MANIUNIT: A MANIPULATION SKILL DATASET AND BENCHMARK FOR LONG-HORIZON TASKS

Anonymous authors
Paper under double-blind review

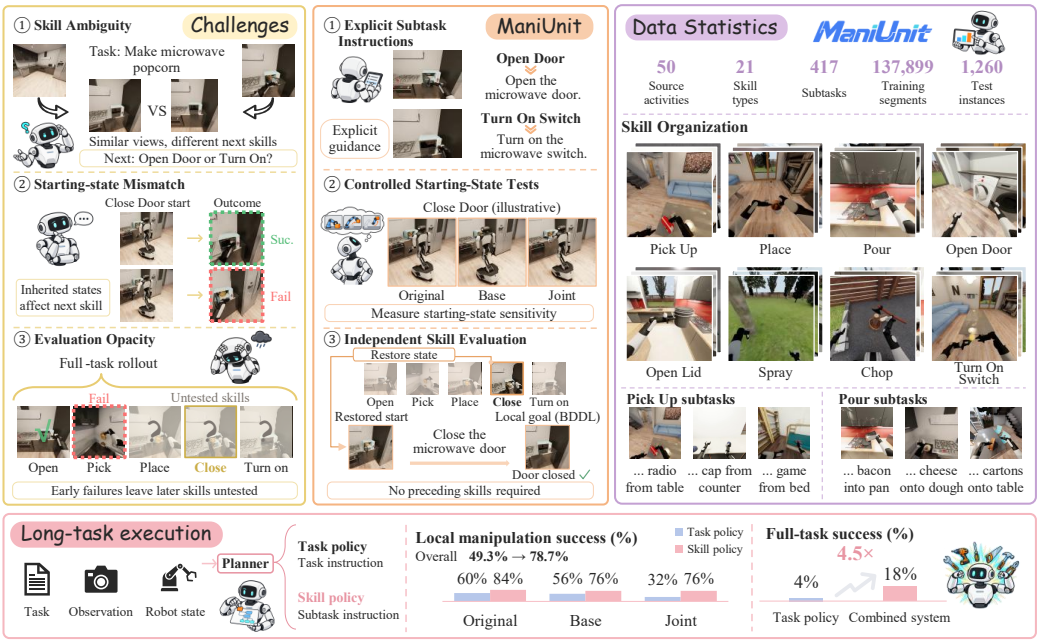


Figure 1: **ManiUnit overview. Left**: Long-horizon execution poses three challenges for skill learning and evaluation. **Center**: Correspondingly, ManiUnit provides explicit subtask instructions to guide manipulation. Controlled tests measure starting-state sensitivity, while restored states and local BDDL goals enable independent skill evaluation. **Right**: ManiUnit organizes manipulation segments by skill type and subtask. **Bottom**: In a focused study on two long-horizon activities, skill-level training improves local manipulation success, and combining the task and skill policies through a planner improves full-task success.

ABSTRACT

Long-horizon mobile manipulation requires a robot to navigate multi-room environments and execute a sequence of manipulation skills under a single natural language instruction. Learning and evaluating these skills present three challenges: similar observations under a fixed task instruction may make skill selection ambiguous; even when a preceding skill succeeds, the robot state inherited by the next skill may deviate from its demonstrated starting states and affect execution; and task-level metrics hinder skill-specific diagnosis, while early failures leave later skills untested. We therefore introduce ManiUnit, a manipulation skill dataset and benchmark built from 50 BEHAVIOR-1K activities. Its dataset contains 137,899 segments across 21 skill types and 417 subtasks, and its benchmark contains 1,260 test instances. Correspondingly, ManiUnit pairs each segment with an explicit subtask instruction; measures sensitivity to perturbations of the robot's starting base position or joint configuration; and restores intermediate simulator states and defines local success conditions so that each skill can be evaluated without executing preceding stages. Evaluations of representative vision-language-action (VLA)

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.

policies show that similar aggregate scores can hide substantial per-skill differences. The tested starting-state perturbations also degrade execution: on the full benchmark, joint perturbations reduce success rates by approximately 56% relative to those from demonstrated starting states. On two long-horizon activities, a skill policy trained on ManiUnit segments achieves 78.7% local manipulation success, compared with 49.3% for a task policy trained on complete demonstrations. The trained skills further support complete-task execution on these activities, as coordinating the task and skill policies through a planner raises full-task success from 4.0% to 18.0%.

1 INTRODUCTION

Enabling robots to complete long-horizon mobile manipulation tasks in everyday environments is a central challenge for embodied AI. Unlike tabletop manipulation, which operates in a fixed workspace, this setting requires robots to navigate across multi-room environments and execute a sequence of interdependent manipulation skills guided by a single natural language instruction. Recent vision-language-action (VLA) models (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b;a; NVIDIA et al., 2025; Gemini Robotics Team et al., 2025) have achieved strong results on tabletop benchmarks: StarVLA-π (StarVLA Community, 2026), for example, reaches 95.7% success on LIBERO (Liu et al., 2023) and 88.8% under the randomized setting of RoboTwin 2.0 (Chen et al., 2026). Despite this progress, reliable execution of long-horizon tasks remains challenging: the winning entry in the 2025 BEHAVIOR Challenge achieved a full-task success rate of only 12.4% on the held-out test set (Larchenko et al., 2025), leaving substantial room for improvement.

Consider one such task, "make microwave popcorn," which requires navigation interleaved with opening the microwave door, picking up the popcorn bag, placing it inside, closing the door, and turning on the microwave. Such sequences pose three challenges for skill learning and evaluation. (1) **Skill ambiguity**: when a policy follows the same task instruction throughout execution, it needs to infer the current stage from its observations and act accordingly. This inference can break down when different stages share similar visual context: before loading food, the robot faces a closed microwave door and must open it, but after placing the food inside and closing the door, it again faces the same closed door yet must now turn on the microwave instead of opening the door. (2) **Starting-state mismatch**: during sequential execution, even a successfully completed skill may leave the robot in a base position or joint configuration that differs from the starting states in the next skill's training demonstrations, and this mismatch can cause a policy that succeeds from demonstrated starting states to fail when following a preceding skill. (3) **Evaluation opacity**: overall task success reflects both navigation and manipulation, making it difficult to identify which skills limit performance; moreover, an early failure can prevent later skills from being attempted.

These challenges motivate treating individual manipulation operations as units of training and evaluation. We therefore introduce ManiUnit, a manipulation skill dataset and benchmark built by segmenting long-horizon BEHAVIOR-1K demonstrations (Li et al., 2023) into such operations. Figure 1 maps each challenge to the corresponding design choice. **For skill ambiguity**, ManiUnit attaches an explicit subtask instruction to each segment, naming the operation, the objects it acts on, and the spatial context, keeping the intended operation explicit during skill learning and evaluation. **For starting-state mismatch**, ManiUnit evaluates each skill under three conditions: the demonstrated state, a base-position perturbation, and a joint-configuration perturbation. These controlled tests measure how variations in the robot's starting state affect skill execution. **For evaluation opacity**, ManiUnit constructs test instances by restoring intermediate simulator states and defining local success conditions for each subtask, so every skill can be evaluated independently without first executing the stages that precede it. The dataset covers 50 activities with 21 skill types, 417 subtasks, and 137,899 segments with a mean duration of 17.9 seconds, each paired with head and wrist RGB streams, robot states, and actions. The benchmark contains 1,260 test instances, drawn from 20 operation starts per skill type under the three starting-state conditions.

Evaluating representative VLA policies on ManiUnit shows that similar aggregate scores can hide substantial differences in per-skill performance. The evaluations also reveal pronounced sensitivity to the tested starting-state perturbations: on the full benchmark, joint perturbations reduce success rates by approximately 56% relative to those from demonstrated starting states. On Turning On

Radio and Picking Up Trash, we compare task and skill policies initialized from the same pretrained weights. The skill policy trained on ManiUnit segments achieves 78.7% local manipulation success, compared with 49.3% for the task policy trained on complete demonstrations. To deploy these skills in complete tasks, a planner coordinates the task and skill policies and supplies the appropriate subtask instructions (Figure 1, bottom). On the same activities, the combined system achieves 18.0% full-task success, compared with 4.0% for the task policy alone.

Our contributions are threefold. (1) We introduce the ManiUnit dataset: 137,899 segments from 50 BEHAVIOR-1K activities with explicit subtask instructions, organized by skill type and subtask to support mixed-skill training and targeted data selection. (2) We construct the ManiUnit benchmark: 1,260 test instances from restored intermediate states with local success conditions and separate base and joint perturbations, enabling independent per-skill evaluation and measurement of starting-state sensitivity. (3) We evaluate representative VLA policies to characterize skill performance and starting-state sensitivity, and show on two long-horizon activities that ManiUnit-trained skills improve local manipulation success and support complete-task execution through policy composition.

2 ManiUnit Dataset and Benchmark

2.1 Overview

To study manipulation skills independently within long-horizon tasks, we construct ManiUnit from the 2025 Challenge demonstrations of 50 BEHAVIOR-1K activities (Li et al., 2023). ManiUnit comprises a dataset of manipulation segments paired with explicit subtask instructions and a benchmark for evaluating selected operations from restored starting states. Figure 2 summarizes how the source demonstrations are processed into these training and evaluation resources.

The dataset distinguishes three levels of organization. A ***skill type*** identifies an operation category, such as Pour. A ***subtask*** is defined by an instruction specifying the operation, objects, and spatial context, such as "pour the grated cheese from the tupperware onto the pizza dough." A ***segment*** records one demonstrated execution of that subtask. Grouping segments by skill type and subtask allows researchers to pool demonstrations across source activities or select data for particular operations. **ManiUnit-Full** contains 21 skill types, 417 subtasks, and 137,899 segments. We also curate **ManiUnit-Lite**, a subset of 10,588 segments across 7 skill types, to reduce the computational cost of repeated experiments. Table 4 in Appendix B.1 reports the per-skill inventory of ManiUnit-Full.

For evaluation, we select operation starts from the source demonstrations and restore the corresponding robot and scene states. For each selected subtask, we manually define a local BDDL goal that specifies the object states and relations required to complete the manipulation operation. The restored state, subtask instruction, and local goal together form a test instance, allowing the operation to be evaluated without executing preceding task stages. To measure sensitivity to starting-state variations, we construct three conditions for each selected start: Original, Base Perturbation, and Joint Perturbation. The Full and Lite evaluation suites contain 1,260 and 186 test instances, respectively. Sections 2.2 and 2.3 describe the dataset and benchmark construction procedures.

2.2 Dataset Construction

As shown in Figure 2, the source demonstrations contain RGB observations, robot states, and actions, together with temporal annotations of navigation and manipulation. Each annotation records an action label, the participating objects, and the action's start and end frames. However, these intervals do not always correspond to the manipulation segments needed for training. For example, pushing an object into reach and picking it up may be annotated separately despite contributing to the same Pick Up operation, while a recorded pickup interval may also include navigation before grasping or inactivity afterward. To preserve complete manipulation operations while reducing surrounding navigation and inactivity, we extract and merge the relevant action intervals and refine their boundaries. We then curate an explicit subtask instruction for each resulting segment.

Segment extraction and merging. We first extract manipulation intervals based on the annotated actions. For each source activity, we manually specify which annotations to retain and apply this selection across its episodes. For example, in the Picking Up Trash activity, we retain all annotated

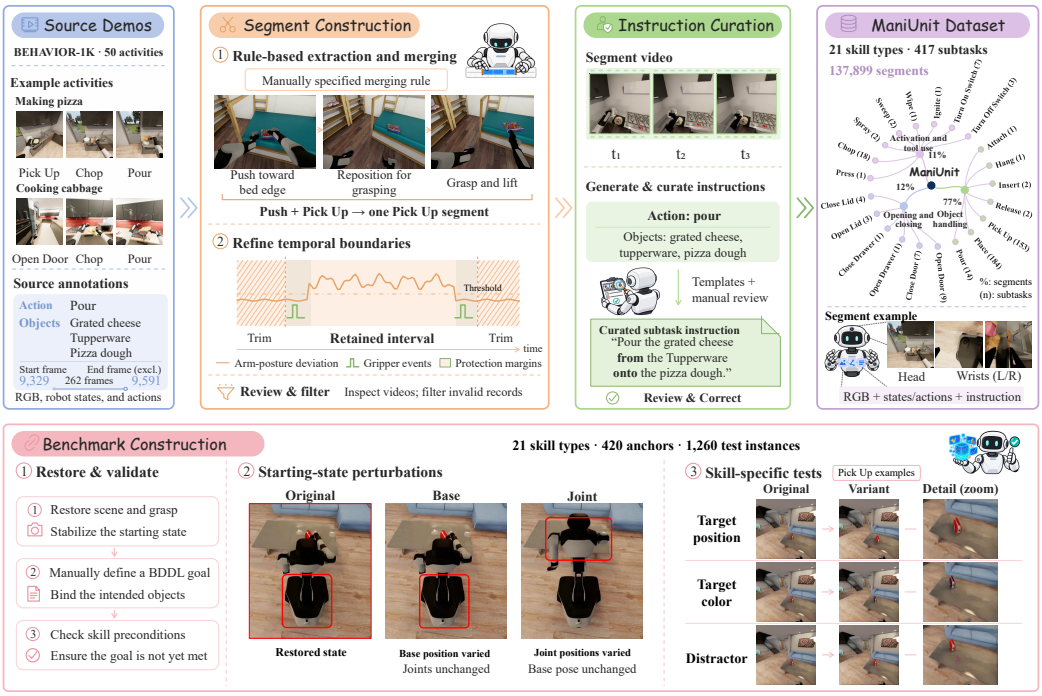


Figure 2: **ManiUnit dataset and benchmark construction.** Top: Source demonstrations are converted into manipulation segments with curated subtask instructions. Bottom: Each evaluation anchor marks a selected operation start, whose restored state, subtask instruction, and local BDDL goal enable independent skill evaluation across original and perturbed starting states.

pickup and placement actions, each yielding an initial segment. The recorded start and end frames of each retained annotation provide the initial segment boundaries.

Since supporting motions may be annotated separately from the operation they serve, we define merging rules to combine them into complete manipulation segments. In the board-game example in Figure 2, pushing the game toward the bed edge prepares it for grasping. We therefore combine the push and subsequent pickup into one Pick Up segment. The merged segment spans the start of the first contributing action through the end of the last, retaining all intervening frames. Appendix B.2 details the extraction and merging rules.

Temporal boundary refinement. We refine the boundaries of the extracted and merged segments to reduce navigation and inactivity around the target operation. Reaching for and handling objects often require raising or extending the arms, so we use deviations from a fixed lowered-arm reference posture as a heuristic cue for manipulation. We measure this deviation as

$$d_t = \max_{j \in \mathcal{J}} \left| \bar{q}_{t,j} - q_j^{\text{ref}} \right|, \tag{1}$$

where $\mathcal{J}$ denotes the monitored joints of both arms, $\bar{q}_{t,j}$ is the smoothed position of joint j at frame t, and q_j^{ref} is its reference position. To filter out brief fluctuations, we identify intervals in which d_t remains above a threshold for a minimum number of consecutive frames. We retain the continuous span from the start of the first such interval to the end of the last. To preserve grasp and release motions, we also detect gripper events from changes in finger-joint positions. We adjust the segment boundaries to retain the detected gripper events and a short interval before the first event and after the last. We also apply checks against excessive trimming and manually inspect selected segment videos to assess the resulting boundaries. Appendix B.3 provides refinement details.

Instruction curation. The extracted segments retain their action labels and participating objects from the source annotations. We group segments by their action and object combination. For each

group, we convert the simulator object identifiers to readable names and define an instruction template that specifies the action and the objects' roles. For example, Pour segments with grated cheese, tupperware, and pizza dough yield "pour the grated cheese from the tupperware onto the pizza dough," explicitly naming what is transferred, its source, and its destination. We then manually review every episode across all source activities, checking each segment's instruction against its video. This review refines the description of the operation, objects, and spatial relations, and corrects errors inherited from the source annotations.

Data organization. The curated segments are grouped by skill type and subtask instruction, supporting pooled training across activities and targeted selection of particular operations. Each segment contains its skill-type label, subtask instruction, three RGB streams from the head and both wrist cameras, and temporally aligned robot states and actions. We retain each segment's source activity and episode, frame boundaries, and object identifiers. These records enable benchmark construction to locate the operation's starting state and identify the corresponding simulator objects (Section 2.3). Appendix E presents recorded examples with their subtask instructions.

2.3 Benchmark Construction

The curated segments provide recorded demonstrations for training, but evaluating a policy on these operations requires running the policy in simulation. For each selected operation, we therefore restore a reference starting state from its source demonstration so that the policy can begin without executing preceding task stages. We also define a local goal that specifies when the operation is complete. Together with the segment's curated subtask instruction, the restored state and local goal form a test instance:

$$e = (s_0, \ell, g), \qquad g : \mathcal{S} \to \{0, 1\}, \tag{2}$$

where $\mathcal{S}$ is the simulator state space, s_0 is the initial robot and scene state, and ℓ is the subtask instruction. The goal function g returns one when the required object states and relations hold, and zero otherwise. To assess sensitivity to starting-state differences that may arise during sequential execution, we further construct variants of s_0 that perturb the robot's base position or joint configuration while keeping ℓ and g fixed.

Initial state restoration (s_0). The reference starting state must preserve the effects of earlier task stages on the robot and scene. For example, a Place operation may begin with the target object already in the robot's gripper. We locate this starting point using the segment's source episode and start frame. A single recorded frame may not describe the complete scene: the states of sleeping objects are not always recorded, and object additions and removals are stored separately. We therefore start from the episode's saved scene configuration and replay the recorded states and transitions up to the selected frame. Because the saved state may lack the simulator's assisted-grasp constraint, we recover it where needed so that held objects remain attached when physics resumes. We also stabilize the restored state to reduce drift caused by residual velocities from the preceding motion. Appendix C.1 details the restoration procedure.

Local success conditions (g). Restoring the starting state allows an operation to be executed independently, but evaluating its outcome also requires a success condition for that operation. The original task goal describes completion of the entire activity and does not directly specify when each intermediate operation has succeeded. We therefore manually define a local goal g for each evaluated subtask using the original task definition, the curated instruction ℓ, and the source action annotations. We express the required object states and relations as BDDL predicates (Srivastava et al., 2022; Li et al., 2023). The segment's recorded object identifiers bind these predicates to the same simulator objects named in the instruction.

For example, placing a soda can o in a trash can c requires the can to be inside c and no longer touching the robot r:

$$g_{\text{place}}(s) = \text{Inside}(o, c; s) \wedge \neg\, \text{Touching}(o, r; s). \tag{3}$$

The goal must also reflect the scope of the demonstrated operation. For Sweep, the full-task goal requires clearing mud from the entire floor, while an individual segment addresses only one patch. We therefore evaluate removal of the mud particles targeted by that segment, allowing other patches

to remain. Appendix C.2 details the local goal definitions and object bindings. Section 4.1 specifies the temporal success rule and execution budget.

Robot state perturbations. During sequential execution, each skill begins from the robot configuration left by preceding operations. Even when those operations succeed, the resulting configuration may differ from those represented in training and affect the next skill's execution (Gu et al., 2023). To measure sensitivity to this starting-state mismatch, we treat each selected operation start as an *evaluation anchor* and construct three starting-state conditions for it. Original uses the reference state restored from the source demonstration. Base Perturbation translates the robot's base while preserving its orientation and the arm, torso, and gripper configurations. Joint Perturbation changes selected arm or torso joint positions while preserving the base pose and gripper configuration. The subtask instruction ℓ, target identities, and local goal g remain fixed across conditions.

Each perturbed state must preserve the preconditions for beginning the operation; a Place operation, for example, must still begin with the intended object in hand. We therefore verify the intended configuration changes and reject variants that introduce unintended collisions or disrupt the required object relations. Appendix C.3 describes these checks and the realized perturbation magnitudes.

Instance validation. Before accepting a test instance, we load it through the evaluation pipeline to verify the restored state, object bindings, and manipulation preconditions. We also check that the local goal is initially unsatisfied, $g(s_0) = 0$, and manually inspect the rendered starting scene. Invalid instances are corrected or replaced and validated again. Appendix C.5 details these checks.

Skill-specific tests. Beyond robot-state variations, the same construction procedure supports targeted tests of how scene factors affect a chosen skill. We demonstrate this with Pick Up by varying target position, target color, and the presence of a nearby distractor, one factor at a time. The robot's starting configuration, target identity, subtask instruction, and local goal remain fixed across these variants. For position changes, associated objects or contents are moved with the target as needed to preserve their spatial relations. Appendix C.4 describes the construction of these variants.

To extend these tests to another skill, researchers can select source demonstrations and a factor of interest, specify the object bindings and local goal, and reuse the restoration and validation procedures above. The shared subtask definitions link each test instance to corresponding training segments, supporting targeted data selection and fine-tuning where evaluation reveals weaknesses.

3 Policy Composition for Long-Horizon Tasks

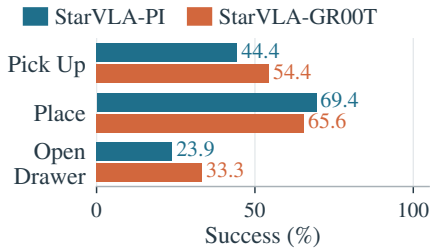
Long-horizon mobile manipulation requires navigation, object manipulation, and coordination across successive task stages. ManiUnit isolates the manipulation stages, providing segments with explicit subtask instructions for training and independent tests for evaluation. A skill policy trained on ManiUnit can therefore focus on executing specified operations, but complete tasks also require navigation between operations and decisions about which one comes next. A task policy trained on complete demonstrations provides control across these stages under a task-level instruction. This suggests a simple composition: retain the task policy as the default controller and delegate selected manipulation stages to the skill policy.

This composition has two requirements. First, the skill policy should perform the delegated operations more reliably than the task policy, a premise supported by the comparison in Section 4.3. Second, the skill policy must be invoked at the appropriate stage and receive the corresponding subtask instruction. A planner provides this coordination, using robot-state and visual histories to decide when to delegate control and which instruction to supply.

This division separates manipulation learning from task-level control and coordination, giving each component a distinct target for improvement. ManiUnit supports focused training and diagnosis of manipulation skills. Future work can improve navigation through task-policy training and coordination through planner design, including task-progress estimation, subtask selection, and policy switching. Complete-task evaluation can then assess whether improvements in these components lead to better overall performance. Appendix D.3 provides implementation details, and Section 4.4 evaluates the combined system on complete tasks.

Model	Original	Base	Joint	Overall
StarVLA-PI	60.1%	51.2%	26.5%	45.9%
StarVLA-GR00T	63.5%	51.3%	28.2%	47.6%

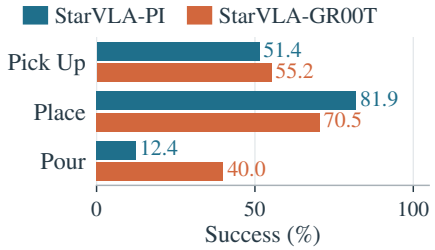
(a) Full: baseline results



(b) Full: skill comparison

Model	Original	Base	Joint	Overall
StarVLA-PI	70.1%	67.3%	26.8%	54.7%
StarVLA-GR00T	77.8%	64.0%	29.6%	57.1%
StarVLA-Cosmos2	63.2%	59.0%	13.8%	45.3%
$\pi_{0.5}$	80.0%	65.8%	53.0%	66.3%

(c) Lite: baseline results



(d) Lite: skill comparison

Figure 3: **Baseline performance on ManiUnit.** (a) and (c) show success rates averaged equally across skill types under each robot starting-state condition; Overall averages the three conditions. (b) and (d) show the performance of PI and GR00T on selected skills, averaged across conditions.

4 Experiments

We evaluate representative VLA policies on ManiUnit to examine per-skill performance and sensitivity to robot starting states and scene conditions (Section 4.2). We then compare skill-level and task-level training on Turning On Radio and Picking Up Trash (Section 4.3) and assess whether the resulting skills improve complete-task execution through policy composition (Section 4.4).

4.1 Experimental Setup

Datasets and policies. We evaluate StarVLA-PI and StarVLA-GR00T (StarVLA Community, 2026) on ManiUnit-Full and compare them with StarVLA-Cosmos2 and $\pi_{0.5}$ on ManiUnit-Lite. For each dataset, we pool segments across skill types and train each model for five epochs. StarVLA-PI and StarVLA-GR00T use pretrained Qwen3.5-2B backbones (Qwen Team, 2026). StarVLA-Cosmos2 uses a pretrained Cosmos-Predict2 backbone. All three use randomly initialized action heads. We fine-tune $\pi_{0.5}$ using RLinf (Yu et al., 2026), starting from policy weights pretrained on diverse multi-robot data (Black et al., 2025a). Appendix D.1 provides the training configurations.

Skill evaluation. We evaluate 1,260 test instances on Full and 186 on Lite, covering 21 and 7 skill types, respectively. Both suites include three starting-state conditions. Original tests whether the policy can execute each skill from a starting state restored from a successful training demonstration. During policy rollouts, even successful preceding skills may leave the robot in a configuration different from the demonstrated one. Base and Joint Perturbation therefore test starting-state sensitivity by perturbing these same reference states. A trial succeeds if the local goal holds for ten consecutive frames within three times the subtask's mean demonstration duration. We run each instance three times on Full and five times on Lite. For each condition, we average success over repeated trials and test instances within each skill type, then give all skill types equal weight. Overall averages the three condition scores. Appendix D.2 reports per-skill results and aggregation details.

4.2 Skill-Level Evaluation

Figure 3(a) and (c) summarize policy performance on Full and Lite. On Full, StarVLA-PI and StarVLA-GR00T achieve similar overall success rates of 45.9% and 47.6%, respectively. On Lite,

$\pi_{0.5}$ achieves the highest overall success rate of 66.3%, followed by StarVLA-GR00T (57.1%), StarVLA-PI (54.7%), and StarVLA-Cosmos2 (45.3%). This advantage may partly reflect differences in pretraining: $\pi_{0.5}$ starts from policy weights pretrained on diverse multi-robot data (Black et al., 2025a), whereas the StarVLA policies use randomly initialized action heads.

Per-skill performance. As shown in Figure 3(b) and (d), results averaged across the three starting-state conditions reveal different skill strengths among policies with similar overall scores. On Full, StarVLA-GR00T achieves higher success on Pick Up (54.4% vs. 44.4%) and Open Drawer, while StarVLA-PI performs better on Place. On Lite, StarVLA-PI again achieves higher success on Place (81.9% vs. 70.5%), but StarVLA-GR00T performs better on Pour (40.0% vs. 12.4%). These profiles identify different weaknesses in each policy and can guide targeted data selection and training.

Robot starting states. As shown in Figure 3(a) and (c), both robot-state perturbations reduce overall success rates for all evaluated policies. On Full, Base Perturbation reduces success relative to Original by 14.8% for StarVLA-PI and 19.3% for StarVLA-GR00T. Under Joint Perturbation, these relative reductions increase to 55.9% and 55.6%, respectively. All four policies on Lite also perform worse under both perturbations, with larger drops under Joint Perturbation. Under this condition, $\pi_{0.5}$ retains approximately two-thirds of its Original success rate, compared with roughly 22–38% for the other three policies. These results identify the robot's starting configuration as a major factor in skill execution. The smaller drop of $\pi_{0.5}$, which is pretrained on diverse multi-robot data, suggests that broader coverage of starting states during data collection or augmentation may improve robustness.

Object and scene conditions. Holding the robot's starting configuration fixed, we examine sensitivity to object and scene changes using Pick Up as a case study. We vary target position, target color, and distractor presence one at a time, while keeping the target identity, instruction, and local goal unchanged. Each of the four Lite policies is evaluated on seven test instances per condition, with five repeats per instance.

Table 1: Pick Up success rates for Original and three scene variations: target position (Pos.), target color, and distractor presence (Dist.).

Model	Original	Pos.	Color	Dist.
StarVLA-PI	68.6%	40.0%	60.0%	62.9%
StarVLA-GR00T	71.4%	20.0%	65.7%	62.9%
StarVLA-Cosmos2	60.0%	22.9%	54.3%	60.0%
$\pi_{0.5}$	91.4%	74.3%	80.0%	80.0%

As shown in Table 1, target-position changes produce the largest performance drop for all four policies among the tested variations, with relative reductions of approximately 19% to 72% from their respective Original scores. Color changes and distractors have smaller effects, reducing success by at most 11.4 percentage points.

4.3 Skill-Level vs. Task-Level Learning

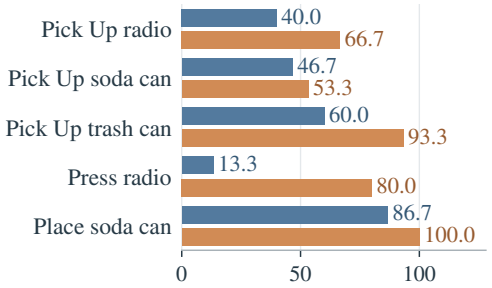
We compare skill-level and task-level training on Turning On Radio and Picking Up Trash to assess their effects on local manipulation performance. The task policy is trained on complete demonstrations with task-level instructions. The skill policy is trained jointly on the corresponding ManiUnit segments across five object-specific subtasks, using their curated subtask instructions. Both policies are initialized from the same pretrained $\pi_{0.5}$ weights and trained for five epochs. We evaluate both policies on the same 75 test instances, with five instances per subtask under each of the three starting-state conditions. Overall success averages the five subtasks and three conditions equally.

As shown in Figure 4(a), the skill policy achieves 78.7% overall success, compared with 49.3% for the task policy. The advantage holds under all three starting-state conditions and is largest under Joint Perturbation, where the skill policy achieves 76.0% success versus 32.0% for the task policy. Relative to Original, success drops by 8.0 percentage points for the skill policy and 28.0 points for the task policy, indicating greater robustness to the tested joint variations.

Figure 4(b) shows higher success for the skill policy on all five subtasks when averaged across the three starting-state conditions. The largest gain is on Press radio, from 13.3% to 80.0%. Success on Pick Up trash can also increases from 60.0% to 93.3%.

Policy	Original	Base	Joint	Overall
Task policy	60.0%	56.0%	32.0%	49.3%
Skill policy	84.0%	76.0%	76.0%	78.7%

(a) Starting conditions



(b) Individual subtasks

Figure 4: **Skill-level vs. task-level learning.** (a) shows overall success rates and success rates under each robot starting-state condition for the task and skill policies. (b) shows their success rates on the five subtasks, averaged across starting-state conditions.

4.4 Long-Horizon Task Execution

We next test whether these local manipulation gains translate into better complete-task performance. Using the policies evaluated in Section 4.3, we compare the task policy alone with the combined system described in Section 3. Both systems are evaluated on Turning On Radio and Picking Up Trash, using the same ten episodes per activity and five repeats per episode. We report full-task success and mean Q-score. Q-score measures the fraction of task-goal conditions satisfied at the end of each trial (Li et al., 2023). Appendix D.3 provides implementation and evaluation details.

Table 2: **Long-horizon task execution.** Full-task success rates and mean Q-scores for the task policy alone and the combined system, which uses a planner to coordinate the task and skill policies. Overall averages the results across Turning On Radio and Picking Up Trash.

	Task policy		Combined system	
Activity	Full-task success	Mean Q-score	Full-task success	Mean Q-score
Radio	4.0%	4.0%	24.0%	24.0%
Trash	4.0%	18.0%	12.0%	26.0%
Overall	4.0%	11.0%	18.0%	25.0%

As shown in Table 2, the combined system increases overall full-task success from 4.0% to 18.0% and mean Q-score from 11.0% to 25.0%. Full-task success increases from 4.0% to 24.0% on Turning On Radio and from 4.0% to 12.0% on Picking Up Trash. Figure 5 in Appendix D.3 illustrates successful rollouts of both activities. ManiUnit-trained skills can thus support complete-task execution through policy composition.

5 Conclusion

We introduced ManiUnit, a manipulation skill dataset and benchmark that treats individual operations within long-horizon tasks as units of training and evaluation. Its dataset reorganizes BEHAVIOR-1K demonstrations into manipulation segments with explicit subtask instructions, while its benchmark pairs restored simulator states with local goals to enable independent evaluation under controlled starting-state variations. Evaluations on ManiUnit support skill-level diagnosis and reveal policy sensitivity to robot starting states. On two activities, skill-level training improves local manipulation success, and policy composition improves full-task success.

Limitations and future work. Despite these gains, reliable skill execution remains challenging, particularly under perturbed starting conditions. Complete-task performance also depends on coordination decisions that determine which subtask instructions to provide and when to switch policies. Future work can use ManiUnit's data and diagnostic evaluations to improve individual skills. In parallel, planners can be developed to better track task progress, select subtasks, and determine appropriate switching times. These directions allow manipulation learning and task coordination to be studied separately, with their combined effectiveness assessed through complete-task execution.

AI USE STATEMENT

We used generative AI tools to assist with manuscript editing and translation, with a focus on grammar, clarity, and presentation. The authors reviewed the AI-assisted content and take responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

The main text and appendix describe the dataset and benchmark construction, training settings, and evaluation protocols. We will release the dataset, code, and trained model checkpoints upon acceptance.

REFERENCES

Junjie Bai et al. Openpi Comet: Competition solution for 2025 BEHAVIOR challenge. *arXiv preprint arXiv:2512.10071*, 2025.

Kevin Black et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b.

Nils Blank et al. Scaling robot policy learning via zero-shot labeling with foundation models. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4158–4187. PMLR, 2025.

Tianxing Chen et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Forty-third International Conference on Machine Learning*, 2026.

Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. SkillMimicGen: Automated demonstration generation for efficient skill learning and deployment. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2750–2790. PMLR, 2025.

Gemini Robotics Team et al. Gemini Robotics: Bringing AI into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill mobile manipulation for object rearrangement. In *The Eleventh International Conference on Learning Representations*, 2023.

brian ichter et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025.

Ilia Larchenko, Gleb Zarin, and Akash Karnatak. Task adaptation of vision-language-action model: 1st place solution for the 2025 BEHAVIOR challenge. *arXiv preprint arXiv:2512.06951*, 2025.

Chengshu Li et al. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 80–93. PMLR, 2023.

Bo Liu et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023.

Mengyuan Liu et al. Eval-Actions: Fine-grained execution quality evaluation for robotic manipulation. *arXiv preprint arXiv:2601.18723*, 2026.

Ajay Mandlekar et al. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

NVIDIA et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Abby O'Neill et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The Colosseum: A benchmark for evaluating generalization for robotic manipulation. In *Proceedings of Robotics: Science and Systems*, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Lucy Xiaoyang Shi et al. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933. PMLR, 2025.

Sanjana Srivastava et al. BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 477–490. PMLR, 2022.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Yi Ru Wang et al. RoboEval: Where robotic manipulation meets structured and scalable evaluation. *arXiv preprint arXiv:2507.00435*, 2025.

Shihan Wu et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Shuai Yang et al. Vision-language-action instruction tuning: From understanding to manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 152896–152943, 2026.

Chao Yu et al. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 829–846. USENIX Association, 2026.

Borong Zhang et al. VLA-Arena: An open-source framework for benchmarking vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026.

Brianna Zitkovich et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

Table 3: Comparison of manipulation datasets and benchmarks in terms of mobility, training annotations, and skill evaluation.

Dataset	Mobile	Training		Evaluation	
		Skill labels	Language instr.	Independent skill eval.	Robot-state perturb.
LIBERO (Liu et al., 2023)	×	×	✓	×	×
CALVIN (Mees et al., 2022)	×	✓	✓	✓	×
RoboTwin 2.0 (Chen et al., 2026)	×	×	✓	×	×
RoboCasa365 (Nasiriany et al., 2026)	✓	✓	✓	×	✓
BEHAVIOR-1K (Li et al., 2023)	✓	✓	✓	×	×
ManiUnit (ours)	✓	✓	✓	✓	✓

✓: provided; ×: not provided in the compared dataset or evaluation protocol. Skill labels are temporally localized annotations within demonstrations. Independent skill evaluation combines intermediate-state restoration with local success conditions for the corresponding operation. Robot-state perturbation varies the initial robot base pose or joint configuration.

A Related Work

Robot Demonstrations and Skill-Level Data Organization. Learning manipulation policies benefits from both broad demonstration coverage and explicit supervision for the operations within each demonstration. Open X-Embodiment (O'Neill et al., 2024) and DROID (Khazatsky et al., 2024) expand coverage through aggregation across robot platforms and distributed data collection, respectively. Beyond collection, MimicGen (Mandlekar et al., 2023) adapts demonstrations to new scene configurations, while SkillMimicGen (Garrett et al., 2025) composes adapted skill segments to generate new demonstrations. Finer-grained annotations make the operations within these demonstrations explicit: NILS (Blank et al., 2025) uses foundation models to segment demonstrations and assign language instructions, and RoboCOIN (Wu et al., 2025) provides annotations at the trajectory, segment, and frame levels. ManiUnit constructs manipulation segments from BEHAVIOR-1K demonstrations (Li et al., 2023) through action extraction and merging, boundary refinement, and manual instruction curation. The resulting corpus groups segments across source activities by skill type and subtask, allowing researchers to pool data for training a shared skill policy or select data for a particular skill or subtask.

Skill Evaluation and Execution Robustness. Diagnosing long-horizon task performance requires understanding which operations fail and under what conditions. LIBERO (Liu et al., 2023) evaluates language-conditioned policies on tabletop manipulation tasks, while RoboTwin 2.0 (Chen et al., 2026) evaluates bimanual manipulation under domain randomization. RoboCasa365 (Nasiriany et al., 2026) includes atomic and composite kitchen tasks, and BEHAVIOR-1K (Li et al., 2023) provides long-horizon household activities in realistic simulation. When a task combines multiple operations, overall success can obscure differences between skills, and early failures can leave later operations untested. CALVIN (Mees et al., 2022) supports both independent task evaluation from restored demonstration states and sequential execution. RoboEval (Wang et al., 2025) and EvalActions (Liu et al., 2026) provide complementary diagnostics of staged task progress and execution quality, respectively. Robustness benchmarks examine changing execution conditions through environmental perturbations in Colosseum (Pumacay et al., 2024) and language and visual variations in VLA-Arena (Zhang et al., 2026).

ManiUnit combines independent skill evaluation with controlled variations in the robot's configuration at the start of an operation. Restored intermediate states and local success conditions allow manipulation operations from long-horizon household demonstrations to be evaluated without executing preceding stages. Perturbations of the robot's base position and joint configuration simulate variation in the starting configurations inherited from preceding operations and measure its effect on skill performance. Table 3 summarizes these capabilities.

VLA Policies and Hierarchical Control. Recent VLA models such as RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2025), and π_0 (Black et al., 2025b) map language instructions and visual observations to robot actions. Under a single task instruction, however, a policy must infer the current operation, which can be ambiguous when different stages share similar visual context. Hierarchical approaches provide explicit intermediate guidance. SayCan (ichter et al., 2023) selects skills by combining language-model estimates of usefulness with value-function estimates of execution feasibility, while Hi Robot (Shi et al., 2025) supplies intermediate instructions through a high-level reasoner. Other approaches integrate this guidance within the policy: $\pi_{0.5}$ (Black et al., 2025a) combines subtask prediction with conditioned action generation, and InstructVLA (Yang et al., 2026) jointly trains embodied reasoning and action generation through instruction tuning. ManiUnit supports instruction-conditioned manipulation by providing segments paired with manually curated subtask instructions for training a shared skill policy. Section 3 describes this coordination: a task policy handles navigation and transitions, while a planner decides when to invoke the skill policy and which subtask instruction to supply.

B Dataset Construction and Curation

This section provides further details on the dataset construction described in Section 2.2 of the main text, including skill segment extraction and merging rules, boundary-refinement parameters, and data consistency checks.

B.1 Skill Inventory and Coverage

Table 4 summarizes the source activities, subtasks, and segments associated with each skill type. Subtask counts reflect the diversity of operations, while segment counts capture the number of recorded executions. The mean segment duration is 17.9 seconds, computed from each segment's frame count and recorded frame rate. Pick Up and Place appear in nearly all source activities and together account for 73.8% of the segments, whereas task-specific operations such as Ignite and Wipe each come from a single activity. Of the 21 skill types, 13 appear in multiple source activities, and 37 exact subtask instructions are shared across activities. Organizing segments by skill type and subtask therefore supports targeted data selection and lets researchers pool demonstrations of the same operation across task contexts. For evaluation, the Full benchmark assigns 20 anchors to each of the 21 skill types, giving every type equal representation regardless of its frequency in the training data (Section 4.1).

We construct ManiUnit-Lite by selecting the eight source activities with the highest task success rates in the 2025 BEHAVIOR Challenge. These activities are Make Microwave Popcorn, Cook Hot Dogs, Turning On Radio, Picking Up Trash, Moving Boxes to Storage, Cook Bacon, Bringing Water, and Wash a Baseball Cap. The curated manipulation segments from these activities form a subset of 10,588 segments across seven skill types.

For the corresponding benchmark, we retain the test instances from ManiUnit-Full whose evaluation anchors originate from these eight activities. This selects 186 of the 1,260 Full test instances, covering 62 anchors under Original, Base Perturbation, and Joint Perturbation. Together, these dataset and benchmark subsets reduce the computational cost of repeated experiments.

B.2 Skill Segment Extraction and Merging

ManiUnit builds on the skill annotations provided with the BEHAVIOR-1K demonstrations. Each source annotation records an action label, the objects involved, and the action's start and end frames. We use these annotations to extract manipulation segments from full-task trajectories. For the 50 source activities, we manually define 312 extraction rules and 48 merging rules.

Table 4: Skill-type inventory of ManiUnit-Full.

Skill type	Source activities	Subtasks	Segments
Pick Up	50	153	52,251
Place	45	184	49,574
Chop	6	18	7,624
Open Door	24	9	7,476
Close Door	20	7	6,048
Pour	7	14	2,993
Sweep	2	2	2,681
Turn On Switch	11	7	2,596
Close Lid	5	4	1,197
Turn Off Switch	3	3	1,000
Spray	2	2	804
Open Lid	3	3	794
Insert	1	2	600
Release	2	2	462
Close Drawer	1	1	400
Open Drawer	1	1	400
Attach	1	1	200
Ignite	1	1	200
Press	1	1	200
Wipe	1	1	200
Hang	1	1	199
Total (21 skill types)	**50**	**417**	**137,899**

Extraction rules specify which annotated actions to retain from each activity. The same action label may appear several times within an episode for different objects. To identify the intended operations, the rules use object information and execution order where needed. Action labels are matched exactly, distinguishing annotations such as *place on* and *place on next to*. For example, an episode may contain several pickup actions involving different objects. The rules specify which pickups to retain, and each retained pickup becomes a separate segment. The recorded start and end frames provide the initial boundaries of each extracted interval. Merging rules combine supporting actions with the main operations they serve when the source annotations record them separately. Each rule specifies the actions to combine and one of two adjacency conditions.

The first condition requires consecutive annotations in the source demonstration. Pushing a board game to the edge of a bed and picking it up, for example, are annotated as two consecutive actions, although the push prepares the object for grasping. The corresponding rule merges these two intervals into one Pick Up segment. The second condition requires consecutive entries in the sequence of extracted manipulation operations, allowing navigation between a supporting action and the main operation. Before hanging a picture, for example, the robot may transfer it between its hands, move to the hanging location, and then hang it. The source annotations can therefore record navigation between the handover and hanging actions. Because navigation is not extracted as a separate manipulation operation, the handover and hanging intervals become consecutive in the extracted sequence when no other selected operation lies between them. The corresponding rule combines them into one Hang segment, beginning with the handover and ending with the hanging action, while retaining the navigation recorded in between.

B.3 Temporal Boundary Refinement

Boundary refinement locates the manipulation phase within each extracted interval using the arm-posture departure defined in Section 2.2. We monitor the first 3 joints of each arm and smooth their positions with a length-preserving moving average using zero padding. The fixed reference posture

is the mean of the smoothed joint positions over 500 lowered-arm frames from source activity 0, episode 1. A consecutive run qualifies when its departure from this reference meets the threshold for at least the minimum duration. We then retain the continuous interval from the start of the first qualifying run to the end of the last, including intervening pauses. Table 5 lists the defaults.

Table 5: Default parameters for temporal boundary refinement.

Parameter	Value
Monitored arm joints	First 3 per arm
Reference window	500 frames
Smoothing window	5 frames
Minimum joint departure	15°
Minimum qualifying duration	5 frames
Gripper position-change threshold	$0.002\,\mathrm{m}$
Gripper protection margin	$5\,\mathrm{s}$ per boundary
Minimum retained fraction	0.2

Two safeguards limit excessive trimming. To preserve motions around grasp and release, we detect gripper events from the maximum absolute change in finger-joint positions between consecutive frames, using the unsmoothed positions of both grippers. The candidate interval is extended as needed to include a margin before the first event and after the last, with both boundaries constrained to the original interval. If the retained fraction falls below the specified threshold, we restore the full original interval. Intervals with no qualifying arm run are retained for manual inspection. This procedure shortens 13,664 segments in the released corpus. We also inspect selected segment videos to check that the refined boundaries preserve the intended operation.

B.4 Quality Control and Data Consistency

Each exported segment records its source activity and episode, together with its frame boundaries before and after refinement. We clip the action annotations to the retained interval and reindex them relative to the segment, preserving their source indices for traceability.

We check the correspondence between segment identifiers and instruction records across the full corpus and find no mismatches. For sampled segments, we also compare video frame counts, the lengths of robot-state and action sequences, and their source-frame indices. These checks confirm that the inspected videos, states, and actions cover the same interval of the source demonstration.

C Benchmark Construction and Validation

This section expands on the benchmark construction in Section 2.3. We describe how source simulation records are used to restore each operation's starting state, how local goals are defined, and how test variants are constructed and checked.

C.1 Initial State Restoration

Evaluating a manipulation skill independently requires restoring the robot and scene to the conditions at the operation's start in the source demonstration. We identify this point using the segment's source episode and frame index. The restoration procedure addresses three requirements: recovering the scene history leading up to the operation, restoring any existing grasp, and reducing residual motion before policy execution.

Replaying the scene history. A single recorded frame may not contain all the information needed to reconstruct the scene. The recording pipeline can omit the states of sleeping objects, relying on their previously recorded states during playback. Object and particle-system additions and removals are stored separately in a transition history. We therefore start from the episode's saved scene

configuration and replay the recorded states and transitions up to the selected frame, collecting the information needed to restore the operation's starting conditions.

Recovering assisted-grasp constraints. Some operations begin with the robot already holding an object. The simulator maintains this attachment through an *assisted-grasp constraint*, which may be absent from the saved state even when the robot and object poses are correct. Without this constraint, the object may slip when physics resumes. We therefore check the grasp relation after restoration.

Stabilizing residual motion. A state captured during a demonstration may retain velocities from the preceding motion, causing the robot or scene objects to drift after restoration. We reduce this residual motion before policy execution. By default, the action-based generator uses 25 settling steps followed by 60 hold-action steps. During the hold phase, the controllers maintain the robot configuration, command zero base velocity, and preserve the recorded gripper commands. Velocity clearing further reduces drift. Finally, we check that the restored state satisfies the operation's preconditions and that the local goal is not already satisfied before evaluation begins.

C.2 Local Goal Definitions and Object Binding

The source BDDL goals describe full-activity outcomes and do not directly specify when individual intermediate operations are complete. We therefore define a local goal for each evaluated subtask using the original task definition, the subtask instruction, and the source action annotations. Each goal specifies the required object states and relations through BDDL predicates bound to the corresponding simulator instances. For example, placing a soda can in a trash can requires the designated can to be inside the designated container and no longer touching the robot, as in Equation 3.

Two skill types require additional handling because the operation either replaces objects or affects only a local region.

Chop. Cutting can replace the source object with pieces that have new simulator identities: a whole pepper, for example, becomes pepper slices once the cut is complete. The local goal therefore tracks both the source object and the required pieces. When the source object disappears, completion requires the corresponding pieces to be present and the remaining goal conditions to hold. Its disappearance alone does not establish that cutting has succeeded.

Sweep. The full-task goal requires the entire floor to be clean, but an individual sweep segment targets one patch. We identify the particles removed by the demonstrated operation and evaluate only that set. To determine which particles the segment actually removes, we compare source-action replay with a reference rollout in which the robot holds still for the same duration: a particle belongs to the target set if it disappears during the sweep but remains present when the robot does nothing. We store the selected particle identities with the test instance and use the same target set across all policy evaluations, so success requires removing those specific particles rather than clearing mud elsewhere.

C.3 Robot State Perturbations

During long-horizon task execution, preceding skills may leave the robot in a configuration that differs from the demonstrated starting state of the next operation. To measure sensitivity to these differences, we construct three starting-state conditions for each anchor while keeping the subtask instruction, target identities, and local goal fixed. Original uses the reference state restored from the source demonstration. Base Perturbation translates the robot's base while preserving its orientation and the arm, torso, and gripper configurations. Joint Perturbation changes selected arm or torso joint positions while preserving the base pose and gripper configuration.

A perturbed state must preserve the conditions needed to begin the operation. For a Place anchor, for example, the robot must still be holding the intended object after the perturbation. We therefore check each candidate for unintended collisions, disrupted grasp relations, and displacement of contents relative to held containers. We also verify that components intended to remain fixed have not changed beyond the allowed tolerances. Candidates that fail these checks are rejected.

In Full, base translations range from 8.0 to 11.8 cm, with a median of 9.9 cm. For Joint Perturbation, we measure the largest absolute joint change in each anchor; the median across anchors is 0.916 rad.

C.4 Skill-Specific Test Extensions

The benchmark construction procedure also supports targeted tests of how scene factors affect individual skills. We demonstrate this with Pick Up, varying target position, target color, and distractor presence separately. Each variant keeps the robot's starting configuration, target identity, subtask instruction, and local goal fixed, allowing comparison with Original to assess sensitivity to the selected factor.

Color variants change the target's color through an object-specific texture override while preserving its geometry and pose. Distractor variants add a nearby object from a different category. The distractor receives a separate simulator identity, so the instruction and local goal continue to refer to the intended target. Both variants use the Original state as their starting point.

Position variants require additional care when the target contains or supports other objects. Moving a popcorn bag, for example, also requires moving its popcorn particles to preserve their arrangement within the bag. The generator applies the same rigid transformation to the container and its associated particles, using their states from the same source frame. This preserves their spatial relations at the modified starting state. Each variant records its construction settings.

To extend these tests to another skill, researchers select source demonstrations and a scene factor of interest, specify the corresponding object bindings and local goal, and reuse the restoration and validation procedures described in this section. The shared subtask definitions also link test instances to matching training segments for targeted data selection and fine-tuning.

C.5 Test Records and Validation

Each exported test instance records its source episode and frame indices, subtask instruction, local goal, and construction settings, linking it back to the source operation and allowing it to be regenerated. Before final release, we load every instance through the evaluation pipeline for a final validation pass. This checks state restoration, object bindings, and manipulation preconditions, and verifies that the local goal is not already satisfied at the starting state. We also manually inspect the rendered starting scene of every test instance to catch restoration errors or unintended scene changes that automated checks may miss. Instances that fail either check are corrected or replaced.

D Experimental Details and Additional Results

This section provides the training configurations, complete per-skill results, and details of the supplementary studies reported in the main text.

D.1 Training Configurations

Benchmark policies are trained for 5 epochs on their corresponding ManiUnit-Full or ManiUnit-Lite training sets, with one policy per model trained jointly across all skill types in the set. Full evaluates StarVLA-PI and StarVLA-GR00T; Lite additionally evaluates StarVLA-Cosmos2 and $\pi_{0.5}$. Each skill prompt specifies the skill type and full subtask instruction, for example: *Task Skill Type: pick up. Task Prompt: pick up the radio from the coffee table.*

StarVLA. We largely follow the training configurations provided by StarVLA (StarVLA Community, 2026). For StarVLA-PI and StarVLA-GR00T, we initialize the vision-language backbone from Qwen3.5-2B (Qwen Team, 2026) and randomly initialize the action heads. Policies predict chunks of 30 actions, each with 23 dimensions, and use a global batch size of 512. Training uses AdamW with $\beta = (0.9, 0.95)$, 5,000 warm-up steps, gradient clipping at 1.0, and a cosine learning-rate schedule. Learning rates are 10^{-5} for the backbone and 10^{-4} for the action head.

StarVLA-Cosmos2 loads its text encoder, visual VAE, and world-model transformer from the Cosmos-Predict2-2B-Video2World checkpoint. The text encoder and VAE remain frozen. We jointly train the transformer and a randomly initialized GR00T-style flow-matching action head.

We adapt the camera inputs for Cosmos2 and the joint-control interface for StarVLA. The upstream Cosmos2 wrapper interprets an image list as a temporal sequence, whereas BEHAVIOR provides 3 simultaneous camera views. We therefore combine a 224×224 head image with two 112×112 wrist images placed side by side below it, forming a 336×224 spatial mosaic (height $\times$ width). This layout presents all 3 views as a single observation. Resizing and padding preserve their aspect ratios, and training and inference use the same layout.

For joint control, we express the 14 arm-joint targets and the first 3 torso-joint targets as offsets from the joint positions observed at the start of each predicted chunk. The reference positions remain fixed throughout the chunk. Base velocities, the fourth torso joint, and gripper commands retain their native representation. At inference, the adapter adds the reference positions back after denormalization to recover absolute joint targets.

$\pi_{0.5}$. We use the RLinf implementation (Yu et al., 2026), with training settings informed by Openpi Comet (Bai et al., 2025). The policy uses absolute joint targets and a global batch size of 256. Training uses AdamW with a learning rate of 2.5×10^{-5}, $\beta = (0.9, 0.95)$, 1,000 warm-up steps, and a cosine schedule. Sampling follows the recorded frame distribution, and the random seed is 42.

D.2 Per-Skill Success Rates

In long-horizon task execution, a single task-level score cannot identify which specific skills are unreliable or how sensitive each is to starting-state variations. An early failure may also prevent later skills from being attempted at all, leaving their performance unknown. ManiUnit addresses both limitations by evaluating each skill independently from a restored starting state with its own local goal, and by testing each skill under three starting-state conditions to measure how robot configuration at entry affects execution.

Tables 6 and 7 report per-skill success rates under Original, Base Perturbation, and Joint Perturbation on ManiUnit-Full and ManiUnit-Lite, respectively. For each anchor and starting-state condition, we average success over repeated trials. We then average across anchors within each skill type and give skill types equal weight when computing the score for each condition. Overall success averages the three condition scores.

D.3 Planner and Complete-Task Evaluation

This section provides implementation and evaluation details for the policy composition scheme in Section 3. The combined system uses the task and skill policies compared in Section 4.3. Both policies are initialized from the same pretrained $\pi_{0.5}$ weights and trained for five epochs on their respective datasets. The task policy uses complete demonstrations of Turning On Radio and Picking Up Trash, while the skill policy uses ManiUnit segments from these activities with skill-level instructions. During execution, the task policy provides default control for navigation and transitions between stages. The planner decides when to invoke the skill policy and which subtask instruction to supply.

Planner. The planner first distinguishes navigation from manipulation so that the skill policy is invoked during the stages it was trained to execute. Robot-state histories capture arm, torso, and gripper configurations and their changes over time, providing cues about the current execution stage. We therefore train a navigation classifier on these histories to estimate whether the robot is navigating or manipulating.

Identifying the specific operation also requires visual information about the scene and objects. We use two complementary estimates to reduce premature switching on an uncertain prediction. A subtask classifier combines robot-state history with head-camera features to predict the skill type. Separately, a video-retrieval module matches recent observations to a database built from annotated training demonstrations of the two activities. Each database entry associates video features with a subtask label and its object-specific instruction. The classifier and retrieval module must agree on the skill type before the planner invokes the skill policy. Both classifiers use separate task-conditioned causal Transformers with 8 layers, hidden width 128, and 4 attention heads. Each robot-state observation is projected to a 128-dimensional token, and each classifier processes up to 32 observations sampled at 1 Hz. The subtask classifier additionally attends to frozen SigLIP

Table 6: Per-skill success rates on ManiUnit-Full (%). Base and Joint denote Base Perturbation and Joint Perturbation.

	StarVLA-PI				StarVLA-GR00T			
Skill	Original	Base	Joint	Mean	Original	Base	Joint	Mean
Attach	21.7%	11.7%	3.3%	12.2%	26.7%	1.7%	0.0%	9.4%
Chop	41.7%	31.7%	46.7%	40.0%	43.3%	31.7%	41.7%	38.9%
Close Door	83.3%	83.3%	50.0%	72.2%	90.0%	83.3%	46.7%	73.3%
Close Drawer	86.7%	95.0%	35.0%	72.2%	98.3%	98.3%	46.7%	81.1%
Close Lid	73.3%	66.7%	31.7%	57.2%	75.0%	61.7%	35.0%	57.2%
Hang	0.0%	0.0%	1.7%	0.6%	0.0%	1.7%	5.0%	2.2%
Ignite	100.0%	78.3%	8.3%	62.2%	95.0%	75.0%	5.0%	58.3%
Insert	23.3%	8.3%	10.0%	13.9%	26.7%	3.3%	1.7%	10.6%
Open Door	78.3%	75.0%	35.0%	62.8%	81.7%	83.3%	40.0%	68.3%
Open Drawer	36.7%	30.0%	5.0%	23.9%	51.7%	38.3%	10.0%	33.3%
Open Lid	51.7%	13.3%	20.0%	28.3%	45.0%	23.3%	15.0%	27.8%
Pick Up	48.3%	56.7%	28.3%	44.4%	65.0%	61.7%	36.7%	54.4%
Place	91.7%	70.0%	46.7%	69.4%	90.0%	68.3%	38.3%	65.6%
Pour	55.0%	43.3%	10.0%	36.1%	70.0%	53.3%	5.0%	42.8%
Press	45.0%	21.7%	6.7%	24.4%	56.7%	21.7%	1.7%	26.7%
Release	100.0%	100.0%	86.7%	95.6%	100.0%	91.7%	95.0%	95.6%
Spray	90.0%	85.0%	60.0%	78.3%	90.0%	85.0%	71.7%	82.2%
Sweep	85.0%	66.7%	25.0%	58.9%	85.0%	56.7%	33.3%	58.3%
Turn Off Switch	55.0%	56.7%	21.7%	44.4%	46.7%	56.7%	16.7%	40.0%
Turn On Switch	88.3%	80.0%	25.0%	64.4%	86.7%	76.7%	43.3%	68.9%
Wipe	6.7%	1.7%	0.0%	2.8%	10.0%	3.3%	3.3%	5.6%

Table 7: Per-skill success rates on ManiUnit-Lite (%). Base and Joint denote Base Perturbation and Joint Perturbation.

	StarVLA-PI			StarVLA-GR00T		
Skill	Original	Base	Joint	Original	Base	Joint
Close Door	95.0%	97.5%	45.0%	95.0%	90.0%	42.5%
Open Door	80.0%	96.7%	13.3%	93.3%	83.3%	10.0%
Pick Up	68.6%	77.1%	8.6%	71.4%	80.0%	14.3%
Place	91.4%	74.3%	80.0%	85.7%	57.1%	68.6%
Pour	22.9%	5.7%	8.6%	57.1%	25.7%	37.1%
Press	47.0%	31.0%	1.0%	42.0%	29.0%	9.0%
Turn On	85.7%	88.6%	31.4%	100.0%	82.9%	25.7%
	StarVLA-Cosmos2			$\pi_{0.5}$		
Skill	Original	Base	Joint	Original	Base	Joint
Close Door	62.5%	62.5%	22.5%	92.5%	92.5%	75.0%
Open Door	83.3%	80.0%	0.0%	100.0%	100.0%	63.3%
Pick Up	60.0%	54.3%	2.9%	91.4%	60.0%	71.4%
Place	97.1%	97.1%	57.1%	68.6%	51.4%	31.4%
Pour	0.0%	0.0%	0.0%	31.4%	20.0%	17.1%
Press	65.0%	36.0%	0.0%	76.0%	51.0%	27.0%
Turn On	74.3%	82.9%	14.3%	100.0%	85.7%	85.7%

features from the 8 most recent head-camera images. The retrieval module uses a VideoMAEv2 encoder adapted on BEHAVIOR-1K videos to extract features from recent head- and wrist-camera clips.

The retrieval module accepts a subtask instruction when its confidence meets a predefined threshold. The planner activates the skill policy when the navigation classifier identifies manipulation and the subtask classifier predicts the same skill type as the accepted instruction, with both classifiers meeting their respective confidence thresholds. Otherwise, the task policy retains control.

Once a skill is active, the planner uses a lower subtask-confidence threshold to reduce interruptions. If confidence falls below this threshold, a fixed grace period allows it to recover before control returns to the task policy. This tolerance applies only while the navigation decision remains manipulation and the skill type and target instruction remain unchanged. All policy switches occur at action-chunk boundaries.

Evaluation. We evaluate the composed system and the task-policy baseline on Turning On Radio and Picking Up Trash. Each activity has 10 test episodes, each repeated 5 times, yielding 50 trials per activity for each system.

We report success rate and final Q-score. Success rate is the proportion of trials that satisfy the complete-task BDDL goal. Final Q-score measures partial task completion as the fraction of goal conditions satisfied at the end of a trial, using the best-matched goal clause (Li et al., 2023). We compute success rate and mean final Q-score for each activity and give the two activities equal weight in the overall results. Table 2 reports both metrics as percentages. Figure 5 illustrates successful rollouts with their policy selections and subtask-instruction transitions.



Figure 5: Successful rollouts of (a) Turning On Radio and (b) Picking Up Trash. A planner coordinates a task policy for navigation and task-level control with a ManiUnit-trained skill policy for manipulation. Colors indicate the active policy.

E VISUAL CATALOG OF MANIPULATION SKILLS

Figures 6–11 illustrate ManiUnit's 21 manipulation skill types through recorded examples paired with their subtask instructions. We group the skills into three categories according to the type of interaction. *Object handling* includes moving, placing, attaching, and releasing objects, as well as transferring their contents. *Opening and closing* includes interactions with doors, drawers, and lids. *Activation and tool use* includes pressing buttons, operating switches, and igniting materials, as well as chopping, spraying, sweeping, and wiping.

Each row shows five examples of a skill type, together with their full subtask instructions. When all five examples share the same instruction, we display it once below the images.

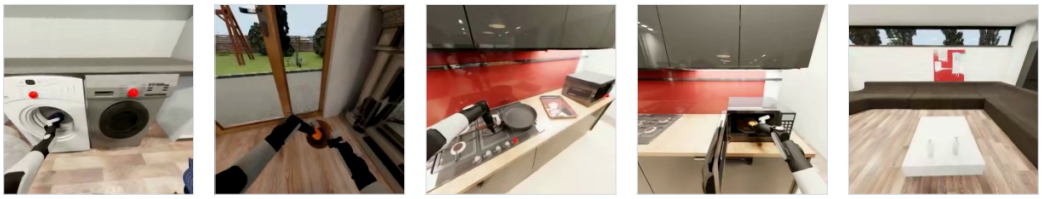
Object handling

Pick Up



Pick up the radio from the coffee table. | Pick up the baseball cap from the countertop. | Pick up the beer bottle from the fridge. | Pick up the frying pan from the burner. | Pick up the can of soda from the floors.

Place



Place the baseball cap in the washer. | Place the can of soda in the trash can. | Place the frying pan on the burner. | Place the hotdog in the microwave. | Place the beer bottle on the coffee table.

Release



Release the broom. | Release the camera tripod. | Release the broom. | Release the camera tripod. | Release the broom.

Figure 6: Object handling: recorded examples with their subtask instructions.

Object handling

Insert



Insert the pen into the pencil case.



Insert the pencil into the pencil case.



Insert the pen into the pencil case.



Insert the pencil into the pencil case.



Insert the pen into the pencil case.

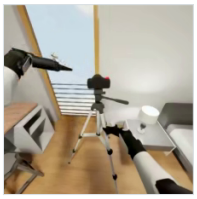
Attach









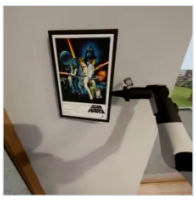


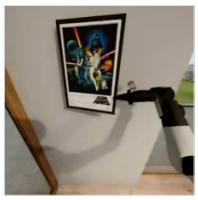
Attach the digital camera to the camera tripod.

Hang











Hang the poster on the wall nail.

Pour



Pour the bacons from the tray into the frying pan.



Pour the diced vidalia onions from the cutting board into the bowl.



Pour the grated cheese from the tupperware onto the pizza dough.



Pour the diced pineapples from the cutting board into the bowl.



Pour the mushrooms from the tupperware onto the chopping board.

Figure 7: Object handling: recorded examples with their subtask instructions.

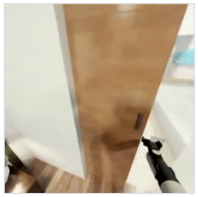
Opening and closing

Open Door


Open the microwave door.


Open the fridge door.

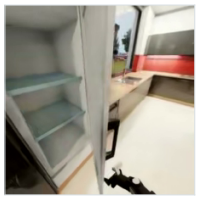

Open the washer door.


Open the sliding door.


Open the oven door.

Close Door


Close the microwave door.


Close the fridge door.


Close the washer door.


Close the oven door.


Close the cabinet door.

Open Drawer


Open the bottom cabinet drawer.

Figure 8: Opening and closing: recorded examples with their subtask instructions.

Opening and closing

Close Drawer



Close the bottom cabinet drawer.

Open Lid



Open the car lid. | Open the toolbox lid. | Open the hinged jar lid. | Open the toolbox lid. | Open the hinged jar lid.

Close Lid



Close the car lid. | Close the toolbox lid. | Close the hinged jar lid. | Close the laptop lid. | Close the car lid.

Figure 9: Opening and closing: recorded examples with their subtask instructions.

Activation and tool use

Press



Press the radio.

Turn On Switch



Turn on the burner switch. | Turn on the microwave switch. | Turn on the washer switch. | Turn on the lighter switch. | Turn on the oven switch.

Turn Off Switch



Turn off the insectifuge atomizer switch. | Turn off the pesticide atomizer switch. | Turn off the lighter switch. | Turn off the insectifuge atomizer switch. | Turn off the lighter switch.

Ignite



Ignite the firewood.

Figure 10: Activation and tool use: recorded examples with their subtask instructions.

Activation and tool use

Chop



Chop the beet. | Chop the bell pepper. | Chop the pineapple. | Chop the steak. | Chop the log.

Spray



Spray the pot plant. | Spray the tree. | Spray the pot plant. | Spray the tree. | Spray the pot plant.

Sweep



Sweep the floors. | Sweep the half log off the chopping block onto the driveway. | Sweep the floors. | Sweep the half log off the chopping block onto the driveway. | Sweep the floors.

Wipe



Wipe the trumpet.

Figure 11: Activation and tool use: recorded examples with their subtask instructions.