

From Rollout to Reset: A Graph-Based Harness for Autonomous Long-Horizon Manipulation Evaluation

Jing Jiang^{1,*}, Yue Yang^{2,*}, Xinkai Jiang¹, Gedas Bertasius², Daniel J. Szafrir², Rudolf Lioutikov^{1,†}

Abstract— Robot manipulation policies are improving quickly, and real-robot evaluation remains the standard evidence for that progress. It still relies on a human to reset the scene between rollouts, which consumes operator time and leaves the initial state distribution unspecified, so results reproduce poorly. A recent system, AutoEval, automates both reset and scoring, but only for single-step tasks, because a long-horizon rollout can terminate in combinatorially many configurations that no single learned reset policy covers. We present HALTER, a Harness for Autonomous Long-horizon Task Evaluation and Reset, which restores the scene by planning over a library of learned atomic reset skills, so demonstration cost scales with the size of that library rather than with the number of terminal states. HALTER builds a spatial scene graph online from point clouds and vision foundation models, and an LLM reasons over this graph to score the rollout, plan the reset, and verify that the reset succeeded, without collecting labeled success images for any task. On four long-horizon tasks on a Franka arm, HALTER restores the scene in 76% of episodes, against 52% for AutoEval and 65% for a motion-planning reset, and it estimates the completed-skill fraction correctly in 90% of episodes, against 76%. Its reset-verification verdict is correct in 91% of episodes, compared with 78% for AutoEval. It also cuts the operator time of an evaluation campaign by 72% relative to manual reset. We further measure compositional generalization on three held-out tasks, where HALTER resets 74.7% of episodes against 1.3% for a per-task reset policy, and we ablate the scene representation and the graph update rate. Project website: [link](#).

I. INTRODUCTION

Robot learning algorithms have advanced rapidly in recent years, and manipulation policies now handle an increasingly broad range of tasks [1]–[4]. As the number of such policies grows, fair and efficient evaluation becomes the foundation on which the field measures progress. Simulation already supports this. A simulator restores the scene to a common initial distribution before every trial and executes many rollouts in parallel, which makes evaluation both reproducible and inexpensive [5]–[7]. Real-robot evaluation lags well behind. The operator restores the initial state according to individual judgment, so the initial state distribution remains unspecified and results reproduce poorly across trials and across laboratories [8], [9]. Restoring every object by hand also consumes substantial operator time, which bounds how many rollouts a campaign can afford (Fig. 1, left). Even recent real-robot benchmarks still depend on manual reset,

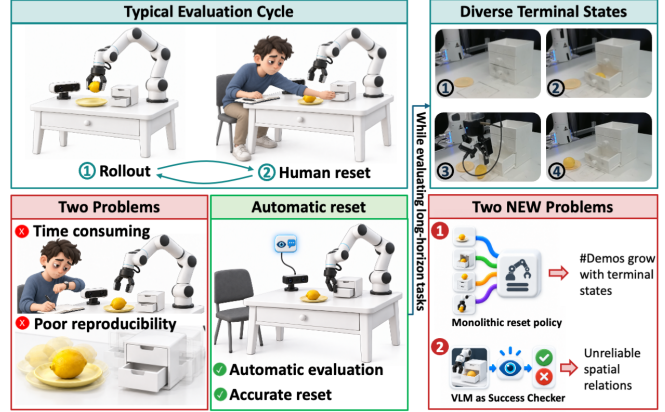


Fig. 1. Real-robot evaluation alternates a rollout with a reset (left). A human reset costs operator time and leaves the initial state unspecified, and an automated reset removes the human for single-step tasks. A long-horizon rollout, however, stops in many different terminal states (right): this four-step task fails at opening the drawer (1), at removing the lemon (2), in transit (3), and at closing the drawer (4). Covering them breaks both parts of an automated reset, because a monolithic reset policy needs demonstrations that grow with the number of terminal states, and a single-image success checker reads spatial relations unreliably.

and they identify it as a non-parallelizable bottleneck on evaluation scale [10], [11].

AutoEval [12] removes the human operator from this loop. It learns a task-specific reset policy and success classifier, which lets it evaluate policies continuously without supervision. This design assumes that a rollout ends in a narrow set of configurations, which holds for the single-step tasks it targets. Long-horizon tasks violate that assumption. A policy may fail at any skill, and a single task often admits many valid skill orderings, so the set of reachable terminal states grows combinatorially with the length of the task (Fig. 1, right). Learning one reset policy that covers all of them is impractical, because it would require a combinatorial number of demonstrations [13]. We therefore ask: *how can a system automatically reset the scene for long-horizon manipulation tasks in real-world robot experiments?*

We present HALTER, a system that evaluates long-horizon manipulation policies on real hardware with little human intervention. HALTER removes this blocker with a hierarchical reset, in which a high-level planner sequences low-level atomic skills to restore the scene. Demonstration cost then scales with the size of the skill library rather than with the number of terminal states, because the planner recombines a fixed set of skills instead of learning a separate policy for every state. Such a design depends on understanding the scene state, which HALTER needs in three places: scoring

*These authors contributed equally. †Corresponding author.

¹Karlsruhe Institute of Technology, Germany. {unisc@student.kit.edu, {xinkai.jiang, lioutikov}@kit.edu}

²University of North Carolina at Chapel Hill, USA. {yygx, gedas, daniel.szafrir}@cs.unc.edu

the rollout, planning the reset, and verifying that the reset succeeded.

Recent systems commonly ask a vision-language model to read the scene state directly from images [12], [14]. An image, however, leaves object geometry implicit in its pixels, and current VLMs read the fine spatial relations [15], [16] that all three steps require unreliably. HALTER therefore separates perception from reasoning. It recovers geometry from point clouds and vision foundation models [17], [18], assembles the result into a spatial graph over objects and their relations, and leaves the reasoning over that graph to an LLM, so each component operates where it is strong. HALTER builds this graph online as the rollout proceeds. The scene is only partially observable from any single view, so temporal information is essential. Online construction aggregates observations across the horizon, which lets the graph retain state that is observed at one moment and occluded later. On four long-horizon tasks on a Franka arm, HALTER resets the scene more reliably than both a per-task reset policy and a motion-planning reset, and it extends to held-out tasks without new demonstrations.

We make the following contributions:

- We present the first framework for automatic evaluation of long-horizon manipulation tasks on real robots, which restores the scene with a hierarchical planner over atomic reset skills.
- We propose a graph-based scene state estimator that pairs point clouds and vision foundation models with LLM reasoning, and requires no per-task trained classifier.
- We evaluate HALTER on a Franka arm across four long-horizon tasks, where it restores the scene in 76% of episodes against 52% for a per-task reset policy and 65% for a motion-planning reset. On three held-out tasks the planner recombines the same skill library into new reset sequences, so a held-out task costs no new demonstrations, and an ablation shows that the scene graph accounts for the gain.

II. RELATED WORK

A. Real-Robot Policy Evaluation

Early real-robot benchmarks focused on reproducibility. REPLAB [19] standardizes the hardware and fixes a protocol for scattering the objects, and SceneReplica [8] generates scenes in simulation and has an operator rebuild them on the table. FurnitureBench [20] carries the idea to long-horizon assembly. These benchmarks constrain the initial state, but a person still produces it before every rollout, and rigorous protocols still rely on human raters to judge the outcome [9].

A more recent line targets the operator time that these protocols require. RoboDojo [10] centralizes the hardware and opens it to remote submission, while RoboArena [21] distributes evaluation across institutions and raters. Both keep the human in the loop, so they raise throughput without lowering the cost of a single rollout. AutoEval [12] removes the operator instead, learning a reset policy and a

success classifier for each task, which its single-step setting makes tractable. Concurrent systems also reset real scenes without a human, for data collection and for policy self-improvement [22]–[24], though each prepares the reset for a task fixed in advance, by inverting the plan it just executed or by synthesizing a routine for that task. HALTER instead plans the reset at run time from the estimated terminal state of an external policy that may fail at any subgoal. The word harness also carries a different sense in recent work: Harness VLA [25] and Guava [26] wrap a manipulation policy in order to steer it, whereas HALTER wraps one in order to measure it.

B. Long-Horizon Manipulation

Long-horizon manipulation is commonly solved by composition. Task and motion planning sequences predefined skills under a symbolic planner [27], [28]. Learned variants replace those skills with policies: LEAGUE [29] lets a symbolic planner guide skill learning, GSC [30] chains skills with diffusion models, and NOD-TAMP [31] transfers them through neural object descriptors. Plan-Seq-Learn [32] and ManipGen [33] pair a language-model planner with learned local policies. Scene graphs serve such planners as the state representation: Zhu et al. [34] plan long-horizon manipulation over geometric and symbolic scene graphs, Moma-Graph [35] builds state-aware scene graphs with vision-language models for embodied task planning, and Yu et al. [36] read a scene graph to detect a failure and replan around it. HALTER reuses this machinery and claims only the reset system that it supports. Composition is attractive for one reason. The number of distinct configurations a task can reach grows with its horizon, so any method that needs data or code for each configuration stops scaling, while a fixed skill library recombines into many of them.

Recent policies extend the horizon inside a single model instead. Long-VLA [37] and RoboTTT [38] scale a single policy to tasks with many stages, and they meet the same growth. LiLo-VLA [13] and BOSS [39] show that such a rollout can break at any subgoal, so the configurations a policy leaves behind multiply with the number of subgoals the task contains. Foresight [40] detects failures from rollout history, and REFLECT [41] explains them and prompts a planner to correct the task. Neither restores the scene for the next rollout. HALTER composes skills in the same way, but to undo the rollout rather than to perform it.

III. METHODOLOGY

We present HALTER in four parts. Section III-A formalizes the task, the rollout, and the reset problem, and it identifies the three questions that an autonomous evaluation system must answer. Section III-B presents the hierarchy HALTER uses to reset the scene. Section III-C describes the scene representation, and Section III-D describes how HALTER answers the three questions from that representation.

A. Problem Formulation

A natural language instruction x specifies a long-horizon manipulation task. Following the task and motion planning

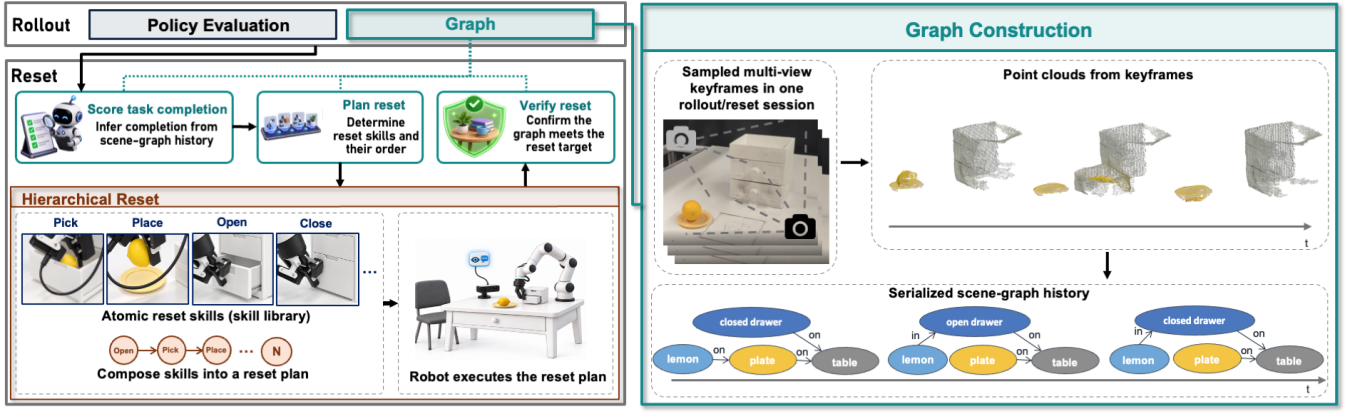


Fig. 2. **Overview of HALTER.** During a rollout, HALTER samples multi-view RGB-D keyframes, recovers a point cloud for every object, and serializes the result into a scene-graph history (right). A shared LLM interface reads that history to score task completion, to plan the reset, and to verify it (top left), and the planner composes atomic skills from the library into the plan that the robot executes (bottom left). **Black** arrows give the execution order, **green** marks the graph construction, and **brown** marks the hierarchical reset.

(TAMP) formulation [28], we write the task as a set of atomic skills $\mathcal{S} = \{a_1, \dots, a_H\}$ together with a causal partial order $\prec$ that their preconditions and effects induce, where each skill a_h applies an operator such as `Pick` or `Open` to a reference object o_h . Any total order consistent with $\prec$ executes the task correctly, and we write $\text{Lin}(\mathcal{S}, \prec)$ for the set of these linearizations. At each step a policy π under evaluation observes an RGB image I_t and the proprioceptive state q_t , and it emits an action u_t . A rollout starts from a physical scene state s_0 and ends at a terminal state s_T .

Evaluating a policy requires repeated rollouts from the same initial condition. We specify that condition by relations between objects rather than by object poses, $\mathcal{C}_0 = \{r_j(o_j, o'_j)\}$, for example $\{\text{on}(\text{lemon}, \text{plate}), \text{on}(\text{plate}, \text{table})\}$. Such a target constrains the relations and leaves each object pose free to vary within them. We state the problem this work addresses as follows. *Reset problem*: given the terminal state s_T of a rollout and a target $\mathcal{C}_0$, find a sequence of robot actions that carries the scene from s_T to a state s satisfying $s \models \mathcal{C}_0$.

Repeating this process autonomously raises two subproblems. (P1) *Terminal state coverage*. A policy under evaluation can fail at any skill, and it can follow any of the linearizations in $\text{Lin}(\mathcal{S}, \prec)$, so the number of terminal states a rollout can reach grows combinatorially with the task length H . Any reset that maps a terminal state directly to actions has to cover that set, and the demonstrations it needs grow with it. (P2) *Scene state estimation*. Meanwhile, the system has to read the state of the scene at three points: (A) which skill effects held during the rollout, which gives a completion score $S_{\text{comp}} \in [0, 1]$, the fraction of the H skills the rollout completed; (B) what the robot must do to bring the scene to a state satisfying $\mathcal{C}_0$; and (C) whether the state the robot reaches satisfies $\mathcal{C}_0$. All three read the same object: (A) and (C) evaluate relations on the scene state, and (B) plans over it.

B. Hierarchical Reset

Subproblem (P1) shows that any reset which maps a terminal state directly to actions needs data that grows with the number of reachable terminal states. HALTER therefore separates the reset into two layers. Fig. 2 (bottom left) shows the resulting two layers. At the low level, instead of preparing a complete reset policy for each terminal state, we prepare a set of atomic skills $\mathcal{K} = \{k^{(1)}, \dots, k^{(M)}\}$, where each skill performs one thing, such as picking up an object or closing a drawer (we assume this set suffices to return any reachable terminal state to a state satisfying $\mathcal{C}_0$). This library is its own set and need not coincide with the skills $\mathcal{S}$ that the evaluated task performs. The amount of data then depends on the size of the skill set rather than on the number of terminal states. We learn these skills with a VLA, because one VLA architecture covers diverse skills that act on different kinds of objects, from rigid objects to containers to articulated objects. At the high level, a planner reads the current scene state and the target $\mathcal{C}_0$, decides which skills to invoke and in what order, and emits an ordered reset plan $P = (k_1(\alpha_1), \dots, k_N(\alpha_N))$. The same set of skills therefore composes into different reset sequences for different terminal states.

C. Scene Graph Construction

Subproblem (P2) requires the system to read the state of the scene at three points. A raw image leaves object geometry implicit in its pixels, and a raw point cloud carries geometry without grounding it on named objects, so neither supports the relation tests that (A) and (C) perform nor the search space that (B) needs. HALTER therefore represents the scene as a spatial graph [34]–[36], which states objects, their metric attributes, and the relations between them explicitly, so that (A) and (C) evaluate relations directly on the graph and (B) grounds every step of its plan on a named object. HALTER builds one graph from each sampled RGB-D frame, as Fig. 2 (right) shows. Given the names of the task-relevant objects, it segments every object instance in the RGB image with an open-vocabulary segmentation model [42], [43], projects the

resulting masks into the registered depth image, and recovers a point cloud for each object. A node carries an object label together with geometric attributes that HALTER derives from that point cloud, such as position and spatial extent. For the edges, we design a set of geometry-based functions that take a pair of object point clouds and annotate the pair with a spatial relation such as *on*, *in*, or *under*. These functions read geometric quantities alone, so the same scene always yields the same annotation.

The moment at which HALTER builds these graphs matters as much as their content. The environment is partially observable by nature: the state inside a container or a drawer does not become visible by adding cameras or changing viewpoints. Observing the scene only after a rollout ends therefore cannot recover the state it stopped in. For example, no camera placement reveals whether the lemon sits inside a closed drawer. Reset has an advantage here, because it always follows a rollout that the system could observe from beginning to end, and the information that resolves such ambiguity is visible at the moment the state changes. This history is available to any evaluation system, yet only a system that accumulates state while the rollout runs retains it. HALTER therefore builds a graph at a fixed sampling rate during the rollout instead of estimating the state after the rollout ends. When an object becomes occluded, HALTER keeps its most recent valid observation together with the relations it supports, and updates both when the object reappears. The resulting sequence of graphs is the common input to the three uses in Section III-D.

D. Scoring, Planning, and Verification

HALTER uses an LLM for all three reasoning steps (Fig. 2, top left). It serializes each graph into object records and relation triples, so the three steps read the same input and differ only in their prompt.

Evaluation first has to report what the rollout achieved. The LLM reads the instruction and the graph sequence, decides for each skill whether its effect held at some point during the rollout, and returns $\hat{S}_{\text{comp}}$. The graph sequence matters here, because the effect of a skill can hold in the middle of a rollout and then become occluded or undone, and a reading taken at the terminal state alone misses it.

Once the rollout is scored, the scene has to return to C_0 , and the hierarchy of Section III-B assigns that decision to a planner above the skill library. The LLM reads the current state from the graph sequence, together with C_0 , the skill library $\mathcal{K}$, and a set of reference cases, and it returns the ordered reset plan P . Every skill argument in the plan names a node that the graph already contains, so the plan is a sequence the robot can execute rather than a description. The LLM chooses which skills to call and in what order, and the VLA skills produce the actions.

Executing a plan does not guarantee that the scene reached C_0 . HALTER therefore verifies at two levels. After each skill it checks whether the expected effect of that skill appears in the new graph. A missing effect stops the plan, because the remaining skills assume that this one succeeded. When

the plan runs to completion, HALTER checks the final graph against C_0 , and it requests human intervention if either check fails. Without this verification, the next rollout could begin from an unverified state. Checking the graph stands as a proxy for checking the physical scene.

IV. EXPERIMENTS

Our experiments ask three questions. **(Q1)** Does HALTER make autonomous evaluation of long-horizon tasks work with little human involvement? **(Q2)** Does the reset hierarchy transfer to held-out tasks without new demonstrations? **(Q3)** Does the scene graph of Section III-C account for these results? Section IV-A and Section IV-B describe the setup, and the three subsections that follow answer the questions in order.

A. Experimental Setup

1) *Tasks*: We evaluate on four long-horizon tasks in two tabletop scenes, which Fig. 3 shows. In the drawer scene, *Drawer-Store* opens the lower drawer, places a lemon inside, and closes it, and *Drawer-Retrieve* opens the drawer, takes a strawberry out, and places it on a plate. In the stove scene, *Pot-Cook* places a stock pot on the stove and cooks a carrot in it, and *Pan-Cook* places a pan on the stove and cooks a carrot in it. Every task takes several dependent steps, so a rollout can stop in many partially completed states, and each of them calls for a different reset sequence. For each of the four long-horizon tasks and three held-out tasks, all compared reset methods evaluate rollouts generated by the same task-specific BESO [44] policy checkpoint. Each rollout terminates when it completes the task or reaches a fixed time limit. We run 25 episodes for every task and method.

2) *Real-Robot Setup*: All experiments run on a Franka Panda 7-DoF arm with a Robotiq 2F-85 parallel gripper. Three RGB-D cameras observe the workspace: two DepthAI OAK-D Lite cameras, one at the side of the table and one on the gripper, and one ZED camera on the opposite side.

3) *Baselines*: We compare HALTER against two systems. **AutoEval** [12] learns one task-specific reset policy per task. To hold the LLM backbone constant, we use the same LLM as HALTER for task scoring and post-reset verification. Its reset policy is an X-VLA [45] fine-tuned on complete reset trajectories, with 30 to 40 demonstrations for each failure case and for the successful case. **MP skills** keeps every component of HALTER except the skills themselves, and executes each planned skill with a motion-planning stack: GraspNet generates the grasp pose and CuRobo plans the motion that reaches it.

4) *Metrics*: **Scoring accuracy** is the fraction of episodes whose predicted completion score $\hat{S}_{\text{comp}}$ matches the manually annotated completed-skill fraction. **Plan validity** is the fraction of episodes with a valid reset-skill sequence for restoring C_0 ; it does not apply to AutoEval, which emits actions rather than plans. **Verification accuracy** is the fraction of episodes in which the post-reset verification verdict matches the manual annotation of whether C_0 was restored. **Reset SR** is the fraction of episodes whose reset

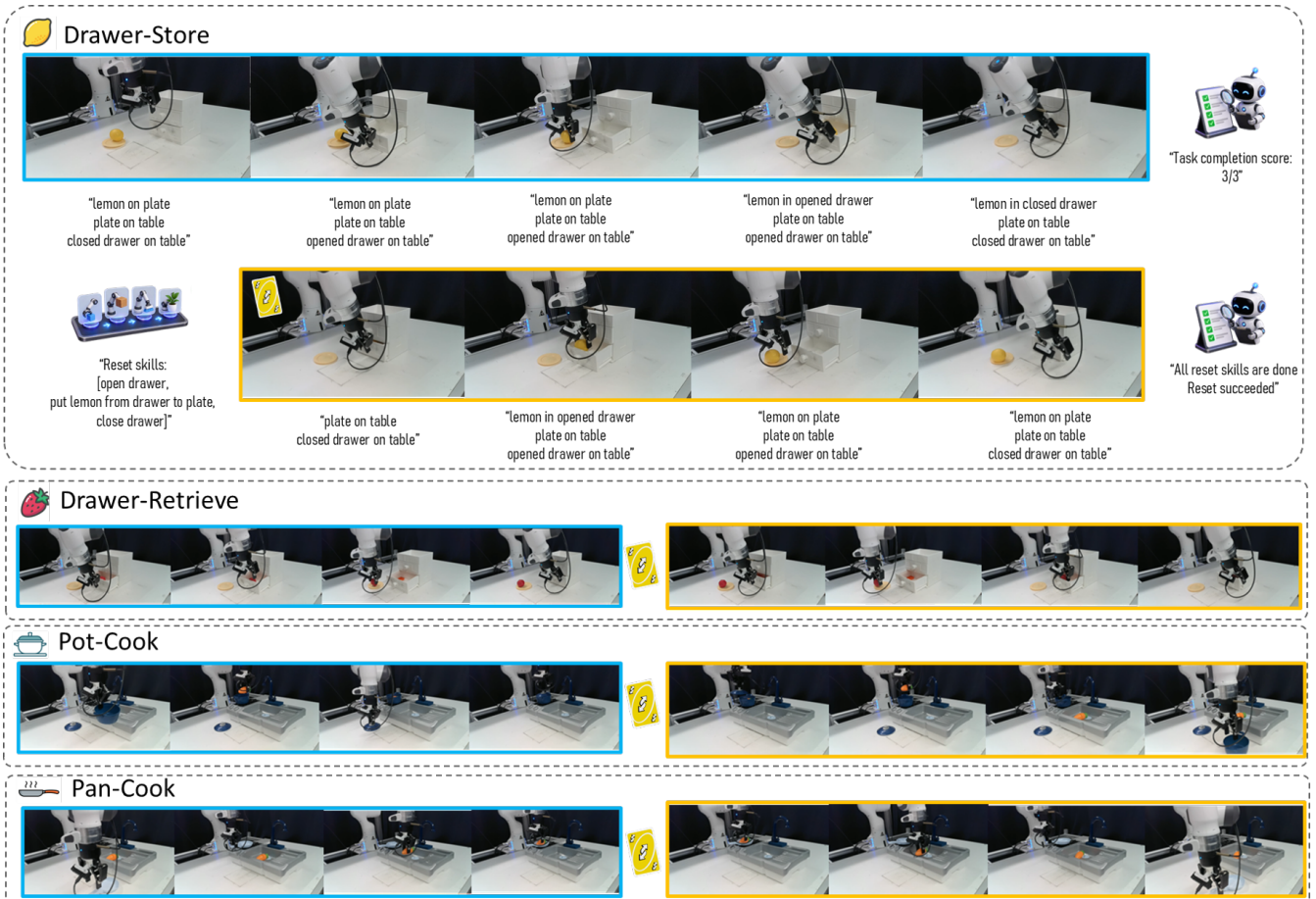


Fig. 3. Four evaluated long-horizon tasks across two tabletop scenes.

restores $\mathcal{C}_0$, according to the manual final-scene annotation. All four metrics use 25 episodes per task, and the main-task averages pool 100 episodes. Uncertainties are binomial standard errors over episodes. The last three metrics measure cost. **Interventions** counts episodes requiring a human, **cycle time** is the wall-clock seconds per episode, and **operator time** charges every intervention the cost of one manual reset.

B. Implementation Details

1) *Reset Skill Policies*: For each atomic skill in $\mathcal{K}$, we train two X-VLA policies using 30 to 50 demonstrations per skill. The drawer-related tasks require six skills: *Open the lower drawer*, *Close the lower drawer*, *Put the lemon from the drawer back on the plate*, *Put the lemon from the table back on the plate*, *Put the strawberry from the plate back in the drawer*, and *Put the strawberry from the table back in the drawer*.

The kitchen-related tasks likewise require six skills: *Put the pot lid back in place*, *Put the stock pot from the stove back in place*, *Put the stock pot from the table back in place*, *Put the pan from the stove back in place*, *Put the pan from the table back in place*, and *Put the carrot back in the sink*.

2) *Scene Graph Construction*: HALTER samples RGB-D keyframes at 0.5 Hz, segments task-relevant objects using GroundedSAM [42], [43], and projects the resulting masks into the registered depth image to recover object-level

point clouds. It infers spatial relations from world-coordinate bounding boxes: *on* from vertical proximity and horizontal support, *in* from volumetric containment. Drawer states are determined from their positions along the world y -axis.

3) *LLM Prompting*: We use Qwen3.5-397B for task-completion scoring, reset planning, and reset verification, with a stage-specific prompt for each function. For each task, the scoring and planning prompts contain four task-specific rollout reference cases: one successful execution and three failed executions. Each case pairs a chronological scene-graph history with the corresponding reset-skill sequence. Reset verification uses four successful, skill-specific reference cases. The rollout cases are defined separately for each task, whereas reset cases can be reused across tasks that invoke the same atomic skill.

C. Autonomous Evaluation and Reset (Q1)

Table I reports the main results. HALTER restores the scene in 76.0% of the episodes, which is 24 points above AutoEval and 11 points above MP skills. Pooled over the 100 episodes, the gap over AutoEval is more than three standard errors. AutoEval trains one reset policy on demonstrations that span a whole long-horizon reset, and a single policy degrades as that horizon grows, whereas every skill of HALTER covers one step. MP skills shares the scoring and the planning of HALTER, so the table compares it

TABLE I
MAIN RESULTS ON THE FOUR LONG-HORIZON TASKS.

Method	Metric	Task (%)				Avg (%)
		Drawer-Store	Drawer-Retrieve	Pot-Cook	Pan-Cook	
AutoEval	Scoring accuracy	72	64	80	88	76.0 $\pm$ 4.3
	Plan validity	-	-	-	-	-
	Verification accuracy	76	80	80	76	78.0 $\pm$ 4.1
	Reset SR	52	64	16	76	52.0 $\pm$ 5.0
MP skills	Reset SR	68	76	72	44	65.0 $\pm$ 4.8
HALTER (ours)	Scoring accuracy	100	96	80	84	90.0 $\pm$ 3.0
	Plan validity	92	84	72	88	84.0 $\pm$ 3.7
	Verification accuracy	96	84	88	96	91.0 $\pm$ 2.9
	Reset SR	88	84	48	84	76.0 $\pm$ 4.3

TABLE II
HUMAN EFFORT AND TIME OVER THE 100 EPISODES. OPERATOR TIME CHARGES EACH INTERVENTION ONE MANUAL RESET.

Method	Interventions	Cycle time (s/ep)	Operator time (min)
Manual reset	100	53	88
AutoEval	46	104	44
MP skills	34	137	29
HALTER (ours)	25	121	24

on end-to-end reset alone: it picks a grasp pose once and then executes a fixed trajectory, so a small deviation is enough to fail the skill, whereas the learned skills act on the current observation. Scoring accuracy reaches 90.0% and plan validity 84.0%, so each step of the graph pipeline is reliable on its own and not only in combination.

Verification separates the two systems further. Across the 100 episodes, HALTER achieves a verification accuracy of 91.0%, against 78.0% for AutoEval. These errors have asymmetric costs: false acceptance can start the next rollout from an incorrect state, whereas false rejection triggers an unnecessary reset attempt or human intervention.

Table II reports what the evaluation costs a human. HALTER cuts the interventions from 100 to 25 and the operator time from 88 to 24 minutes, and it raises the mean number of episodes between interventions from 1.0 to 4.0. Additional LLM reasoning between successive reset skill executions makes each episode slightly longer in wall-clock time than for AutoEval. The operator, however, takes part in far fewer episodes, and operator availability rather than wall-clock time bounds how many rollouts an evaluation campaign can afford. The advantage also grows with the length of the reset plan. Stratified by that length on the three tasks for which we measured it, HALTER finishes a one-skill reset 2 to 18 seconds faster than MP skills, and a three-skill reset 24 to 57 seconds faster, with every three-skill gap significant at $p < 0.01$. Longer resets are exactly the case that a long-horizon rollout produces.

Pot-Cook is the weakest task for every method, and it is the only task on which MP skills resets more reliably than HALTER, 72% against 48%. It carries the longest reset plans of the four tasks, and the learned skills handle the stock pot less reliably than the analytic grasps do, so the gap over MP

TABLE III
TRANSFER TO THREE HELD-OUT TASKS, 25 EPISODES EACH. HALTER REUSES ITS SKILL LIBRARY AND TRAINS NOTHING NEW.

Method	Metric	Held-out task (%)			
		Store	Retrieve	Pot	Avg (%)
AutoEval	Scoring accuracy	72	72	80	74.7 $\pm$ 5.0
	Plan validity	-	-	-	-
	Verification accuracy	84	80	84	82.7 $\pm$ 4.4
	Reset SR	0	4	0	1.3 $\pm$ 1.3
HALTER (ours)	Scoring accuracy	100	100	84	94.7 $\pm$ 2.6
	Plan validity	92	96	80	89.3 $\pm$ 3.6
	Verification accuracy	96	92	88	92.0 $\pm$ 3.1
	Reset SR	84	84	56	74.7 $\pm$ 5.0

skills rests on the other three tasks. HALTER still resets Pot-Cook three times as often as AutoEval, 48% against 16%, and it scores the rollout correctly on 80% of the episodes there.

D. Composing Skills for Held-Out Tasks (Q2)

Table III reports three held-out tasks that HALTER has not previously evaluated. These tasks retain the same objects but change the task goals. In the drawer scene, a successful rollout leaves either the lemon on the table with the drawer closed or the strawberry on the table rather than on the plate. In the kitchen scene, it leaves the stock pot on the stove with its lid on top without carrot. These task goals produce terminal states that require reset sequences different from those of the original tasks. HALTER reaches 74.7% reset SR and 89.3% plan validity on them without a single new demonstration. AutoEval reaches 1.3%, because it fits one reset policy to the skill sequence of the task it was trained on, and that sequence no longer applies. The difference lies in what each system learns: a library of atomic skills recombines into a new order, while a single policy cannot. This is the compositional generalization that the hierarchy is meant to provide.

The same property also changes what a demonstration has to contain. A demonstration of one atomic skill is far shorter than a demonstration of a whole reset, so resetting the four original tasks costs HALTER 105k timesteps against 138k for AutoEval, 24% less data, and the same skills serve every

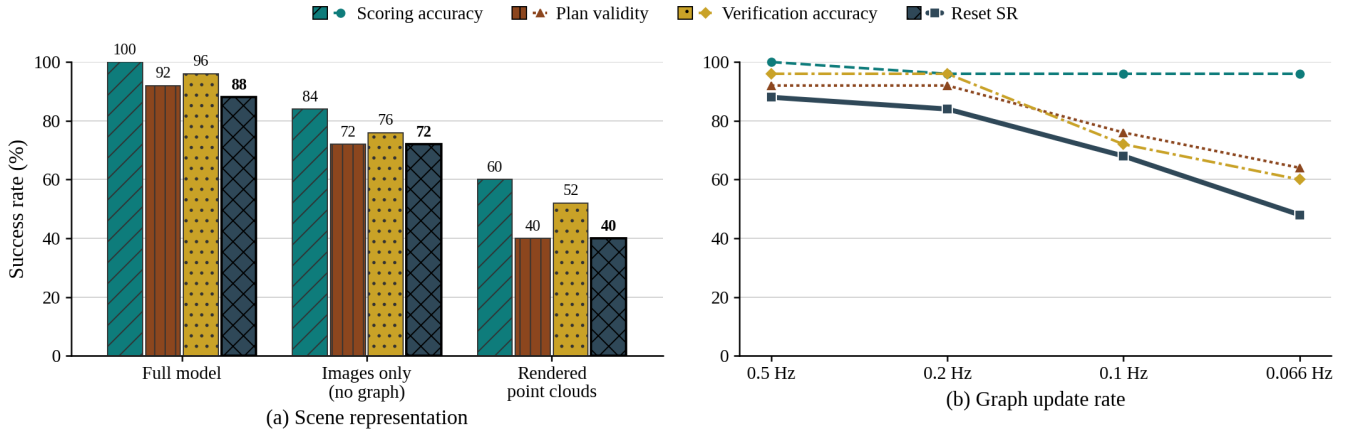


Fig. 4. Ablation of the scene graph on Drawer-Store, 25 episodes per configuration. (a) Replacing the graph with the images themselves, or with rendered point clouds, lowers every metric. (b) As the graph update rate drops, plan validity and reset SR fall while scoring accuracy stays flat.

task of a scene.

E. Ablation of the Scene Graph (Q3)

Fig. 4 ablates the scene graph on Drawer-Store. Replacing the graph with the images themselves lowers plan validity from 92% to 72%, verification accuracy from 96% to 76%, and reset SR from 88% to 72%. Passing rendered point clouds instead lowers them further, to 40%, 52%, and 40%, so the gain does not come from three-dimensional appearance but from turning geometry into explicit relations.

The rate at which HALTER updates the graph matters as well. Performance holds at 0.2 Hz, and below that rate plan validity falls from 92% to 64%, verification accuracy from 96% to 60%, and reset SR from 88% to 48%, while scoring accuracy stays at 96%. This follows Section III-C: a plan needs the states that the rollout passed through, whereas the completion score can often be recovered from the last few keyframes.

V. CONCLUSION

We presented HALTER, an evaluation harness that scores a long-horizon rollout on real hardware and restores the scene for the next one with little human involvement. HALTER builds a spatial scene graph online from RGB-D keyframes, and an LLM reads that graph to score the rollout, to plan a reset over a library of atomic skills, and to verify that the reset succeeded. Planning over a library makes the demonstration cost scale with the size of that library rather than with the number of states a rollout can terminate in, and on four real tasks HALTER restores the scene more reliably than a per-task reset policy while cutting the operator time by 72%. On three held-out tasks, it recombines the same skills without collecting anything new, and an ablation attributes the gain to the graph, since replacing it with images or with rendered point clouds lowers every metric. Two limitations remain. A reset that needs a skill the rollout never performs still calls for demonstrations of its own, and HALTER still needs a handful of reference cases written for each task, although it trains no classifier and collects no labeled images. Reversing the forward demonstrations that a task already

provides is a promising way to obtain the missing skills without a new collection effort.

ACKNOWLEDGMENT

We used Claude and ChatGPT to polish the text of this paper and to generate the illustrations in Fig. 1 and Fig. 2. The authors reviewed every generated item, and all technical claims, experiments, and results are our own.

REFERENCES

- [1] Physical Intelligence, B. Ai, A. Amin, R. Aniceto, A. Balakrishna, G. Balke, K. Black, G. Bokinsky, S. Cao, T. Charbonnier *et al.*, “ $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities,” *arXiv preprint arXiv:2604.15483*, 2026.
- [2] M. J. Kim, C. Finn, and P. Liang, “Fine-tuning vision-language-action models: Optimizing speed and success,” in *Robotics: Science and Systems XXI*, 2025.
- [3] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, and J. Gu, “Cosmos Policy: Fine-tuning video models for visuomotor control and planning,” *arXiv preprint arXiv:2601.16163*, 2026.
- [4] Q. Zhang, L. Li, L. Zhang, S. Yang, Y. Luo, S. Li, R. Wang, J. Wang, J. Shao, G. Xu *et al.*, “Native video-action pretraining for generalizable robot control,” *arXiv preprint arXiv:2607.08639*, 2026.
- [5] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Munoz, X. Yao, R. Zurbrugg, N. Rudin *et al.*, “Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning,” *arXiv preprint arXiv:2511.04831*, 2025.
- [6] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-k. Chan *et al.*, “ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI,” *arXiv preprint arXiv:2410.00425*, 2024.
- [7] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, “robosuite: A modular simulation framework and benchmark for robot learning,” *arXiv preprint arXiv:2009.12293*, 2020.
- [8] N. Khargonkar, S. H. Allu, Y. Lu, J. J. P. B. Prabhakaran, and Y. Xiang, “SceneReplica: Benchmarking real-world robot manipulation by creating replicable scenes,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8258–8264.
- [9] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina *et al.*, “A careful examination of large behavior models for multitask dexterous manipulation,” *Science Robotics*, vol. 11, no. 113, p. eaea6201, 2026.
- [10] T. Chen, Y. Chen, Z. Li, J. Tang, K. Su, H. Lu, W. Wan, B. Chen, S. Liu, H. Yan *et al.*, “RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies,” *arXiv preprint arXiv:2607.04434*, 2026.

- [11] Y. Jangir, Y. Zhang, P.-C. Lo, K. Yamazaki, C. Zhang, K.-H. Tu, T.-W. Ke, L. Ke, Y. Bisk, and K. Fragkiadaki, "RobotArena ∞ : Scalable robot benchmarking via real-to-sim translation," *arXiv preprint arXiv:2510.23571*, 2025.
- [12] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine, "AutoEval: Autonomous evaluation of generalist robot manipulation policies in the real world," in *Proceedings of The 9th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 305. PMLR, 2025, pp. 1997–2017.
- [13] Y. Yang, S. Cheng, Y. Fang, H. Bharadhwaj, M. Ding, G. Bertasius, and D. Szafrir, "LiLo-VLA: Compositional long-horizon manipulation via linked object-centric policies," *arXiv preprint arXiv:2602.21531*, 2026.
- [14] C. Grislain, H. Rahimi, O. Sigaud, and M. Chetouani, "I-FailSense: Towards general robotic failure detection with vision-language models," *arXiv preprint arXiv:2509.16072*, 2025.
- [15] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia, "SpatialVLM: Endowing vision-language models with spatial reasoning capabilities," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 14455–14465.
- [16] A. Pothiraj, E. Stengel-Esklin, J. Cho, and M. Bansal, "CAPTURE: Evaluating spatial reasoning in vision language models via occluded object counting," in *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2025, pp. 8001–8010.
- [17] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khaidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "DINOv2: Learning robust visual features without supervision," *Transactions on Machine Learning Research*, vol. 2024, 2024.
- [18] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "SAM 2: Segment anything in images and videos," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 28085–28128.
- [19] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman, "REPLAB: A reproducible low-cost arm benchmark platform for robotic learning," 2019. [Online]. Available: <https://arxiv.org/abs/1905.07447>
- [20] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "FurnitureBench: Reproducible real-world benchmark for long-horizon complex manipulation," in *Robotics: Science and Systems XIX*, 2023.
- [21] P. Atreya, K. Pertsch, T. Lee, M. J. Kim, A. Jain, A. Kuramshin, C. Eppner, C. Neary, E. Hu, F. Ramos, J. Tremblay, K. Arora, K. Ellis, L. Macesanu, M. T. Villasevil, M. Leonard, M. Cho, O. Aslan, S. Dass, J. Wang, W. Reger, X. Yuan, X. Yang, A. Gupta, D. Jayaraman, G. Berseth, K. Daniilidis, R. Martin-Martin, Y. Lee, P. Liang, C. Finn, and S. Levine, "RoboArena: Distributed real-world evaluation of generalist robot policies," 2025. [Online]. Available: <https://arxiv.org/abs/2506.18123>
- [22] W. Xiao, J. Xie, T. Zhang, H. Lin, L. M. Fu, H. Xue, J. Lu, Y. Yang, C. Dai, Z. Wang, J. Wu, G. Wang, S. S. Sastry, K. Goldberg, L. J. Fan, Y. Zhu, and G. Shi, "ENPIRE: Agentic robot policy self-improvement in the real world," 2026. [Online]. Available: <https://arxiv.org/abs/2606.19980>
- [23] Y. Wang, K. Zhu, Y. Zhong, L. Wang, J. Yang, and F. Zheng, "RADAR: Closed-loop robotic data generation via semantic planning and autonomous causal environment reset," 2026. [Online]. Available: <https://arxiv.org/abs/2603.11811>
- [24] B. Wang, Z. Zhang, Z. Yang, P. Gu, S. Wang, X. Wang, X. Ze, Y. Chang, G. Zhao, J. Shao, G. Huang, H. Liu, Y. Zhang, W. Xue, C. Guan, C. Pu, Y. Guo, X. Wang, and Z. Zhu, "Zero2Skill: Bootstrapping robot skills through autonomous data collection, training, and deployment," 2026. [Online]. Available: <https://arxiv.org/abs/2607.14047>
- [25] Y. Zhang, H. Zhang, F. Gao, X. Li, Z. Liu, C. Zhu, J. Qiu, Y. Yan, J. Liu, W. Tang, Z. Fang, Y. Nie, C. Wei, Y. Wang, W. Ding, and C. Yu, "Harness VLA: Steering frozen vlars into reliable manipulation primitives via memory-guided agents," 2026. [Online]. Available: <https://arxiv.org/abs/2607.08448>
- [26] H. Liu, X. Li, S. Yao, P. Shi, T. Zhou, J.-B. Huang, F. Huang, and J. Mao, "Guava: An effective and universal harness for embodied manipulation," 2026. [Online]. Available: <https://arxiv.org/abs/2606.18363>
- [27] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1470–1477.
- [28] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.
- [29] S. Cheng and D. Xu, "LEAGUE: Guided skill learning and abstraction for long-horizon manipulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6451–6458, 2023.
- [30] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Conference on Robot Learning*. PMLR, 2023, pp. 2905–2925.
- [31] S. Cheng, C. R. Garrett, A. Mandlekar, and D. Xu, "NOD-TAMP: Generalizable long-horizon planning with neural object descriptors," in *Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 2024, pp. 1310–1339.
- [32] M. Dalal, T. Chiruvolu, D. S. Chaplot, and R. Salakhutdinov, "Plan-Seq-Learn: Language model guided rl for solving long horizon robotics tasks," in *International Conference on Learning Representations (ICLR)*, 2024.
- [33] M. Dalal, M. Liu, W. Talbott, C. Chen, D. Pathak, J. Zhang, and R. Salakhutdinov, "Local policies enable zero-shot long-horizon manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 13875–13882.
- [34] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Ieee, 2021, pp. 6541–6548.
- [35] Y. Ju, Y. Liang, Y.-J. Wang, G. Nandiraju, Y. Ju, S. J. Lee, Q. Gu, E. Hsieh, F. Huang, and K. Sreenath, "Momagraph: State-aware unified scene graphs with vision-language models for embodied task planning," in *International Conference on Learning Representations*, vol. 2026, 2026, pp. 96361–96386.
- [36] C. R. Yu, D. Chae, D. Seo, S. Lee, H. Im, and J. Kim, "Scene graph-guided proactive replanning for failure-resilient embodied agent," *arXiv preprint arXiv:2508.11286*, 2025.
- [37] Y. Fan, S. Bai, X. Tong, P. Ding, Y. Zhu, H. Lu, F. Dai, W. Zhao, Y. Liu, S. Huang, Z. Fan, B. Chen, and D. Wang, "Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation," in *Proceedings of The 9th Conference on Robot Learning*. PMLR, 2025, pp. 2018–2037.
- [38] Y. Jiang, Y. Chebotar, R. Zheng, F. Hu, Y. Ge, J. Wu, T. Dai, S. Reed, L. Fei-Fei, Y. Zhu, and L. J. Fan, "RoboTTT: Context scaling for robot policies," 2026. [Online]. Available: <https://arxiv.org/abs/2607.15275>
- [39] Y. Yang, L. Zhao, M. Ding, G. Bertasius, and D. Szafrir, "BOSS: Benchmark for observation space shift in long-horizon task," *IEEE Robotics and Automation Letters*, vol. 10, no. 9, pp. 8882–8889, 2025.
- [40] H. Zhang, Y. Lu, B. Wang, X. Kang, Y.-L. Kuo, Z. Cheng, M. Wang, and O. C. Jenkins, "Foresight: Failure detection for long-horizon robotic manipulation with action-conditioned world model latents," 2026. [Online]. Available: <https://arxiv.org/abs/2606.23085>
- [41] Z. Liu, A. Bahety, and S. Song, "REFLECT: Summarizing robot experiences for failure explanation and correction," in *Conference on Robot Learning (CoRL)*, 2023, pp. 3468–3484.
- [42] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.
- [43] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.
- [44] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.
- [45] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, Y.-Q. Zhang, J. Pang, J. Liu, T. Wang, and X. Zhan, "X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model," *arXiv preprint arXiv:2510.10274*, 2025.