world-model prediction error affects WAP: perturbing the global correction produced by `Agent.Eval` and distorting the relative values of the locally imagined candidates used by `Agent.Rank`.

We are now ready to state the overall error bound.

Theorem 6 (Error bound of WAP)**.** *Under Assumptions 3, 4 and 5, with the error quantities of Definitions 2, 3 and 4, the action $\hat{a}$ selected by WAP from N local candidates satisfies the sharp bound*

$$Q(a^*) - Q(\hat{a}) \leq L_Q\varrho_N + L_Q\left[\Delta_{\mathrm{global}} - h^{\mathrm{local}}\right]_+ + 2L^{\mathrm{local}}_{\mathrm{WM}}\bar{\epsilon}_{\mathrm{WM}} + \epsilon_{\texttt{Agent.Rank}}\,, \tag{24}$$

and consequently, relaxing $\left[\Delta_{\mathrm{global}} - h^{\mathrm{local}}\right]_+$ and substituting the grid rate (16),

$$\begin{aligned} Q(a^*) - Q(\hat{a}) \leq\ & \underbrace{L_Q c_{d_a} h^{\mathrm{local}} N^{-1/d_a}}_{\textit{finite candidate search error}} \\ & + \underbrace{L_Q\left[\epsilon_{\texttt{Agent.Proposal}} - h^{\mathrm{global}} - h^{\mathrm{local}}\right]_+}_{\texttt{Agent.Proposal}\ \textit{coverage error}} \\ & + \underbrace{\epsilon_{\texttt{Agent.Eval}}}_{\texttt{Agent.Eval}\ \textit{error}} + \underbrace{\epsilon_{\mathrm{WM}}}_{\textit{world-model error}} + \underbrace{\epsilon_{\texttt{Agent.Rank}}}_{\texttt{Agent.Rank}\ \textit{error}}. \end{aligned} \tag{25}$$

Proof. We decompose the action-selection error into the error caused by restricting the continuous action space to the N generated candidates and the error caused by selecting among these candidates:

$$Q(a^*) - Q(\hat{a}) = \left[Q(a^*) - Q(a^*_{\mathcal{A}_N})\right] + \left[Q(a^*_{\mathcal{A}_N}) - Q(\hat{a})\right].$$

For the sharp form (24), Assumption 3 applied to the candidate nearest a^* together with (18) gives $Q(a^*) - Q(a^*_{\mathcal{A}_N}) \le L_Q \varrho_N + L_Q \left[\Delta_{\text{global}} - h^{\text{local}}\right]_+$, and adding Lemma 11 yields (24). The additive form follows by relaxing the same terms, which we now do explicitly.

From Eq. 20 and (16),

$$\begin{aligned} Q(a^*) - Q(a^*_{\mathcal{A}_N}) \le & L_Q c_{d_a} h^{\text{local}} N^{-1/d_a} \\ & + L_Q \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}\right]_+ \\ & + \epsilon_{\texttt{Agent.Eval}} + L_Q L^{\text{global}}_{\text{WM}} \bar{\epsilon}_{\text{WM}}. \end{aligned}$$

From Lemma 11,

$$Q(a^*_{\mathcal{A}_N}) - Q(\hat{a}) \le 2L^{\text{local}}_{\text{WM}} \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}.$$

Adding the two inequalities gives

$$\begin{aligned} Q(a^*) - Q(\hat{a}) \le & L_Q c_{d_a} h^{\text{local}} N^{-1/d_a} \\ & + L_Q \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}\right]_+ + \epsilon_{\texttt{Agent.Eval}} \\ & + \left(L_Q L^{\text{global}}_{\text{WM}} + 2L^{\text{local}}_{\text{WM}}\right) \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}. \end{aligned}$$

Using the definition of ϵ_{WM} in Eq. 23 proves Eq. 25. □

Eq. 25 directly associates the error terms with the components of Algorithm 1. The initial error $\epsilon_{\texttt{Agent.Proposal}}$ does not need to be small enough for the initial proposal itself to be optimal. Instead, it only needs to lie within the combined global and local search reach. In particular, when

$$\epsilon_{\texttt{Agent.Proposal}} \le h^{\text{global}} + h^{\text{local}}, \tag{26}$$

the proposal coverage error vanishes and the bound becomes

$$Q(a^*) - Q(\hat{a}) \le L_Q c_{d_a} h^{\text{local}} N^{-1/d_a} + \epsilon_{\texttt{Agent.Eval}} + \epsilon_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}. \tag{27}$$

Therefore, once the global and local search ranges are sufficient to cover the proposal error, increasing the number of candidates reduces the finite-search approximation error at rate

$$N^{-1/d_a}.$$

Equivalently, to make the finite candidate search error no larger than ϵ_{search}, it is sufficient to choose

$$N \ge \left(\frac{L_Q c_{d_a} h^{\text{local}}}{\epsilon_{\text{search}}}\right)^{d_a}. \tag{28}$$

Eq. 28 directly relates planning accuracy to the number of world-model evaluations. Each local candidate requires one world-model rollout, so the local planning cost grows linearly with N. Meanwhile, the number of candidates required for a fixed approximation error grows exponentially with the local action-space dimension d_a. This motivates WAP to search over a low-dimensional goal space rather than directly sampling high-dimensional action sequences.

Coarse-to-Fine Search Tradeoff. The two search radii h^{global} and h^{local} determine how WAP distributes search between coarse correction and fine local evaluation. Increasing h^{global} allows `Agent.Eval` to correct a larger `Agent.Proposal` error and therefore reduces the local search radius required for sufficient coverage. For fixed N, reducing h^{local} directly decreases the finite candidate search error

$$L_Q c_{d_a} h^{\text{local}} N^{-1/d_a}.$$

By the sharp form (24), the smallest local radius that maintains coverage is on the scale of the entire global-stage residual,

$$h^{\text{local}} \approx \Delta_{\text{global}} = \left[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}}\right]_+ + \frac{\epsilon_{\texttt{Agent.Eval}}}{L_Q} + L_{\text{WM}}^{\text{global}} \bar{\epsilon}_{\text{WM}}\,, \tag{29}$$

not merely of the uncorrected proposal error: local search also has to absorb the mistakes made by `Agent.Eval` and by the world model during the global stage.

Therefore, global evaluation is useful not only because `Agent.Eval` directly corrects the initial proposal, but also because it reduces the continuous region that must subsequently be covered by the N candidates. Under a fixed number of world-model evaluations, concentrating the local candidate set within a smaller residual region produces finer effective action resolution. This provides a theoretical interpretation of WAP's coarse-to-fine structure: `Agent.Proposal` provides the initial action hypothesis, `Agent.Eval` performs coarse correction using the imagined outcome, and `Agent.Rank` performs fine-grained discrimination among the locally imagined candidates.

Tightness. A decomposition is cheap if its terms are merely sufficient; one can always add more of them. The proposition below rules that out for (24) by treating each term in turn: for each there is an instance, meeting every assumption, on which the other three vanish and the gap is of the order of the term in question.

Proposition 1 (No term of (24) is removable)**.** *Let the local action space be $\mathbb{R}^{d_a}$ with the Euclidean metric and let $L_Q > 0$. Then:*

(i) (finite search) *For every $h^{\text{local}} > 0$ and $N \geq 1$ there is an instance with $\Delta_{\text{global}} = h^{\text{local}}$, $\bar{\epsilon}_{\text{WM}} = 0$ and $\epsilon_{\texttt{Agent.Rank}} = 0$ on which $Q(a^*) - Q(\hat{a}) = \Omega\left(L_Q \varrho_N\right)$.*

(ii) (proposal coverage) *For every $\epsilon_{\texttt{Agent.Proposal}} > h^{\text{global}} + h^{\text{local}}$ there is an instance with $\epsilon_{\texttt{Agent.Eval}} = \bar{\epsilon}_{\text{WM}} = \epsilon_{\texttt{Agent.Rank}} = 0$ and ϱ_N arbitrarily small on which $Q(a^*) - Q(\hat{a}) = L_Q\left(\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}\right)$.*

(iii) (global evaluation) *For every $\epsilon_{\texttt{Agent.Eval}} > L_Q h^{\text{local}}$ there is an instance with $\epsilon_{\texttt{Agent.Proposal}} \leq h^{\text{global}}$, $\bar{\epsilon}_{\text{WM}} = \epsilon_{\texttt{Agent.Rank}} = 0$ and ϱ_N arbitrarily small on which $Q(a^*) - Q(\hat{a}) = \epsilon_{\texttt{Agent.Eval}} - L_Q h^{\text{local}}$.*

(iv) (local world-model error and ranking) *For every $\bar{\epsilon}_{\text{WM}} \geq 0$ and $\epsilon_{\texttt{Agent.Rank}} \geq 0$ there is an instance with $a^* \in \mathcal{A}_N$, hence $\varrho_N = 0$ and $\Delta_{\text{global}} = 0$, on which $Q(a^*) - Q(\hat{a}) = 2L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}$.*

Proof. In all four cases take $Q(a) = -L_Q \left\|a - a^*\right\|$, which satisfies Assumption 3 with equality.

(i) Set $h^{\text{global}} = 0$, $\epsilon_{\texttt{Agent.Eval}} = 0$ and $\bar{\epsilon}_{\text{WM}} = 0$, so $\Delta_{\text{global}} = \epsilon_{\texttt{Agent.Proposal}}$, and choose $\epsilon_{\texttt{Agent.Proposal}} = h^{\text{local}}$; then $a_g = a_0$ lies at distance exactly h^{local} from a^* and $[\Delta_{\text{global}} - h^{\text{local}}]_+ = 0$. Let $\mathcal{A}_N$ be a grid of resolution ϱ_N in $B(a_g, h^{\text{local}})$ translated so that a^* falls at the centre of a grid cell, so $\min_{a \in \mathcal{A}_N} \|a - a^*\| \geq \varrho_N/2$. With $\bar{\epsilon}_{\text{WM}} = 0$ and $\epsilon_{\texttt{Agent.Rank}} = 0$ we have $\hat{a} = a^*_{\mathcal{A}_N}$, and $Q(a^*) - Q(\hat{a}) \geq L_Q \varrho_N/2$.

(ii) Set $\epsilon_{\texttt{Agent.Eval}} = \bar{\epsilon}_{\text{WM}} = 0$, so $a_g = a^*_{\text{global}}$ lies on the segment $[a_0, a^*]$ at distance $\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} > h^{\text{local}}$ from a^*. Every point of $B(a_g, h^{\text{local}})$ is then at distance at least $\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}$ from a^*, and taking $\mathcal{A}_N$ to be a net of that ball with $\varrho_N \downarrow 0$ and a perfect ranker gives the claim.

(iii) Set $\epsilon_{\texttt{Agent.Proposal}} \leq h^{\text{global}}$, so $a^*_{\text{global}} = a^*$, and let `Agent.Eval` return $a_g^{\text{true}} = a_g$ at distance $\epsilon_{\texttt{Agent.Eval}}/L_Q > h^{\text{local}}$ from a^*, which is consistent with Definition 2. The remainder is as in (ii).

(iv) Let $\mathcal{A}_N = \{a^*, a'\}$ with $\|a' - a^*\| = \left(2L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}\right)/L_Q$, so $Q(a^*) - Q(a') = 2L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}$. Let the world model under-value a^* and over-value a' by $L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}}$ each, which saturates (12); then $Q_{\text{WM}}(a') = Q_{\text{WM}}(a^*) + \epsilon_{\texttt{Agent.Rank}}$, so $a^*_{\text{WM}} = a'$. Letting `Agent.Rank` return $\hat{a} = a'$ realizes ranking regret 0 when $\epsilon_{\texttt{Agent.Rank}} = 0$; for $\epsilon_{\texttt{Agent.Rank}} > 0$, extend $\mathcal{A}_N$ by a third candidate at the corresponding value so that the ranker's regret is exactly $\epsilon_{\texttt{Agent.Rank}}$. In either case $Q(a^*) - Q(\hat{a}) = 2L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}} + \epsilon_{\texttt{Agent.Rank}}$. □

Case *(iii)* also marks the limit of what the bound claims. The `Agent.Eval` term survives in the bound only once $\epsilon_{\texttt{Agent.Eval}} > L_Q h^{\text{local}}$. Below that threshold the sharp form (24) folds it into the local radius while the additive form (25) keeps charging for it. Quote the additive form for intuition, then, and the sharp one when tightness is at stake.

B.6 Empirical Instantiation of the WAP Bound

A bound is only informative to the extent that its symbols name things one can observe, so we walk through (24) term by term and attach each to the configuration or the measurement it corresponds to in the rest of the paper. We make no claim to identify the constants L_Q, $L_{\text{WM}}^{\text{global}}$ and $L_{\text{WM}}^{\text{local}}$ individually. What can be observed, and what we report, are the composite value-level quantities they appear inside.

Search Dimension d_a and Radii $h^{\text{global}}, h^{\text{local}}$. WAP searches over the end-effector goal pose rather than the low-level action sequence, which puts $d_a = 2$ for the planar position search and $d_a = 3$ once yaw is included (Appendix C.2.3). Since d_a sits in the exponent of (28), this is the concrete reason a candidate budget that would be hopeless for sampling an action chunk is ample here. Neither radius is tuned; both come from the geometry of the object, with h^{local} at roughly half the object size and h^{global} at roughly its height. Appendix D.4.3 finds performance insensitive to the exact values, which is what (24) would lead one to expect in the covered regime $\Delta_{\text{global}} \leq h^{\text{local}}$: there the bound drops h^{global} altogether and touches h^{local} only through ϱ_N.

Candidate Budget N and Resolution ϱ_N. WAP evaluates $N = 6$ candidates per planning step against 8 and 16 for the sampling baseline (Table 4). Comparing the raw budgets misses the point; what matters is N at a given ϱ_N. A baseline sampling the same neighborhood isotropically needs $\Theta((h/\varrho)^{d_a})$ samples to reach resolution ϱ, while WAP has already shrunk h in the global stage before the local stage spends anything. Table 4 bears this out: at $N = 6$ with world-model rollouts WAP reaches $72.0/82.0/78.0\%$, ahead of VLMPC handed 8 or 16 *ground-truth simulator* rollouts. The $N = 1$ row, global optimization only, at $58.0/66.0/65.0\%$, is the configuration with the local stage switched off, and so isolates what the $L_Q \varrho_N$ term is worth.

Proposal Coverage. We cannot measure $L_Q[\epsilon_{\texttt{Agent.Proposal}} - h^{\text{global}} - h^{\text{local}}]_+$ directly, since $\epsilon_{\texttt{Agent.Proposal}}$ is a distance to an optimum nobody knows. Its qualitative signature is testable, however, and the zero-shot comparison against the vision-language planner baseline in Section 5 is where it shows: a planner that emits a_0 and executes it pays the full $L_Q \epsilon_{\texttt{Agent.Proposal}}$, while WAP pays only for whatever lies outside the combined reach of its two stages.

World-Model and Ranking Error. The terms $2L_{\text{WM}}^{\text{local}} \bar{\epsilon}_{\text{WM}}$ and $\epsilon_{\texttt{Agent.Rank}}$ are the ones N never enters, so the bound stops improving in N once they dominate. The ranking-agreement study of Appendix D.1.2 is the nearest evidence we have about them, and it is worth saying precisely what it does and does not show. The protocol asks one agent to pick the best of a candidate group twice, once from imagined and once from ground-truth rollouts, and records how often the two picks coincide: $82/90/78\%$ for our world model against $52/64/44\%$ for the cross-attention baseline (Table 6).

That quantity is the stability of one agent's choice when imagination is substituted for reality. It is not the regret of that choice, and it does not measure either error term. The reference pick is the agent's own selection on true rollouts rather than a^*, so it carries $\epsilon_{\texttt{Agent.Rank}}$ itself; the two passes can agree on a candidate that is wrong in both, which is compatible with $\epsilon_{\texttt{Agent.Rank}}$ being large. Agreement is also a $0/1$ event while the terms of (24) are in value units, so a flip between two near-equal candidates and a flip between a good candidate and a useless one enter the statistic identically, and no bound on $Q(a^*) - Q(\hat{a})$ follows from the rate. Nor does the protocol touch Q_{WM}, the quantity $\epsilon_{\texttt{Agent.Rank}}$ is defined against. What the study does support is a necessary condition: a world model that frequently reorders the agent's candidates cannot support planning on imagination in place of reality, and ours reorders them less often than the baseline does.

The direction of (24) deserves the same care. It is an upper bound, so a term in it that does not shrink with N shows that the bound stops improving, not that the error of the deployed system stops improving; a loose bound would behave the same way. Proposition 1 rules out dropping the term from a general bound over the assumption class, which is a worst-case statement and not evidence about our tasks. That the baseline in Table 4 does not close its gap as N goes from 8 to 16 is consistent with the picture, and we claim no more than consistency.

What Remains Uninstantiated. Two quantities escape every experiment we run. The first is $\epsilon_{\texttt{Agent.Eval}}$, which stays confounded with $\epsilon_{\texttt{Agent.Proposal}}$ throughout: ablating the global stage removes the correction and its error together, and prying them apart would take an oracle global corrector we do not have. The second is the set of constants L_Q, $L_{\mathrm{WM}}^{\mathrm{global}}$, $L_{\mathrm{WM}}^{\mathrm{local}}$ and c_{d_a}, which we never measure. Read (24), then, as an account of which quantities matter and how they combine, and not as a numerical prediction of success rates.

C Implementation Details

C.1 World Model

Here we provide the architecture details of our pose-image conditioned world model. Following Song et al. (2025), we add independent noise levels to the history frames with probability 0.5 and retain clear history frames with probability 0.5. For the future frames, we add uniform random noise and denoise them uniformly, without any auto-regressive schedule to speed up inference. We adopt multi-view prediction by concatenating 4 camera views into a grid, so that the model can infer the 3D positions of the robot. Specifically, for table top manipulators, we use wrist-view, front-view, side-view and top-view. For bi-manual robots, we use two wrist-views and the front-view and the side-view. For Robocasa, we utilize the 2 side-views, together with a wrist-view and top-view. Since the robot is mobile in Robocasa, we fix the top view camera, so we can infer the 2D movements from the top-view pose. We share the position embeddings of the frame tokens and pose image tokens, and only compute the flow matching loss of the frame tokens.

C.1.1 Pose-Image Conditioning: Decoupling Physical Simulation and Neural Imagination

While modern video foundation models generate high-fidelity and physically plausible videos (Wiedemer et al., 2025), they often lack the dynamics necessary to predict the specific physical consequences of low-dimensional robot actions. These outcomes are inherently dependent on the robot's unique kinematics and control parameters. To address this, we map abstract action vectors into visual joint pose skeletons, providing a semantically rich action representation. We compute anticipated joint positions by executing forward dynamics of the robot model. In practice, physical simulators such as MuJoCo, PyBullet, or Pinocchio can be employed to achieve these computations efficiently, given standard URDF or MJCF configurations. The simulator only have access to the robot model without any access to scene geometry and object states. Crucially, these pose images are not intended to be exact physical predictions. Instead, they serve as a mapping of actions, where the world model learns accurate real world physical dynamics during training on trajectories. Notably, we do not access any future ground truth robot joint states in both training and inference, relying on computation from the initial states only.

C.2 World Action Planner

Here we provide additional details about our planning framework, including the robot controller, optional policy, and search process.

C.2.1 Robot Controller

We use the robot controller as an inverse-dynamics module that navigates the robot to a desired goal pose. In our simulation experiments, we implement the controller as a diffusion policy with a UNet backbone that takes the current and target end-effector poses as input and generates executable action chunks. We train the controller on completely random trajectories without any task-specific information. For the real-robot experiment with a Franka Panda robot, we interpolate the goal position into waypoints spaced 10 mm apart and directly use the robot's low-level controller to navigate through these waypoints without any neural networks.

C.2.2 Optional Policy

For compositional task generalization and new-layout generalization, we additionally use an optional policy to perform local manipulation, such as grasping. We implement this policy as a diffusion transformer, with details provided in Appendix D.2.1. We emphasize that this lightweight diffusion policy is trained using the same amount of expert data as the baseline generalist policies, such as $\pi_{0.5}$ and Cosmos-Policy, for compositional task generalization, and substantially less expert data for new-layout generalization. When using the policy, we further imagine its rollout after each candidate action to verify that the resulting state allows the policy to successfully complete the subsequent manipulation.

C.2.3 Search Details

To efficiently search for plausible actions using our world model, we adopt a global-local search strategy inspired by the framework of Zhang et al. (2025). This approach follows a coarse-to-fine hierarchy: we use a larger search scale for global action optimization guided by agent feedback, followed by a finer grid scale for local search. The prompt templates for agent action proposal, feedback-driven optimization, and candidate ranking are provided in Figs. 20, 21, and 22, respectively.

During global action optimization, we prompt the agent to propose directional modifications rather than precise coordinates. For `GridSearch`, we search the planar coordinates of the target gripper position over the set $\{(x, y), (x \pm \delta, y), (x, y \pm \delta)\}$. We set the search scale δ for local grid search and global optimization to approximately half the object size and object height in magnitude, respectively, thereby encouraging physically plausible candidate generation. As shown in Appendix D.4.3, our method is insensitive to the exact values of these hyperparameters. In practice, the physical dimensions of objects can either be obtained directly or estimated from camera observations using computer vision models. We emphasize that these hyperparameter choices are physically grounded and substantially more interpretable than the sampling variances used in MPPI-based methods. When modifying the orientation, we search the target yaw angle over the set $\{\psi, \psi + 45°, \psi - 45°\}$. Importantly, our search operates in the low-dimensional goal-pose space rather than over individual low-level actions, resulting in a substantially more compact search space.

D Experiment Details

In this section, we provide detailed setup and more qualitative results for our experiments and ablations.

D.1 World Model

Here we provide the details of the action-conditioned world model experiments in Section 5.1.

D.1.1 Setup

To simulate the action sampling typical of the widely used Model Predictive Path Integral (MPPI) algorithm and to improve model generalization across novel exploratory actions, we augment expert demonstrations with Gaussian noise prior to trajectory replay. Specifically, we generate 100 trajectories per task: 40 from clean demonstrations, 40 with Gaussian noise added to the actions ($\sigma = 0.08$), and 20 with higher-intensity noise ($\sigma = 0.16$). We observe that model-based exploration primarily necessitates spatial reasoning regarding the end-effector's location; thus, for the high-variance noise ($\sigma = 0.16$), we limit perturbations to the first three dimensions of the action vector (corresponding to target gripper positions) while maintaining fixed orientations. Incorporating such varied trajectories, including potential failures, is a standard practice in the literature (Guo et al., 2025; Jain et al., 2025) for improving world model robustness. The model is trained with a global batch size of 64 on 16 H100 GPUs. For the cross-embodiment baselines, we adopt cross-attention as our base action-conditioning mechanism due to its superior performance (Table 1), and then apply the cross-embodiment-unification methods. Since our conditioning mechanism does not require any text input, we use a dummy text embedding for the `Wan`-T2V model for all inputs.

Table 5: **World model design ablations.** Each cell reports PSNR↑ / LPIPS↓ on LIBERO-Long, which evaluates generalization to novel action sequences. Both ablations share the same model and training data, and differ only in the component under study; the second row of each subtable is our final configuration.

(a) Sparse history frames.

History FPS	wrist-view	third-view
20 FPS (dense)	15.68 / 0.326	22.04 / 0.094
7 FPS (sparse)	**15.98 / 0.320**	**22.14 / 0.093**

(b) Action conditioning representation.

Conditioning	wrist-view	third-view
Joint position	14.62 / 0.346	18.25 / 0.138
Pose image	**15.98 / 0.320**	**22.14 / 0.093**

World Model Design. We ablate two design choices of our world model in Table 5, both evaluated on LIBERO-Long. First, we use a sampling rate of 7 FPS for history frames, in contrast to the 20 FPS

used for prediction frames and environment control. This sparse history allows for a longer temporal window without increasing the total number of history frames. Additionally, utilizing sparse frames helps mitigate spurious correlations rooted in local dynamics between continuous frames, forcing the model to learn the global environment configuration rather than relying on short-term motion patterns. As shown in Table 5a, this approach improves performance when generalizing to novel action sequences. Second, we compare our pose-image conditioning against cross-attention conditioning on joint positions computed through the same forward dynamics. As shown in Table 5b, given the same joint positions, rendering them as pose images generalizes substantially better, indicating that the benefit comes from the spatial rendering rather than from the forward-dynamics information itself.

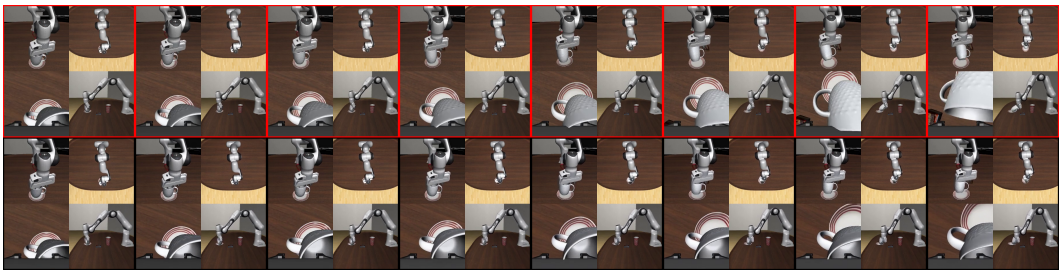
D.1.2 Additional Results

Ranking Agreement. Beyond pixel-level fidelity, what a planner needs from imagined rollouts is that they preserve the *ordering* the VLM agent would assign to candidate actions had it observed the real outcomes. For each generalization setting, we collect 50 groups of candidate actions from our planner experiments and ask the same VLM agent to select the best candidate twice: once from world-model imagination, and once from ground-truth simulator rollouts. As shown in Table 6, our model preserves the VLM's choice far more often than the cross-attention baseline, indicating that its advantage in pixel-space metrics translates into the ranking fidelity that planning depends on. We read this as a necessary condition rather than as a measure of action quality. The reference is the agent's own choice on true rollouts, not the optimal action, so the two passes may agree on a candidate that is wrong in both; and agreement is recorded as a binary event, so a flip between two near-equal candidates counts the same as a flip between a good candidate and a useless one. The comparison shows that our world model reorders candidates less often, not by how much the reorderings cost.

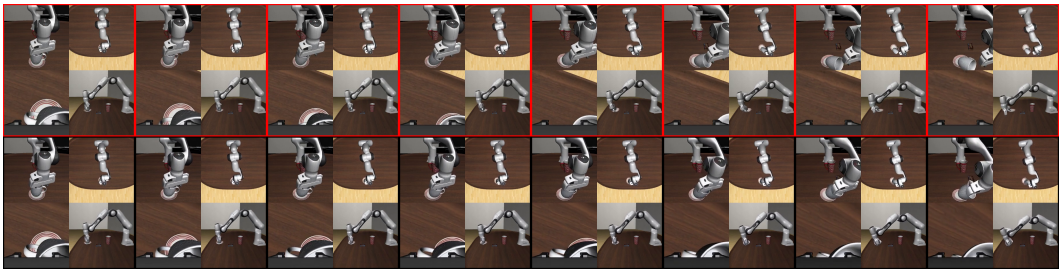
Table 6: **Agreement between imagined and ground-truth candidate rankings.** For each setting we report the percentage of 50 candidate groups on which the VLM agent selects the same best action from world-model imagination as from ground-truth simulator rollouts. Higher is better.

World model	Compositional task	New layout	Novel task
Ctrl-World (Guo et al., 2025)	52	64	44
Pose-image conditioned (ours)	**82**	**90**	**78**

Qualitative Results. We first evaluate the generalization capabilities of our pose-image conditioned world model by demonstrating its ability to predict the physical consequences of novel actions that lead to failures not present in the training set. As illustrated in Fig. 5, the world model successfully imagines the physical outcomes of actions that knock a mug over in various directions, such as backward or to the right. Notably, while Gaussian noise was incorporated during training, the noise levels were minimal and insufficient to produce such collisions; therefore, these "knocking over" trajectories are strictly out-of-distribution, highlighting our model's capacity for zero-shot physical simulation and generalization. We further demonstrate the cross-embodiment modeling capabilities of our framework in Fig. 6. Our pose-image conditioned model successfully captures the dynamics across diverse robotic platforms, effectively generalizing from high-degree-of-freedom humanoid hands to mobile manipulators with side view by leveraging the embodiment-agnostic nature of the pose-image representation.

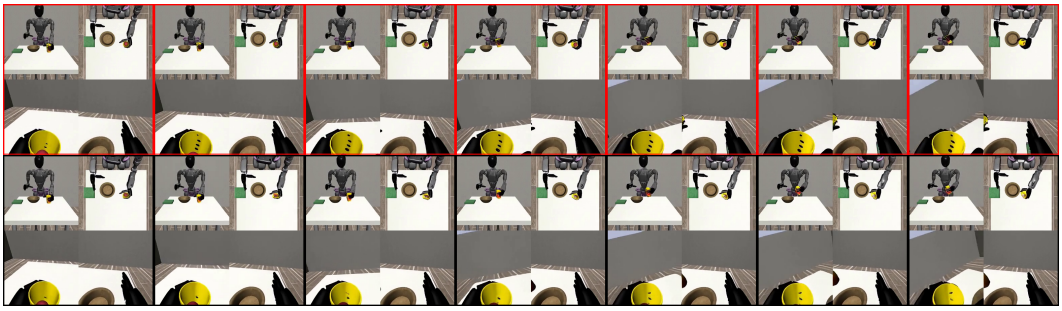


(a) Qualitative results for knocking the mug over backwards.

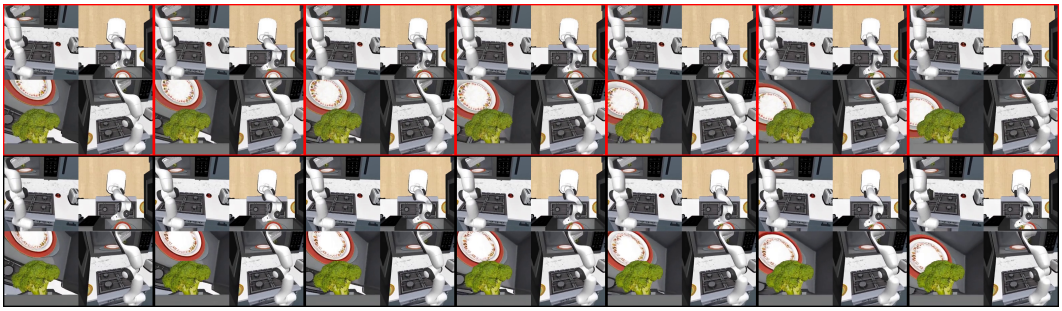


(b) Qualitative results for knocking the mug over to the right.

Figure 5: Qualitative results for OOD actions that knocks the mug over, with the first row being ground truth frames with red borders, and the second row being generated imagination frames. The videos are downsampled 5x from 40 frames (2 seconds) to 8 frames.



(a) Qualitative results for dexterous humanoid in DexMimicGen dataset.



(b) Qualitative results for the Panda Omron robot in the Robocasa dataset.

Figure 6: Qualitative results cross-embodiment modeling, with the first row being ground truth frames with red borders, and the second row being generated imagination frames. The videos are downsampled 3x from 20 frames (1 second) to 7 frames.

D.2 World Action Planner

Here we provide the details of the world action planner experiments in Section 5.2.

Baseline Setup. For the generalist policy baselines $\pi_{0.5}$ and Cosmos-Policy, we follow their original recommended training setups and ensure both training convergence and strong in-distribution performance. Additional details are provided in the corresponding experimental sections below.

For the policy enhancement baselines, we adapt each method as closely as possible to its original setup while using the same world model and VLM-based evaluation where applicable. For SAILOR (Jain et al., 2025), because we do not have task-specific trained reward models, we use the same VLM as the reward model. For GPC-RANK (Qi et al., 2025), we follow the ranking setup from the original work and use the same VLM reward function. For DREAMSTEER (Cui et al., 2026), we sample action proposals from the $\pi_{0.5}$ VLA, incorporate the additional action primitives used in the original method, and use the same VLAC critic model as in the original implementation.

For the vision-language planner baseline, we use the same agent, action proposal prompts and triangulation procedure, optional diffusion policy, and robot controller as in WAP. The proposed actions are executed directly, without world-model imagination, global action optimization, or local action search. To ensure a fair comparison, we explicitly include common failure modes observed during VLM exploration in the prompt, providing the VLM planner with additional guidance to avoid them. Therefore, the performance gains of our method over the VLM planner baseline arise from the world-model imagination and action-search process, isolating the benefit of explicit visual world modeling and action optimization. Notably, even when the VLM agent is explicitly instructed to avoid collisions and maintain sufficient clearance, such failures still occur and are only mitigated after reasoning over imagined world-model rollouts and refining the proposed actions, further demonstrating the effectiveness of our approach.

Resource Comparison. Here we summarize the data resources used by finetuning each component in the experiments below. All expert demonstrations are drawn from the same dataset, though the quantities differ across methods. The expert demonstrations used by the world model and for perturbation are the same as the one received by the policy in WAP, thus the unique number of expert demonstrations are 30 for the compositional task setting and 5 for the new layout setting.

Among the policy enhancement baselines, SAILOR (Jain et al., 2025) and GPC-RANK (Qi et al., 2025) use a diffusion policy finetuned on the same data as the generalist policy baselines, and DREAMSTEER (Cui et al., 2026) uses the $\pi_{0.5}$ VLA from those baselines.

Table 7: **Data resources consumed by each component.** A dash indicates that the component is not used in that setting. Perturbed trajectories are demonstration replays with action noise, as described in Appendix D.1.1.

Component	Compositional task	New layout	Novel task
Policy in WAP	30 sub-task demonstrations	5 unmodified-layout demonstrations	–
World model in WAP	The 30 demos above plus 70 perturbed	The 5 above plus 10 perturbed	50 exploratory trajectories in simulation; 10 failure trajectories on the real robot
Generalist policy baselines ($\pi_{0.5}$, cosmos-policy)	40–50 sub-task demonstrations	40–50 unmodified-layout demonstrations	0 target trajectories (zero-shot); 50 exploratory trajectories (same-data)

D.2.1 COMPOSITIONAL TASK GENERALIZATION

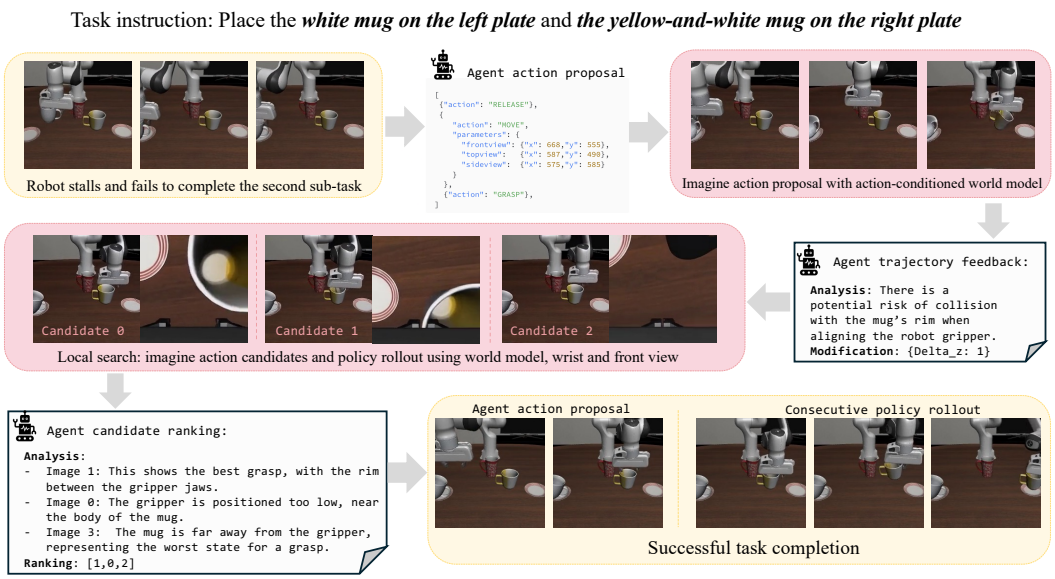


Figure 7: **Illustration of world action planner in compositional task generalization**

Environment Setup. We select the five tasks from LIBERO-Long, each composed of two tasks from the LIBERO-90 suite, involving pick and place operations as well as articulated object manipulation. Table 8 lists the selected tasks and their respective subtasks. We select all the tasks in LIBERO-Long that has direct corresponding subtasks in LIBERO-90 with the same scene, excluding ones that do not have direct counterparts.

Table 8: **Composition of LIBERO-Long tasks from LIBERO-90 subtasks.** Both subtasks in each row share the scene specified in the first column. Instructions are shortened for readability; task membership follows the official LIBERO task map (Liu et al., 2023).

LIBERO-Long composed task	LIBERO-90 subtask 1	LIBERO-90 subtask 2
Living room scene 2 Put alphabet soup and tomato sauce in the basket.	Put alphabet soup in the basket.	Put tomato sauce in the basket.
Living room scene 5 Put the white mug on the left plate and the yellow and white mug on the right plate.	Put the white mug on the left plate.	Put the yellow and white mug on the right plate.
Living room scene 6 Put the white mug on the plate and chocolate pudding to the right of the plate.	Put the white mug on the plate.	Put chocolate pudding to the right of the plate.
Living room scene 1 Put alphabet soup and the cream cheese box in the basket.	Put alphabet soup in the basket.	Put the cream cheese box in the basket.
Kitchen scene 4 Put the black bowl in the bottom drawer of the cabinet and close it.	Put the black bowl in the bottom drawer of the cabinet.	Close the bottom drawer of the cabinet.

World Action Planner Setup. We train our diffusion policy, the $\pi_{0.5}$ baseline, and the cosmos-policy using an action chunk length of 40 (equivalent to a 2-second horizon) following Kim et al. (2026), maintaining a 10-step execution horizon for each chunk before re-planning the subsequent sequence. Following Chi et al. (2025), we implement our diffusion policy using a transformer architecture, with

Table 9: **Diffusion policy architecture and training hyper-parameters.**

Hyper-parameter	Value
Layers	12
Heads	8
Dimension	512
Observation length	1
Prediction length	40
Execution action steps	10
Input images	wrist view and agent view
Image resolution	128×128
Crop size	116×116
Batch size	64

the specific hyper-parameters detailed in Table 9. The policy is trained on the LIBERO-90 dataset using 30 expert demonstrations per task, using the slightly less data as the VLA and WAM baselines. The world model is trained for 10k steps with a global batch size of 64, using 100 trajectories per task from the LIBERO-90 suite exclusively, as per the protocol in Section 5.1.

Baseline Details. The baselines follow the same configuration as our diffusion policy, utilizing an action chunk length of 40 and a 10-step execution horizon. These models are trained on the LIBERO-90 dataset, with 40 to 50 demonstrations per task after filtering out unsuccessful demonstrations and no-op actions following Kim et al. (2024). Crucially, we also included our VLM agent to decompose the subtasks and set the prompt to the baseline policies to be the prompt of the current sub-task, and the VLM switching to the prompt of the second sub-task when finishing the first one, to ensure the prompt input are completely in-distribution.

We train the $\pi_{0.5}$ baseline for 40k steps using the official LIBERO JAX implementation, achieving an average in-domain success rate of 95.8% on the LIBERO-90 tasks. Evaluation on the LIBERO-Long suite indicates that while $\pi_{0.5}$ achieves high success rates on non-compositional tasks which only involves environmental variations, it struggles with compositional generalization, particularly in scenarios requiring active navigation toward secondary targets. Detailed performance metrics are summarized in Table 10. Success is highest when a test task closely mirrors the training distribution.

Table 10: **Performance of $\pi_{0.5}$ trained on LIBERO-90 and evaluated on LIBERO-Long.** Success is highest on the task with a near-duplicate counterpart in the training suite; the Kitchen Scene 4 task also has a nontrivial success rate.

Task	Success rate (%)
LIVING ROOM SCENE2 put both the alphabet soup and the tomato sauce in the basket	4
LIVING ROOM SCENE2 put both the cream cheese box and the butter in the basket	0
KITCHEN SCENE3 turn on the stove and put the moka pot on it	6
KITCHEN SCENE4 put the black bowl in the bottom drawer of the cabinet and close it	44
LIVING ROOM SCENE5 put the white mug on the left plate and put the yellow and white mug on the right plate	0
STUDY SCENE1 pick up the book and place it in the back compartment of the caddy	90
LIVING ROOM SCENE6 put the white mug on the plate and put the chocolate pudding to the right of the plate	0
LIVING ROOM SCENE1 put both the alphabet soup and the cream cheese box in the basket	0
KITCHEN SCENE8 put both moka pots on the stove	0
KITCHEN SCENE6 put the yellow and white mug in the microwave and close it	0

For instance, the task "STUDY SCENE1: pick up the book and place it in the back compartment of the caddy" in LIBERO-Long has a direct counterpart in LIBERO-90 ("STUDY SCENE2"), with only minor environmental variations, and reaches 90%. The Kitchen Scene 4 task reaches 44%. This comparatively strong result may reflect the short transition between its two subtasks: after placing the bowl, the robot need only close the nearby drawer. By contrast, $\pi_{0.5}$ scores at most 4% on the other four selected compositional tasks in Table 11. In these pick-and-place compositions, the gripper commonly stalls after completing the first subtask and rarely moves toward the second

target object. These results reinforce our observation that imitation-learning policies rely heavily on memorized motion patterns and lack the test-time reasoning necessary to synthesize novel trajectories for adaptation. For the cosmos-policy baseline, constrained by computational resources, we trained the model following the official setup for 30k steps, achieving an action L1 loss of 0.025. This configuration yields an average in-domain success rate of 93% on LIBERO-90 tasks. Among the five selected compositional tasks, cosmos-policy scores 12% on the Kitchen Scene 4 task and 0% on the other four.

Policy-enhancement methods such as SAILOR, GPC-RANK, and DREAMSTEER (Jain et al., 2025; Qi et al., 2025; Cui et al., 2026)—which refine policy-generated actions rather than our VLM agent proposals—remain weak on average, reaching 16.4%, 12.0%, and 20.0%, respectively. Kitchen Scene 4 is an exception, with success rates of 54%, 50%, and 60%; on the other four selected compositions, SAILOR reaches at most 18%, GPC-RANK at most 10%, and DREAMSTEER at most 26%. In those pick-and-place compositions, the base policy often produces near-zero-magnitude actions after finishing the first subtask. Gaussian perturbations in SAILOR's MPPI exploration (Jain et al., 2025) can move the gripper toward the next object, but rarely yield successful task completion. These results highlight the importance of agent intervention in our system.

Table 11: **Per-task success rates (%) for compositional task generalization.** These results underlie the compositional task generalization averages in Table 3.

Compositional task	$\pi_{0.5}$	cosmos	SAILOR	GPC-RANK	DREAM STEER	VLM planner	**WAP**
PnP alphabet soup & tomato sauce	4	0	18	10	26	56	**72**
PnP white mug & yellow and white mug	0	0	0	0	0	28	**68**
PnP white mug & chocolate pudding	0	0	8	0	14	46	**78**
PnP alphabet soup & cream cheese box	0	0	2	0	0	32	**70**
Put black bowl in bottom drawer & close drawer	44	12	54	50	60	48	**72**

D.2.2 New Layout Generalization

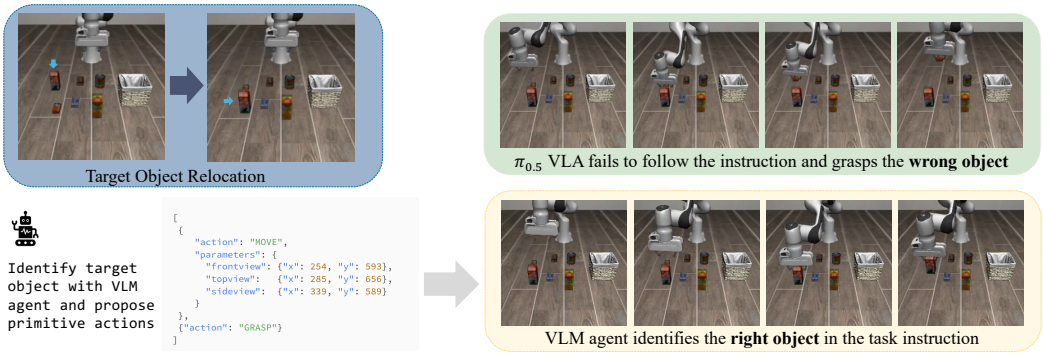


Figure 8: **Illustration of new layout generalization**

Environment Setup. We provide the images of the original layout and our modified layout below in Fig. 9.

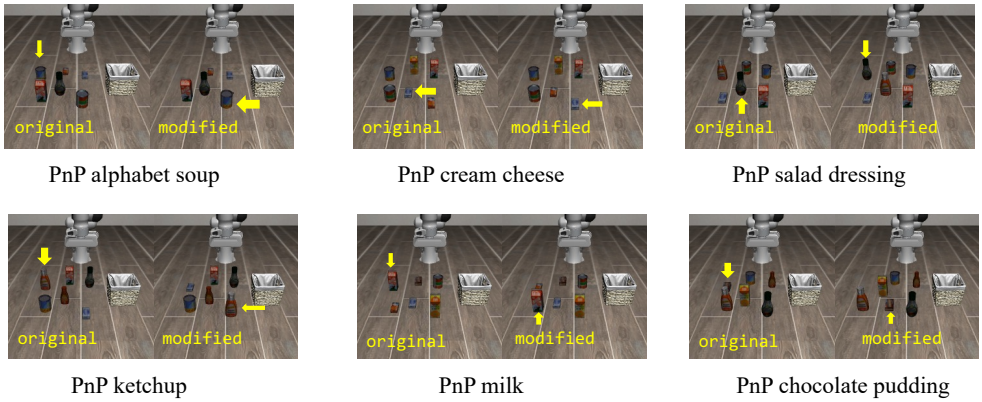


Figure 9: Original object layout and our modified object layout in LIBERO-Object. Yellow arrow points to the target object to be grasped into the basket.

World Action Planner Setup. For our method, we fine-tune the diffusion policy and world model (pre-trained on LIBERO-90 as in the previous compositional task generalization experiments) using the official LIBERO-Object dataset under its original layout without the position modification introduced at test time. The diffusion policy is fine-tuned with only 5 demonstrations per task. To enhance the robustness of the world model, we augment the fine-tuning set with 10 additional trajectories generated by adding Gaussian noise (with std of 0.08 and 0.16) to the expert actions.

Baseline Setup. We directly evaluate the official LIBERO checkpoints for the $\pi_{0.5}$ and cosmos-policy baselines. We observe that $\pi_{0.5}$ achieves non-zero success rates on the "PnP Chocolate Pudding" task because the target object is not sufficiently displaced from its original training location; this allows the policy to occasionally capture the pudding during subsequent re-grasp attempts, as shown in Fig. 10a. In most other cases, however, the policy fails by erroneously targeting distractor objects near the original coordinates, as illustrated in Fig 10b. For other tasks where the target object is displaced far away, the $\pi_{0.5}$ model exhibits zero success rates, with similar patterns of grasping distractor objects placed near the original coordinates.

The cosmos-policy baseline performs worse than $\pi_{0.5}$, yielding a zero success rate across all tasks. Notably, the WAM consistently repeats grasp attempts at the original training coordinates, failing to exhibit the local exploration observed in $\pi_{0.5}$. Qualitative results of the environment rollout and the predicted future states from the WAM are illustrated in Fig. 11. While the predicted images are visually realistic, they do not facilitate successful task completion. These results suggest that because the model is trained primarily to mimic demonstration trajectories and predict action outcomes



(a) Low probability success example of $\pi_{0.5}$ on the modified layout of PnP chocolate pudding.



(b) Failure example of $\pi_{0.5}$ on the modified layout of PnP chocolate pudding.

Figure 10: Qualitative evaluation of $\pi_{0.5}$ on the modified layout of PnP chocolate pudding, down-sampled from 20 FPS to 1 FPS for clarity. The VLA exhibits low success rates, as the gripper rarely drifts toward the correct target and instead frequently grasps distractor objects placed at the original position during re-grasp attempts.

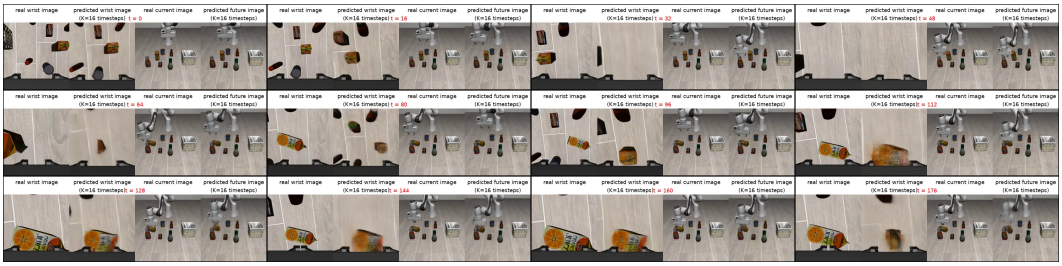


Figure 11: Qualitative evaluation of the cosmos-policy on the modified layout of the PnP chocolate pudding task, illustrating ground-truth environment rollouts alongside predicted future states from the WAM. Despite the modified layout, the model continues to target the original target object coordinates, resulting in repetitive and unsuccessful re-grasp cycles at the original position of the target object.

based on pre-trained video generation, it lacks the necessary reasoning and planning capabilities to generalize to novel tasks or environmental layouts.

For policy enhancement methods such as SAILOR, GPC-RANK, and DREAMSTEER (Jain et al., 2025; Qi et al., 2025; Cui et al., 2026), average success rates remain low at 3.7%, 2.7%, and 10.7%, respectively, because the actions generated by the policy often target incorrect positions. These results underscore the importance of independent reasoning and planning when encountering novel scenarios, rather than relying exclusively on the demonstrations within the imitation learning dataset.

Table 12: **Per-task success rates (%) for new-layout generalization.** These results underlie the new-layout averages in Table 3.

New-layout task	$\pi_{0.5}$	cosmos	SAILOR	GPC-RANK	DREAM STEER	VLM planner	**WAP**
PnP alphabet soup	0	0	0	0	8	30	**88**
PnP cream cheese	0	0	0	0	12	50	**86**
PnP salad dressing	0	0	0	0	14	32	**90**
PnP ketchup	0	0	0	0	0	16	**66**
PnP milk	0	0	0	0	20	34	**84**
PnP chocolate pudding	10	0	22	16	10	64	**78**

D.2.3 NOVEL TASK GENERALIZATION

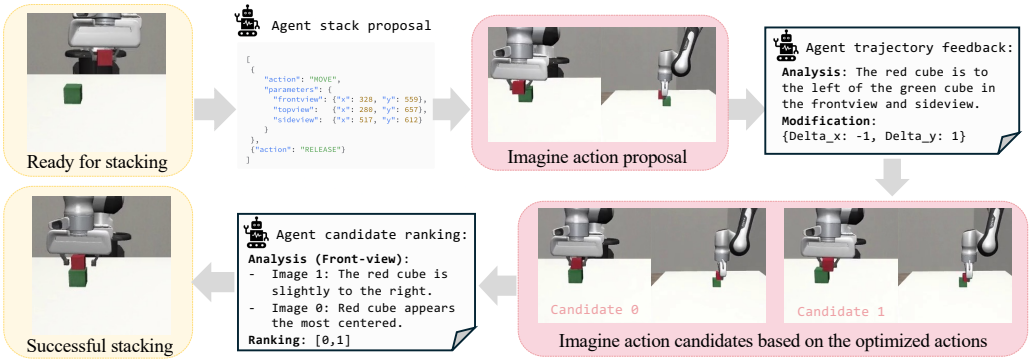


Figure 12: **Illustration of world action planner with novel task planning in `StackCube`.**

Environment Setup. We evaluate on the `PickPlaceCan` and `StackCube` tasks of Robosuite (Zhu et al., 2020), with 50 trials per task. Unlike the previous two settings, no task-specific policy and no expert demonstration are available here: the World Action Planner proposes and refines target end-effector poses, and the gripper is actuated by the two hard-coded primitives described below. This setting therefore isolates the planner itself, since the only learned components in the loop are the VLM agent and the action-conditioned world model.

Action Primitives. Actions are 7-dimensional end-effector operational space velocity commands $[\Delta x, \Delta y, \Delta z, \Delta\psi_r, \Delta\psi_p, \Delta\psi_y, g]$, where $g = 1$ closes and $g = -1$ opens the gripper, and each segment below is held for 10 control steps:

- `GRASP`: descend with the gripper open, close the gripper in place, then lift with the gripper closed,
  $[0, 0, -0.5, 0, 0, 0, -1] \times 10 + [0, 0, 0, 0, 0, 0, 1] \times 10 + [0, 0, 1, 0, 0, 0, 1] \times 10$.
- `RELEASE`: open the gripper in place, then retract upward,
  $[0, 0, 0, 0, 0, 0, -1] \times 10 + [0, 0, 1, 0, 0, 0, -1] \times 10$.

We point out that these action primitives are not policies and they require no knowledge or prior about the specific task and environment configuration, while requiring only basic knowledge about the robot: they are expressed purely in the gripper frame and reference no object pose, target location, or scene layout. The same two primitives are reused unchanged across both tasks, and across the real-robot experiments in Appendix D.2.4. All object-dependent decisions—where to grasp, at what orientation, and where to place—are made by the planner rather than encoded in the primitives.

World Action Planner Setup. We initialize the world model checkpoint from the LIBERO-90 world model of the compositional task generalization experiments (Appendix D.2.1). At test time, we collect 50 trajectories through exploration using the VLM agent without any expert demonstrations or robot policies, training the world model on the data for 10 epochs. We point out that although some of these exploratory trajectories happen to be successful, we use them solely as video data for world model test-time training and never as demonstrations for a policy; prior methods (Jain et al., 2025; Qi et al., 2025) instead require expert demonstrations in addition to the in-distribution sampled trajectories in order to succeed. Search scales follow Appendix C.2: a global modification scale of 0.06, and a local grid scale δ of 0.02 for `StackCube` and 0.05 for `PickPlaceCan`, following the rule of setting the scale to roughly half the size of the manipulated object. The tighter scale on `StackCube` reflects the smaller cube and the tighter placement tolerance of stacking, which is also where the gap to the vision-language planner is largest: without imagined rollouts to verify the placement pose, the VLM planner reaches only 22% on `StackCube` against 58% on `PickPlaceCan`, while WAP reaches 76% and 80% respectively (Table 13).

Baseline Setup. The policy-based baselines depend on a learned base policy, whereas WAP uses no learned task policy in this setting. We evaluate them under two regimes: *zero-shot*, using the pretrained base policies without target-environment finetuning; and *same-data adaptation*, finetuning the corresponding base policies on the same 50 VLM-collected exploratory trajectories used to train WAP's world model. These trajectories are diverse exploration data rather than expert demonstrations.

The base policies remain weak after same-data adaptation, and the policy-enhancement methods improve them only modestly because their candidates are still sampled from the base-policy distribution. Table 13 reports both regimes.

Table 13: **Per-task success rates (%) under two policy-training regimes.** For policy-based baselines, "zero-shot" denotes no target-environment policy finetuning, while "same data" denotes finetuning on the same 50 exploratory trajectories used to train WAP's world model. The vision-language planner and WAP use no learned task policy, so each has one result per task spanning both regimes. Each result is over 50 trials.

Method	PickPlaceCan		StackCube	
	Zero-shot	Same data	Zero-shot	Same data
$\pi_{0.5}$	0	16	0	4
cosmos-policy	0	4	0	0
SAILOR	0	8	0	4
GPC-RANK	0	10	0	6
DREAMSTEER	8	24	0	12
Vision-language planner	58		22	
World action planner	**80**		**76**	

D.2.4 Real Robot Novel Task Experiments

Here we provide additional details for the real robot experiments.

Hardware and Task Setup. We evaluate on a 7-DoF Franka Panda arm with a parallel-jaw gripper, observed by two exterior cameras as shown in Fig. 4; the world model imagines both views jointly as two panels of 256×256 each. The task is to pick a can from the table and place it inside a white basket. We adopt the same protocol as the novel task generalization setting of Appendix D.2.3: no task-specific policy and no expert demonstration are available, gripper closing and opening are issued as hard-coded primitives, and the planner is responsible only for the target end-effector poses, which are executed by the controller shipped with the Franka Panda without any customized or learned low-level controller. The entire real-robot pipeline therefore consists of only the VLM agent, the world model, and the shipped controller. We randomize the position and orientation of both the can and the basket for every rollout.

Failure Exploration Data Collection. We first collect 10 trajectories on the `PnP green can` task using the VLM proposal alone. All 10 exploration trajectories fail: the VLM either collides with the rim or drop the can outside the basket. (Fig. 13). This failure mode is a direct consequence of the VLM reasoning about the scene semantically rather than physically. We stress that these trajectories contain *no successful demonstration*; unlike prior world-model policy enhancement methods (Jain et al., 2025; Qi et al., 2025), we never provide the system with an expert trajectory for the real robot.

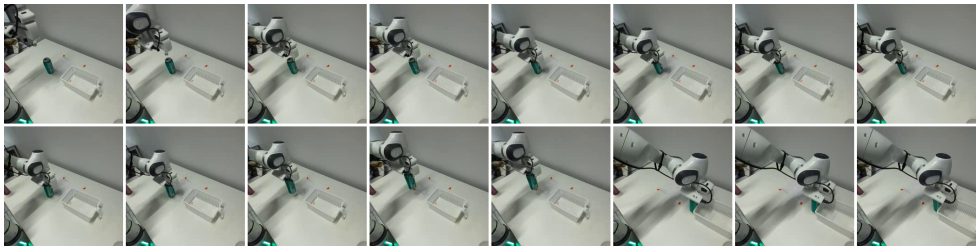


Figure 13: **Representative failed VLM trajectory on the real robot.** Frames are ordered left to right, top to bottom. The VLM grasps the can correctly but commands a placement pose that drives the can into the basket rim.

World Model Training at Test Time. Starting from the same LIBERO-90 world model of the compositional task generalization experiments (Appendix D.2.1), we train the world model on these 10 failed exploration trajectories for 100 steps at test time deployment. Fig. 14 shows imagined rollouts after finetuning for both the green can and the *previously unseen* purple can: in both cases the world model reproduces the contact between the can and the basket rim, indicating that the collision

is predicted from the geometry of the commanded motion rather than memorized from the green-can trajectories. This transfer is what makes the planner usable on a new object without collecting any new data.



(a) Placing the green can, seen during world model finetuning.



(b) Placing the purple can, never seen during world model finetuning.

Figure 14: **Qualitative examples of real-robot world-model imagination.** Each panel shows the two imagined exterior camera views, one per row. The imagined rollout exposes the collision between the can and the basket rim before any action is executed.

Planning and Results. Given the imagined rollout, the VLM agent reports the predicted collision and global action optimization raises the placement pose for the can to clear the rim, after which the trajectory is executed and the can is released inside the basket (Fig. 15). Per-task success rates over 20 trials per task are reported in Table 14. The vision-language planner reaches only 10% on both cans, reproducing the rim collision it cannot foresee, whereas WAP reaches 70% on the green can and 60% on the purple can. The imitation-learning and policy-enhancement baselines score 0% on most cases because they require finetuning on successful demonstrations, while the exploratory trajectories collected are all failures. The exception is DREAMSTEER where one trial succeeded due to sampling from the primitive library (Cui et al., 2026). Of the 14 failed trials across both tasks, six leave insufficient clearance over the basket rim, four lose the can when the gripper slips during grasping, and four release the can slightly outside the basket.

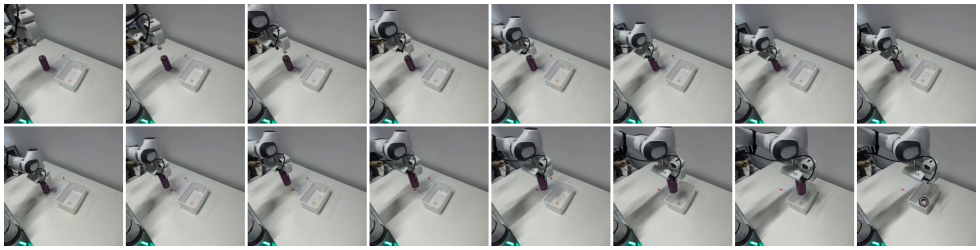
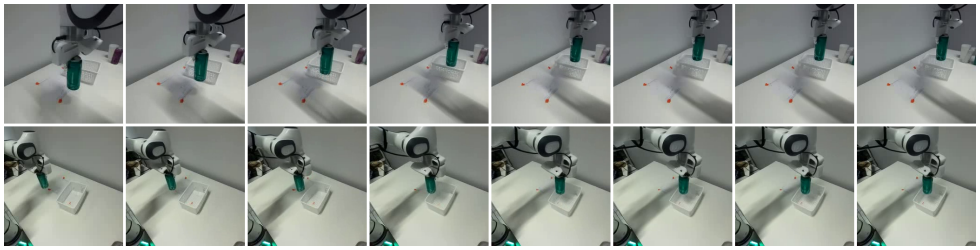


Figure 15: **Successful `PnP purple can` rollout with the World Action Planner.** Frames are ordered left to right, top to bottom. The optimized placement pose clears the basket rim, and the can is released inside the basket.

Successful Imagination of the World Model. Here we demonstrate that our world model can imagine the successful trajectories after placement, which it has never seen during training. As shown in Fig. 16, the world model accurately imagined the successful clearance during placement, an action trajctory it has never seen during finetuning. Such results shows that the world model successfully generalizes to the real world scene using only 10 failure trajectories, despite being trained on only limited simulation data, demonstrating the superior generalization capabilities of our world model.

Table 14: **Per-task success rates (%) under two policy-training regimes.** For policy-based baselines, "zero-shot" denotes no target-environment policy finetuning, while "same data" denotes finetuning on the same 10 exploratory trajectories used to train WAP's world model. The vision-language planner and WAP use no learned task policy, so each has one result per task spanning both regimes. Each result is over 20 trials.

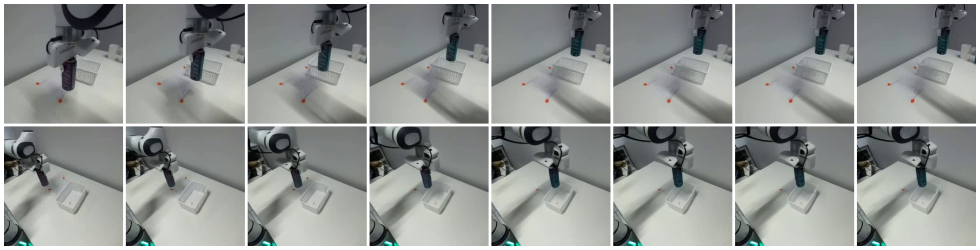
Method	PnP green can		PnP purple can	
	Zero-shot	Same data	Zero-shot	Same data
$\pi_{0.5}$	0	0	0	0
cosmos-policy	0	0	0	0
SAILOR	0	0	0	0
GPC-RANK	0	0	0	0
DREAMSTEER	0	10	0	0
Vision-language planner	10		10	
World action planner	**70**		**60**	



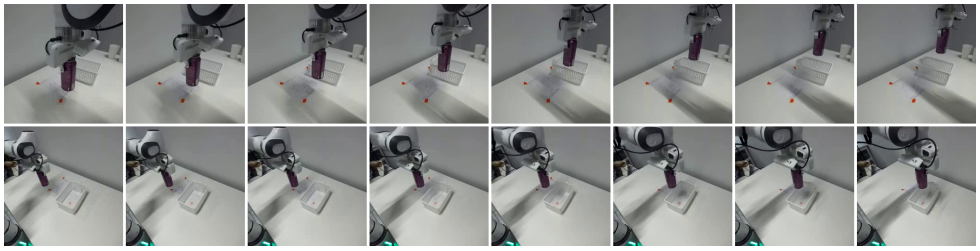
(a) World model imagination of successfully placing the green can.



(b) Real Execution of successfully placing the green can.



(c) World model imagination of successfully placing the purple can.



(d) Real Execution of successfully placing the purple can.

Figure 16: **Qualitative examples of real-robot world-model imagination versus ground truth for successful placement actions.** Each panel shows the two exterior camera views, one per row. The imagined and real execution corresponds to the same actions and states. The imagined rollout versus ground truth demonstrates that the world model can accurately imagine the successful lifted actions that it has never seen during training.

D.3 Planning Sample Efficiency

For the baseline methods, we use the same initial action proposal and rollout horizon as WAP to ensure a fair comparison. We provide the baselines with ground-truth simulator rollouts to ensure a stronger baseline, while WAP instead relies on trajectories imagined by the world model.

We adopt the same MPPI-style update rules, and search the hyper-parameters starting from their original hyper-parameter choices, and ensure using the same number of VLM calls, so wall-time is determined by the number of samples. We also provide the results with the original curated reward design instead of our VLM agent in Table 15.

Table 15: **Planning sample efficiency against MPC.** Success rates (%) are averaged within each setting. VLMPC uses ground-truth simulator rollouts, whereas WAP uses world-model imagination.

Method	Compositional task	New layout	Novel task
VLMPC original reward (8 samples)	34.0	36.3	64.0
VLMPC original reward (16 samples)	48.0	44.0	72.0
VLMPC agent reward (8 samples)	36.0	38.3	64.0
VLMPC agent reward (16 samples)	46.0	50.0	76.0
WAP w/ global optimization only (1 sample)	58.0	66.0	65.0
WAP (6 samples)	**72.0**	**82.0**	**78.0**

We find that MPC methods based on Gaussian sampling often waste computation on implausible candidates produced by an undirected proposal distribution. In contrast, our global optimization is guided by VLM feedback and can mitigate potential risks from a single world-model-imagined trajectory (Fig. 17), whereas MPPI typically requires multiple sampling iterations to refine its action distribution. In addition, our local grid search evaluates physically grounded candidates at neighboring poses, further reducing the number of implausible samples that can arise from the Gaussian sampling used in CEM or MPPI (Ning et al., 2025).

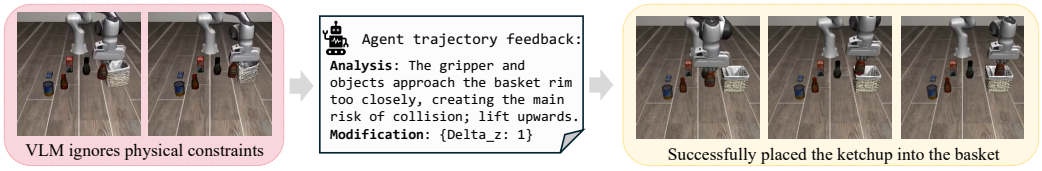


Figure 17: **Global Action Optimization.** While VLMs often ignore physical constraints, our framework leverages world model imaginations to identify potential collisions and optimize action sequences for safe execution.

D.4 Ablation Experiments

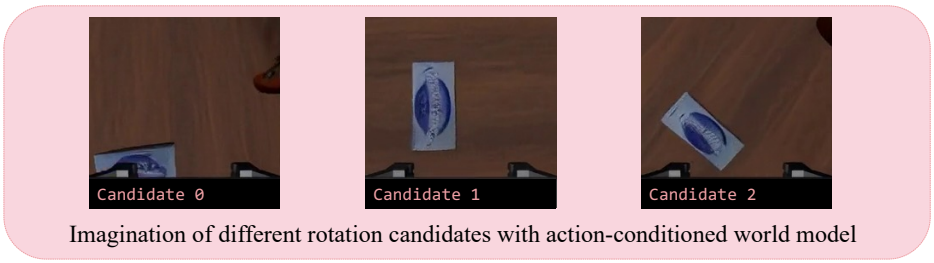
Here we provide the results for the ablation experiments.

D.4.1 Ablation of Planning Stages

We conduct a stage-wise ablation study to evaluate the contributions of each module within the World Action Planner. The results, presented in Table 16, demonstrate that each stage incrementally improves the designed pipeline. For easy tasks, object clearance via global action optimization is sufficient. Notably, for the PnP alphabet soup and cream cheese task, when grasping the cream cheese box after placing the alphabet soup, the gripper is rotated, requiring the agent to find the right orientation for the grasp. While the VLM struggles to describe the required rotation analytically, it can effectively identify the optimal orientation when evaluating imaginations of candidate poses, as illustrated in Fig. 18. For the fine-grained PnP white mug and yellow-and-white mug task, transitioning from the first mug to the next presents a specific challenge: it is often unclear which state serves as a valid "in-distribution" starting point for the policy to successfully trigger a grasp and complete the second sub-task. To resolve this ambiguity, we imagine consecutive policy rollouts following various action candidates to determine which state allows the policy to effectively continue

Table 16: **Results for ablations of our action planning pipeline.** We ablate different components in our world action planner using the LIBERO-Long tasks, where each component contributes to higher success rates.

Task	VLM planner	+ Global	+ Local	+ Policy rollout
PnP alphabet soup & tomato sauce	56	64	64	**72**
PnP white mug & yellow and white mug	28	40	54	**68**
PnP white mug & chocolate pudding	46	70	72	**78**
PnP alphabet soup & cream cheese box	32	48	66	**70**



Agent Ranking for Rotation Search

```
1. Analyze the object: The target object is the cream cheese box, which is a small, blue,
   rectangular box. To achieve a firm and clear grasp, the gripper should be horizontal and
   aligned to grasp the long edges of the box.
2. Evaluate Candidate 0: In this image, the gripper is oriented vertically. This would result
   in grasping the short edges of the box, which is less stable.
3. Evaluate Candidate 1: In this image, the gripper is oriented horizontally. The opening of
   the gripper fingers is parallel to the long edges of the box, which is the recommended
   orientation for a stable grasp.
4. Evaluate Candidate 2: In this image, the gripper is tilted at an angle. This orientation is
   not horizontal and would not result in a clean grasp along the long edges of the box.
{best_candidate_id: 1}
```

Figure 18: Illustration of local search for gripper rotation. The VLM successfully identifies the right rotation from imagination of candidates, while failing to describe the correct rotation analytically.

the task. As illustrated in Fig. 19, we can identify the optimal state for completing the grasp by imagining the consecutive policy rollout (shown in row 2), whereas selecting the correct candidate remains difficult without the benefit of further imagination (as in row 1).

D.4.2 Ablation of VLM Agent

Here we provide ablation of the choice of VLM agent from the default Gemini 3.0 Flash against an additional GPT-5.4 in the new-layout generalization setup. We report the Vision-language-planner baseline for each agent that directly executes action proposal without search along our world action planner results. As shown in Table 17, the results remain consistent across VLM agent choices.

Table 17: **Ablation of VLM agent.** Per-task success rates (%) across VLM agent choices.

VLM Agent	PnP soup	PnP cheese	PnP salad dressing	PnP ketchup	PnP milk	PnP pudding
GPT-5.4 baseline	26	52	30	14	38	68
GPT-5.4 with WAP	84	82	88	64	80	76
Gemini-3-flash baseline	30	50	32	16	34	64
Gemini-3-flash with WAP	88	86	90	66	84	78

D.4.3 Robustness Analysis

Here we evaluate the hyperparameter robustness of our method, especially sensitivity to local search scale and global search scale. As described in Appendix C, we chose the search scale to be

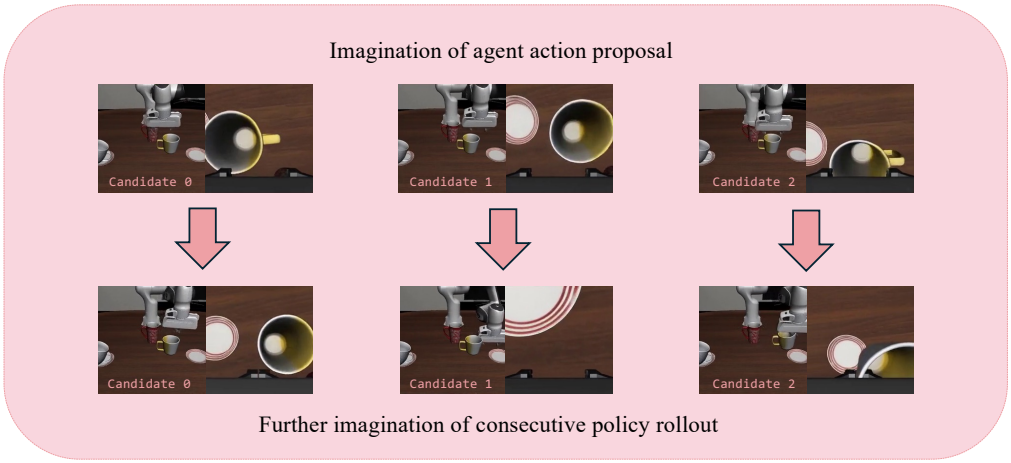


Figure 19: Illustration of imagining consecutive policy rollout. While in the first row, it is hard to identify which one is the good state to continue the second sub-task by grasping the mug, we can easily see that for Candidate 2, the policy successfully grasped the mug and continues the second sub-task from the consecutive policy rollout imagination in the second row.

approximately half the object size or height in magnitude. We further provide ablation results on these parameters in Table 18, highlighting the robustness of our method.

Table 18: **Robustness to local and global search scales.** Per-task success rates (%) across search-scale choices. "Original" denotes the default setting used in our main experiments.

Local search scale ablation

Local search scale	PnP soup & tomato sauce	PnP white mug & yellow and white mug	PnP mug & pudding	PnP soup & cheese
0.04	66	66	74	72
0.05 (original)	72	68	78	70
0.07	70	72	74	66

Global search scale ablation

Global search scale	PnP soup	PnP cheese	PnP salad dressing	PnP ketchup	PnP milk	PnP pudding
0.04	78	80	86	62	80	76
0.06 (original)	88	86	90	66	84	78
0.08	86	82	84	70	90	80

E WALL CLOCK TIME ANALYSIS

Here we report the wall clock time of our world model. When training on a H100 GPU with batch size 2, one single training step takes approximately 4.8 seconds. When running inference on a A100 GPU with batch size 1, one single forward takes 0.85 seconds, so predicting 20 frames requires 20 denoising steps and takes around 17 seconds. As a result, the global optimization will take around 30 seconds including world model imagination and VLM reasoning, and the local search can take 2 to 3 minutes. For the real world experiments, a complete task completion will take approximately 2 minutes with global optimization. We note that the computation cost of local search can be significantly decreased by adopting an auto-regressive world model and using KV cache for the shared history frames among all the candidates, which we leave as future work.

We emphasize that while generating imagined rollouts with the world model is computationally intensive, the system does not require continuous re-planning at every control step. In our experiments, we only invoke the planning system at automatically with the VLM at sub-task starting points. Furthermore, the overall planning latency is significantly influenced by the VLM agent's reasoning time. This process is analogous to "System 2" reasoning in LLM agents, where a deliberate thinking phase and chain-of-thought reasoning is required to solve difficult problems (Jaech et al., 2024; Li et al., 2024).

You are a helpful assistant for controlling a robot arm in a simulated environment. You will give advice on how to complete the task based on the image inputs.
Task instruction. Here is the task instruction: `{task_instruction}`
Observation. Here is the starting state of the current episode `<images>` and here is the current state images `<images>`. The execution history so far is `<images>`.
Action Proposal. Now, please propose the next actions for the robot to complete the task. You should complete the task following the order in the instruction. The available atomic actions are:

- **MOVE** Here you move the gripper to a target position, and you should point it out in the multiview current state images. The position should be represented by x,y pixel coordinates normalized to 0-1000. To grasp or operate an object, move towards it. If you need to place an object, point to the target gripper position so the object can be placed following the instruction. Examples:

```
{"action": "MOVE",
"parameters": {"frontview": {"x": 500, "y": 300},
"topview": {"x": 450, "y": 350},
"sideview": {"x": 480, "y": 320}}}
```

- **ROTATION** Here you rotate the gripper, and you return the rotation in Euler angles `[delta_roll, delta_pitch, delta_yaw]` in degrees. Example:

```
    {"action": "ROTATE","parameters": {"delta_roll": 0,
    "delta_pitch": 15, "delta_yaw": 0}}
```

- **RELEASE** Here you release the object by opening the gripper.
  Example: `{"action": "RELEASE","parameters": {}}`
- **GRASP** Here you grasp the object by closing the gripper.
  Example: `{"action": "GRASP","parameters": {}}`

Finally, return a list of actions in json format. **Skills**

- When placing objects into baskets or target areas, ensure sufficient clearance to avoid collision.
- When grasping objects, ensure the gripper is aligned with the grasp region to enable successful grasp.
- When navigating the gripper, ensure sufficient clearance and avoid collision with objects.

Figure 20: Example prompt template for agent action proposal.

Optimize gripper position. Here is the gripper position of our next robot actions `<images>`, and I want you to look carefully and analyze the position of the gripper and the object, and optimize the gripper position following the guidelines below. Return the gripper position adjustments in x y z directions: The coordinate system and axis are defined as `{coordinate_system}`.

Guidelines.

- Understand what the gripper is trying to do, based on your task instruction understandings. `{task_description}`.
- Analyze the trajectory frame by frame, and inspect whether there may be potential collisions with any object along the trajectory. If there is any potential collision with rims and objects, adjust the gripper so that the trajectory is fully clear of any obstacles.
- When the gripper is trying to place the object, first verify whether the position is above the target region to be placed. If not, zoom in and adjust the posiiton of the gripper.
- If the gripper is about to grasp an object, the gripper should be aligned and directly above the object. Adjust the gripper if it is clearly misaligned with the object.

Format. Please return your optimization direction in json format, for example:

```
{"Delta_x": 0,"Delta_y": -1,"Delta_z": 1}
```

where x, y, z can only be -1, 0 or 1, with 0 being no movement in that direction, 1 being move towards positive direction and -1 being move towards negative direction. Return 0 in the corresponding axis if the gripper is mostly aligned and safe.

Figure 21: Example prompt template for agent feedback for action optimization.

I have a set of images `<images>` showing the robot actions trying to complete the task `{task_description}`. I want you to rank them from best to worst in terms of whether the action is accomplishing the task goal safely. Here are more detailed guidelines: **Guidelines**.

- If the gripper is trying to grasp an object, I want you to rank them from best to worst in terms of whether the grasp is firm and clear. The best image should be a clear and firm grasp at the right place.

  Reflect on the trajectory history and task instructions, and understand which object is the gripper trying to grasp.
- If the gripper orientation is different in the candidate images, you should identify the best orientation for a clear and stable grasp.

Reminder.

- You should rank all the images following the same standard. If none of them is perfect, you should rank them by which one is closest.
- When ranking the later images, refer and reflect the previous candidates to rank them faithfully. For example, if the first image is blurry and the second image is clear, then the second image should be ranked higher than the first image.

Return the ranking result in json format, for example: `[0, 2, 1]`, where the ids in the list are the image ids ranked from best to worst, with the first being the best. The range of the ids should be from 0 to N-1, where N is the total number of candidate images.

Figure 22: Example prompt template for agent candidate ranking for local search.