HUMANOIDHARNESS: SELF-IMPROVING HUMANOID TASK EXECUTION THROUGH EXPERIENCE-GUIDED AGENTIC POLICY REVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained robot policies provide reusable action capabilities, but completing humanoid tasks requires maintaining situational awareness, adjusting body–target alignment, and adapting execution to the environment. Whole-body motion changes these conditions throughout an interaction, making feedback and recovery essential to coordinating learned capabilities. We introduce HumanoidHarness, a self-improving agentic framework that organizes pretrained controllers through an editable execution policy. The policy specifies controller stages, observation checks, and recovery transitions, and is validated and compiled into a state machine. During execution, the harness bounds each action segment and uses new observations to assess progress and select the next state. Across attempts, a reflection journal and task memory retain execution evidence to guide policy revision and reuse. This design separates immediate execution feedback from experience-guided changes to the policy, while keeping pretrained controller parameters fixed. The same agent logic supports simulation and hardware using egocentric vision, proprioception, and task instructions. Results on HumanoidArena show improved task completion and a generally steady increase in success rate across policy revisions with episode memory, while hardware examples illustrate how coordinated actions can adapt to environmental constraints.

1 INTRODUCTION

Pretrained robot policies, including vision–language–action models (VLAs) and world action models (WAMs), provide learned action capabilities conditioned on visual observations and task instructions (Kim et al., 2024; Black et al., 2024; GigaWorld Team et al., 2026). Completing a humanoid task requires organizing these capabilities as whole-body motion changes visibility, target alignment, reachability, and balance (Wang et al., 2026). Effective execution must therefore connect what a policy can do with when and how it should be used.

The need for a harness arises from a gap between producing plausible actions and reliably achieving a task. A policy may execute a familiar motion yet fail to ground the intended target, account for a changed layout, or recover from an unsuccessful interaction (Zhang et al., 2026). New image inputs alone do not guarantee failure diagnosis or appropriate recovery. Reliable deployment requires deciding what to act on, whether current conditions support that action, and how to verify progress and respond to failure. An external harness makes these decisions explicit while retaining learned control capabilities. Figure 1 illustrates this broader need through three humanoid examples.

Recent manipulation-oriented harnesses organize perception, control primitives, and feedback around object interactions (Zhang et al., 2026; Chen et al., 2026b). Harness VLA assigns contact-rich manipulation to a learned policy and uses analytic primitives for preparation, transport, and recovery, including mobile-base staging. HumanoidHarness addresses a different coordination problem: the learned controller may itself perform locomotion, posture changes, or whole-body interaction, while a supporting turn or sidestep changes both the ego view and the body configuration from which execution resumes. Recovery must restore the conditions for continued whole-body behavior and assess progress using ego vision and proprioception, while underlying controllers maintain bal-

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

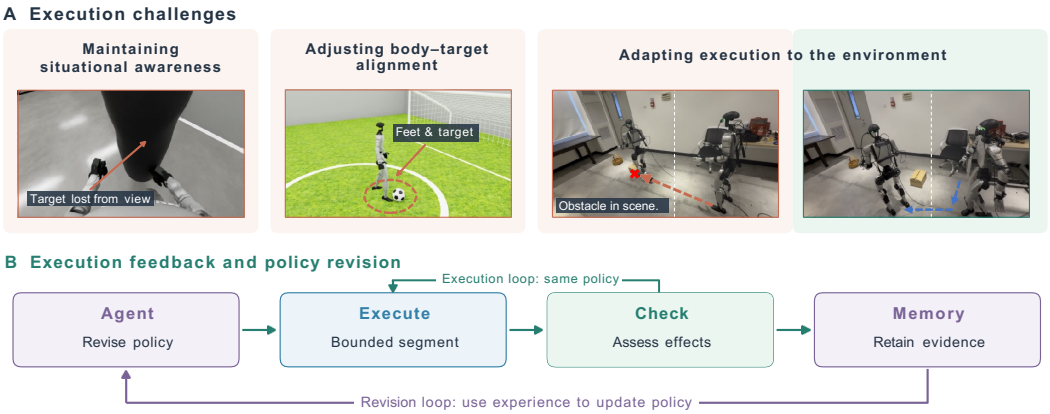


Figure 1: **From execution conditions to agentic policy revision.** Top: maintaining situational awareness (Boxing), adjusting body–target alignment (Football), and adapting execution to the environment (hardware obstacle avoidance). Hardware montages combine left/right crops of snapshots showing direct VLA execution failing to avoid a box and Harness-assisted bypass toward a basket, in separate trials with different obstacle placements. Dashed routes are schematic, not measured trajectories. World/external views are included for visualization and are excluded from agent input. Bottom: checks guide execution, while retained experience supports policy revision.

ance. Our harness therefore coordinates learned-controller calls, body adjustments, and observation checks within an execution policy spanning tasks beyond manipulation.

Closing this loop enables recovery; self-improvement requires using experience to revise the execution policy itself. Retained evidence can motivate changes to controller stages, checks, and recovery transitions—for example, adding a reobservation condition after repeated target loss. Subsequent execution then tests the revised policy. This separates immediate feedback-driven execution from experience-guided changes to the policy that governs it.

We introduce **HumanoidHarness**, an agentic framework integrating an editable policy template, supervised execution, and experience memory. The template exposes controller stages, observation checks, and recovery transitions for agent revision; validation and compilation turn the resulting policy into an executable graph. A runtime executes bounded action segments, while a supervisor uses tool-based checks or agent judgments to select subsequent graph transitions. Reflection Shinn et al. (2023) and task memory connect policy revisions to observed effects and guide later reuse without changing pretrained controllers or support primitives. Simulation and hardware share this loop, with agent inputs limited to ego vision, proprioception, and a human-provided task name and completion-criterion description. We evaluate HumanoidHarness on simulated and real-world humanoid tasks, examining self-improvement through experience-guided policy revision and its effect on subsequent task execution. Our contributions are:

1. **The first harness for self-improving humanoid whole-body control tasks:** we introduce an executable harness designed to revise execution policies through experience. It integrates observation, pretrained control, body adjustments, and verification into a shared closed-loop execution framework for simulation and hardware.

2. **An editable policy template:** we expose controller stages, checks, and recovery transitions as structured fields that an agent can revise and compile into an executable graph.

3. **Experience-guided agentic self-improvement:** we couple reflection with task memory to revise execution policies from observed outcomes and support reuse across attempts and task instances.

2 Related Work

Autonomous humanoid task execution. Autonomous humanoid tasks require connecting instruction understanding with coordinated locomotion, perception, and manipulation. Language-grounded behavior planning composes action and sensing behaviors to execute instructions and correct failures on the CENTAURO platform (Wang et al., 2024). Being-0 connects foundation-model reasoning to modular locomotion and manipulation skills through a vision–language Connector, with execution feedback informing subsequent skill selection (Yuan et al., 2025). EgoActor grounds instructions and egocentric observations into spatially aware locomotion, head movements, manipulation commands, and human interaction actions (Bai et al., 2026). Complementary work learns visual policies for coordinated humanoid behavior. LeVERB connects a vision–language policy to a reinforcement-learned whole-body controller through latent instructions (Xue et al., 2025). WholeBodyVLA combines latent action learning from egocentric videos with manipulation-aware locomotion control (Jiang et al., 2026). MotionWAM uses video-world-model features and unified motion tokens for autonomous whole-body loco-manipulation (Zheng et al., 2026), while HALOMI learns loco-manipulation with active perception from human demonstrations through coordinated head–hand tracking (Zhao et al., 2026). HumanoidArena benchmarks egocentric hierarchical policies and motion trackers on tasks requiring lower-body coordination (Wang et al., 2026). Our focus is experience-guided revision of the execution policy that selects, checks, and coordinates existing controllers across task attempts.

Robot foundation policies and hierarchical control. RT-2 transfers vision–language pretraining to action prediction, while OpenVLA provides an open model and efficient adaptation procedures for robot manipulation (Brohan et al., 2023; Kim et al., 2024). Flow-based models such as π_0 learn continuous robot actions across diverse tasks and embodiments (Black et al., 2024). World action models form a related family: GigaWorld-Policy-0.5 learns from coupled visual dynamics and actions, but decodes actions without generating future video at inference time (GigaWorld Team et al., 2026). Our harness operates on the action interface of these pretrained robot policies; it does not identify WAMs with VLAs or require access to their internal representations. Task and motion planning couples discrete task structure with geometric feasibility (Garrett et al., 2021); SayCan combines language-model task reasoning with learned skill affordances (Ahn et al., 2023). Our work studies how an external policy selects, checks, and revises the use of existing controllers during deployment, with their parameters and primitive implementations held fixed.

Executable programs and embodied harnesses. Code as Policies and ProgPrompt express robot behavior as programs over perception and action interfaces (Liang et al., 2023; Singh et al., 2023). Inner Monologue feeds observations and execution feedback into language-based planning (Huang et al., 2023), while behavior trees provide modular control structures for switching and recovery (Colledanchise & Ögren, 2018). Recent harnesses make the model–robot interface a central design choice. Show-Harness exposes discrete semantic motion and gripper units, grounds them through embodiment-specific interpreters, and organizes perception, reasoning, action history, and recovery around this interface (Chen et al., 2026b). It supports both zero-shot frontier VLM control and adaptation of smaller models. HarnessVLN coordinates perception, memory, and execution for instruction-following and object-goal navigation, including humanoid deployment (Chen et al., 2026a). HumanoidHarness exposes an editable policy of controller segments, checks, and guarded transitions, which is validated and compiled into a state machine for whole-body task execution.

Memory and experience-driven improvement. Harness VLA uses execution traces, success rules, and failure models to guide the composition of a frozen VLA with fixed analytic primitives (Zhang et al., 2026). Voyager iteratively improves executable skills from feedback in Minecraft (Wang et al., 2023), while ASPIRE diagnoses and repairs robot programs and distills validated fixes into a reusable skill library (Lu et al., 2026). ENPIRE supports real-world policy improvement through automated reset, rollout, verification, and agent-driven changes to training infrastructure and algorithm code (Xiao et al., 2026). HumanoidHarness uses reflection and task memory to revise controller stages, checks, and recovery transitions for humanoid whole-body tasks. These revisions update a restricted execution policy while keeping pretrained controllers and support primitives fixed; retained experience connects each revision to its observed effects across attempts.

3 HUMANOIDHARNESS

3.1 PROBLEM SETUP AND SYSTEM OVERVIEW

Let g be the human-provided task name and completion-criterion description, o_t the egocentric and proprioceptive observations, and π_θ a pretrained robot policy (VLA or WAM). HumanoidHarness adapts an execution program C that invokes π_θ or a permitted support primitive from Π_{aux}. The editable object specifies controller stages, action arguments, checks, and recovery transitions; model parameters and primitive implementations remain unchanged. Self-improvement thus concerns how existing capabilities are organized and revised using execution experience.

Figure 2 separates policy authoring from compilation, supervision, and actuation. The agent proposes a policy; the compiler admits its executable form; the supervisor selects the next stage; and the runtime executes a bounded segment. New observations and checks then determine how execution continues. Simulation and hardware share this loop, with platform adapters providing sensing, actuation, and stop/reset semantics.

The information contract is also shared. Agent-visible inputs comprise ego images, available proprioceptive channels, and g; history contains its own policies, actions, and assessments derived from these inputs. Raw simulator rewards, privileged target coordinates, and evaluator-private labels are excluded. The task criterion describes the objective, not its current truth value. This boundary applies equally to checkpoint checks, reflection, and memory.

3.2 RESTRICTED POLICY AND PROGRAM COMPILATION

The agent authors a structured policy rather than directly modifying low-level control code. Its executable representation is

$$C = (Q, q_0, T, P, \Gamma, B), \tag{1}$$

where Q contains states, q_0 is the entry, T contains guarded transitions, P specifies checkpoint checks, Γ binds states to controllers, and B contains execution limits. States invoke a pretrained policy segment, perform support/recovery actions, check observations, or terminate. Preparation and recovery belong to this same control structure: they can change the robot's viewpoint or configuration before returning control to the learned policy.

Admission and compilation. The validator checks schema and task identity, authorized primitives, command modes, finite arguments, and permitted predicate inputs. Graph checks cover unique identifiers, edge references, declared data types, structural reachability, and bounded visits for cycles. The compiler binds semantic stages to command templates, normalizes conditions, and injects deployment-specific speed and execution limits. The executable retains stage identifiers, guards, and controller bindings for supervision. Harness-owned limits cannot be overridden by agent-authored fields. Rejected edits return diagnostics for repair without granting motion authority. These checks constrain capabilities and resources; structural validity does not prove physical success or dynamic stability.

3.3 CHECKPOINT CHECKS AND SUPERVISORY TRANSITIONS

The supervisor maintains the active state, stage cursor, and checkpoint history. After each action segment, it obtains new observations and gathers evidence through tool-based checks, agent judgments, or both:

$$e_{t+1} = \text{Check}_P(o_{t+1}, \tau_t), \qquad q_{t+1} = \delta_C(q_t, e_{t+1}), \tag{2}$$

where τ_t supplies the permitted execution history. Checks can assess visual progress or conditions derived from proprioception. Their results retain their source and associated observations in the trace. The admitted guards determine whether to continue the current behavior, select a support action, enter recovery, hand off, or terminate. An agent judgment supplies transition evidence; it does not itself authorize arbitrary motion.

For example, after a learned-policy segment, a visibility check may indicate that the target has left the ego view. The current policy can route execution to a short scan or repositioning stage, inspect the new view, and resume the learned controller if its guard is satisfied. This example describes a

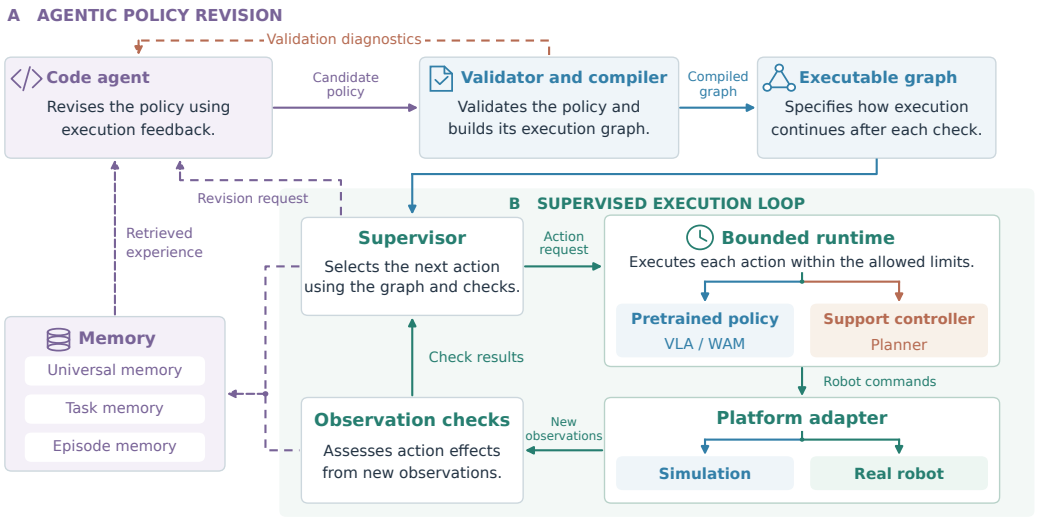


Figure 2: **HumanoidHarness: executable control and experience-guided revision.** The code agent edits a restricted policy template; validation and compilation produce an executable graph or return diagnostics. The supervisor follows graph guards using tool-based checks or agent judgments. The runtime dispatches bounded segments to a pretrained policy (VLA/WAM) or support planner through a shared platform adapter. Supervisor decisions and observation checks jointly form execution traces retained in memory, which is organized into universal, task, and episode layers. Solid arrows show execution/data flow; dashed arrows show feedback and reuse. Graph transitions reuse the admitted policy; edits require recompilation.

possible control sequence. If the existing stages or guards cannot express an appropriate continuation, the agent instead revises the policy using the observation history and termination reason. A routine transition changes only the active state; a policy edit changes C and requires validation and compilation again.

3.4 Bounded Runtime and Platform Adapters

For the supervisor-selected control source z_t and exposed arguments u_t, the runtime dispatches

$$a_t = \begin{cases} \pi_\theta(o_t, g), & z_t = \text{policy}, \\ \pi_j(o_t; u_t), \quad \pi_j \in \Pi_{\text{aux}}, & z_t = \text{aux}. \end{cases} \tag{3}$$

Support commands include permitted backoff, sidestep, settling, turning, and slow-advance primitives. Their arguments specify motion mode, speed, and movement/facing direction; available hand actions depend on the platform and task. A primitive name denotes a command template, not a guarantee that its intended physical effect occurs. Both learned-policy calls and support actions execute in segments, with requested horizon h_t bounded by

$$\widehat{h}_t = \min\{h_t, h_{\max}, B_{\text{remaining}}\}. \tag{4}$$

The runtime checks remaining resources and available interlocks, dispatches through the adapter, accounts for consumed execution, and applies stop/hold semantics. Consequently, a segment permitted by the state machine may still be shortened or rejected at runtime. New observations are checked before the next transition; agent-based assessment occurs at this boundary, outside low-level actuation. Restarting consumes the attempt budget, while an ordinary transition or rejected edit alone does not constitute another physical attempt.

Platform adapters preserve this division of responsibility while exposing different capabilities. The G1 SONIC adapter supports body motion and hand control through its Body Only interface, subject to task permissions. Its runtime serializes controller sources and checks command identity, fresh observations/actions, and remaining budget; failed source confirmation triggers hold. Source

acknowledgement establishes controller selection, not achieved displacement or task success. A terminal state or checkpoint judgment likewise does not replace independent outcome verification. Algorithm 1 in Appendix A specifies the full shared protocol and separates agent-visible execution evidence from private evaluation results.

3.5 Experience-Guided Policy Revision

After unsuccessful execution, the agent inspects retained camera checkpoints and action history, records a causal hypothesis, proposed change, expected effect, and lesson, and produces

$$C_{k+1} = \mathcal{A}(g, C_k, \tau_k, H_k), \tag{5}$$

where H_k contains prior proposals and their assessed effects. Revisions may change controller selection, stage order, movement arguments, checks, or recovery conditions. For example, evidence of recurring loss of visibility may motivate adding a reobservation condition before resuming a learned-policy stage, rather than repeating the same sequence. Such a diagnosis remains a hypothesis until subsequent execution supports it.

The reflection journal connects each proposed change to the policy actually executed. A behavior signature ignores descriptive metadata and binds the reflected proposal to that policy; repeating a previously failed policy requires an explicit retry rationale. Runtime events and checkpoint observations then provide evidence for assessing the expected effect. This linkage supports comparison across revisions without treating an agent's explanation as a verified cause. The journal retains uncertainty and distinguishes assessed completion from independent task success.

3.6 Interaction Memory and Reuse

Persistent memory is indexed by task identity and records the observation/action contract and provenance. It stores prior policies, observations, actions, assessments, and lessons, but no random seeds or evaluator-private labels. The host owns historical records and provides a bounded snapshot for later authoring. Thus, the current execution trace describes what just happened, while retrieved experience supplies earlier interventions and their assessed effects. The agent can use both when selecting what to retain or change in the next candidate.

Memory supports improvement within repeated attempts and reuse on later task instances. Optional task rules link summarized lessons to their source executions and remain hypotheses until evaluated beyond those instances. Reuse does not presume that a successful action sequence will work from a different viewpoint or posture; current observations still govern checkpoint decisions.

4 Experiments

We evaluate task performance, revision, memory, and recovery. Tables 1 and 2 report experimental results. Figures 3 and 4 examine interaction budgets and improvement across successive policy revisions.

4.1 Tasks, Controls, and Evaluation Protocol

Task setting. We evaluate Unitree G1 tasks in HumanoidArena (Wang et al., 2026): Football, Double Desk, Pick Place Box, Open Door, Sit Sofa, Boxing, and Vision Navigation. GR00T N1.7 and GWP-0.5 provide learned controllers. Checks and memory obey Section 3's ego-vision/proprioception boundary; private benchmark measurements remain outside agent input.

Study design. We follow the HumanoidArena evaluation protocol (Wang et al., 2026): three random seeds and 20 rollouts per seed, totaling 60 rollouts per task and configuration, with the same benchmark object configurations and initial robot states across methods. Task-specific success conditions, failure termination rules, and episode limits follow the benchmark. We compare direct execution with Harness-assisted execution for GR00T N1.7 and GWP-0.5, using GPT-5.6 sol as the agent. Success rate is the primary metric; Table 1 reports task rates and their macro-average. Section 4.3 specifies the separate comparison of execution strategies.

Metrics and accounting. We report task success and agent-call costs. Agent calls include judgment, revision, and repair. Tool-call budgets additionally count execution, observation, and checking operations.

4.2 Overall Task Performance

GWP-0.5 with HumanoidHarness improves on six tasks and tie on Open Door, with the largest gains on Football (+25.0 points) and Boxing (+21.7). These task-level rates alone do not establish gains under matched interaction budgets.

GR00T + Harness improves on six tasks and ties on Double Desk, raising the macro-average from 29.4% to 37.4%. Its largest gains are on Boxing and Vision Navigation. Table 1 reports success rates for both controllers; gains and means are computed from the displayed values.

Dependence on the underlying controller. HumanoidHarness improves both controllers, but their performance gap persists: harness-assisted macro-average success is 37.4% for GR00T and 81.9% for GWP-0.5. GR00T remains unsuccessful on Double Desk, while GWP-0.5 reaches 28.3%. These results indicate that better execution organization improves the use of existing skills, while achievable performance remains constrained by the underlying controller's capabilities.

4.3 Does Revision Help beyond Additional Retries?

Study setting. We compare execution strategies using GWP-0.5 on the same seven HumanoidArena tasks, following the benchmark's task definitions, success criteria, and termination rules. Each task uses three seeds and 20 episodes per seed, totaling 420 episodes across the seven tasks. Each evaluation episode allows up to four attempts, stopping at the first success; resetting after a failed attempt remains within the same episode. Agent-based variants use GPT-5.6 sol. Table 2 reports the arithmetic mean success rate across all seven tasks, counting an episode as successful if any of its four attempts succeeds, and the mean number of agent calls per episode, summed across all attempts and including failed episodes. Figure 3 compares rollout and tool-call budgets on Boxing only. Rollout budgets of 60, 120, 180, and 240 per task allow one to four attempts for each of the 60 episodes.

Comparison methods. *Retry only* repeats direct GWP-0.5 execution. *Fixed state machine* uses manually designed stages and transitions without policy revision. *Agent-only checks* delegates every checkpoint judgment to the agent without visual checking tools. A *primitive-call agent* selects controller and support-primitive calls directly, conceptually following Harness VLA (Zhang et al., 2026). *HumanoidHarness* combines checked execution, policy revision, and persistent memory.

Success under interaction budgets. Figure 3a shows that increasing the rollout budget from 60 to 240 raises Retry only from 73.3% to 75.0%, while the primitive-call agent and HumanoidHarness rise from 75.0% to 86.7% and 95.0%, respectively. At the largest budget, HumanoidHarness exceeds these baselines by 20.0 and 8.3 percentage points. Figure 3b also shows lower tool-call costs at shared success levels: reaching 85% requires 960 calls for HumanoidHarness versus 1,035 for the primitive-call agent. The final reported points reach 95.0% with 1,097 calls and 86.7% with 1,297 calls, respectively. Retry only gains just 1.7 points over this budget range, compared with 20.0 points for HumanoidHarness. Additional attempts alone therefore yield limited improvement in this setting; using execution feedback to revise the strategy converts interaction budgets into success more effectively. The comparison supports the complete harness, with the contribution of episode memory examined in Section 4.4.

Improvement across policy revisions. We run consecutive episodes on the 60 HumanoidArena settings for each task, following the benchmark's three-seed, 20-setting-per-seed protocol, and revise the execution policy using the resulting experience. Figure 4 plots success rate across these settings against the number of policy revisions, up to 200. Revision 0 is the initial policy; subsequent points are recorded every 25 revisions. Each rate is the number of successful settings divided by 60. HumanoidHarness shows a generally steady upward trend: Football increases from 55.0% to 83.3%, and Boxing from 73.3% to 95.0%. Boxing is nondecreasing at the reported points; Football has a temporary dip at revision 75 before improving further.

Table 1: **Task success on HumanoidArena.** Success rates (%) for direct and Harness-assisted execution. Gains are percentage points; means are macro-averages across the seven tasks.

	GR00T N1.7 (VLA)			GWP-0.5 (WAM)		
Task	Direct	+ Harness	Gain	Direct	+ Harness	Gain
Football	0.0	10.0	+10.0	55.0	80.0	+25.0
Double Desk	0.0	0.0	0.0	10.0	28.3	+18.3
Pick Place Box	75.0	86.7	+11.7	93.3	95.0	+1.7
Open Door	11.7	16.7	+5.0	100.0	100.0	0.0
Sit Sofa	13.3	15.0	+1.7	83.3	100.0	+16.7
Boxing	68.3	83.3	+15.0	73.3	95.0	+21.7
Vision Navigation	37.7	50.0	+12.3	55.0	75.0	+20.0
Mean	29.4	37.4	+8.0	67.1	81.9	+14.8

Table 2: **Execution strategy comparison.** Seven-task mean success and mean agent calls per episode; at most four attempts.

Method	Success (%)	Agent calls
Retry only	67.6	0.0
Fixed state machine	73.3	5.0
Agent-only checks	79.3	7.2
Primitive-call agent	76.1	5.3
HumanoidHarness	81.9	3.8

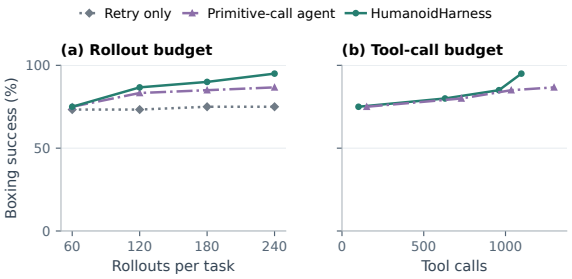


Figure 3: **Budgeted Boxing performance.** Cumulative success versus (a) rollout budget over 60 episodes, allowing up to four attempts, and (b) tool calls. Retry only is shown in (a).

4.4 Episode Memory Ablation

We compare HumanoidHarness with *No Episode Memory*, which removes stored experience from earlier episodes while retaining the current policy, observations within the current episode, and policy revision. Both variants retain the same checks, recovery actions, validation, and runtime bounds. This comparison examines whether past episode experience improves subsequent policy revision and reuse. Figure 4 reports the Football and Boxing results.

Effect of episode memory. Removing episode memory lowers final success and makes improvement less consistent. At revision 200, No Episode Memory reaches 68.3% on Football and 90.0% on Boxing, compared with 83.3% and 95.0% for HumanoidHarness: gaps of 15.0 and 5.0 percentage points. It also shows more regressions at the sampled revisions: two on each task, versus one on Football and none on Boxing for HumanoidHarness. These trends support retaining experience across episodes to improve both final performance and the consistency of policy improvement; individual edits can still reduce success.

4.5 Physical Deployment Evaluation

We evaluate safety-oriented online takeover and long-horizon task reasoning on a physical Unitree G1. Task-specific GR00T N1.7 VLAs are trained beforehand and remain fixed during harness development. Each VLA predicts SONIC tokens and hand opening/closing; the tokens are passed to the SONIC decoder, a motion-control model (Luo et al., 2026). GPT-5.6-sol revises the execution policy, while local Qwen 3.8 27B on an RTX 5090 assesses camera observations in brief-answer mode (Appendix B).

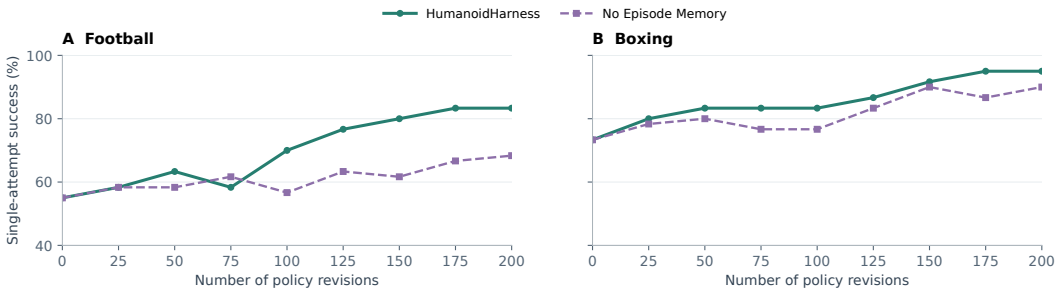


Figure 4: **Policy evolution with and without episode memory.** Success rate over 60 HumanoidArena settings during consecutive episodes, sampled every 25 policy revisions up to 200. The horizontal axis counts policy edits.

Walk to the basket. GR00T N1.7 reaches the basket along an unobstructed route. With a box in its path, direct execution continues forward and pushes the box with the robot's feet, exposing a collision risk (Figure 5). In the harness trial, GR00T begins walking along a clear route before a box is introduced. Visual assessment triggers a switch to auxiliary locomotion: the robot moves left around the obstacle, then resumes GR00T execution toward the basket. The ego/exo sequence in Figure 6 illustrates this safety-oriented intervention during ongoing execution.

Walk to the table and pick up the cola. The cola-task GR00T N1.7 VLA grasps from near the table when the cola is visible, but does not complete the task from a distant initial position. The agent reasons about the table location, directs the locomotion Planner to search for and approach the target, and hands off to GR00T once the robot is beside the table with the cola in view (Figure 7). This demonstrates long-horizon reasoning across navigation, target acquisition, and whole-body grasping, with the VLA handling fine manipulation.

Final-system evaluation. After iterative development, the final system achieves **10/10 successes on each task**, without further policy edits, corrective human guidance, or emergency stops. The results support online takeover and long-horizon task composition in these settings; isolating individual revisions requires matched-budget controls.

Additional experimental details and demonstration videos are available on our project website: `https://dapper-kringle-05f5a1.netlify.app`.

5 Conclusion

We presented HumanoidHarness, an agentic framework for humanoid task execution that combines pretrained controllers with an editable execution policy, supervised feedback, and episode memory. Validation and compilation turn policy revisions into bounded state-machine execution, allowing the agent to adapt controller stages, checks, and recovery transitions while keeping pretrained parameters fixed. On HumanoidArena, harness-assisted execution improves task completion, and successive policy revisions produce a generally steady increase in success rate. Episode memory supports both higher final performance and more consistent improvement by retaining experience across episodes. Physical G1 experiments demonstrate obstacle-responsive takeover and navigation-to-grasp coordination under the same execution framework. These results support experience-guided revision of execution policies as a practical route to improving humanoid behavior. Broader task coverage and evaluation of transfer across environments remain directions for future work.

AI Usage Statement

We used generative AI tools to assist with manuscript writing, language polishing, and the statistical analysis of experimental results. All AI-assisted text was reviewed and edited by the authors, who take full responsibility for the analyses and the final content of the paper.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, pp. 287–318. PMLR, 2023. URL https://arxiv.org/abs/2204.01691.

Yu Bai, Mingming Yu, Chaojie Li, Ziyi Bai, Xinlong Wang, and Börje F. Karlsson. EgoActor: Grounding task planning into spatial-aware egocentric actions for humanoid robots via visual-language models. *arXiv preprint arXiv:2602.04515*, 2026. URL https://arxiv.org/abs/2602.04515.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Yang Chen, Lirong Che, Zhenyu Huang, Wenbo Fu, Chuang Wang, Xu Cao, Daqi Liu, Yuzhe Yang, Jian Su, and Lan-Zhe Guo. HarnessVLN: Unifying training-free embodied navigation through an agent harness. *arXiv preprint arXiv:2609.15195*, 2026a. URL https://arxiv.org/abs/2609.15195.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-Harness: Just a VLM agent can play robots. *arXiv preprint arXiv:2609.10522*, 2026b. URL https://arxiv.org/abs/2609.10522.

Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018. doi: 10.1201/9780429489105. URL https://arxiv.org/abs/1709.00084.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021. doi: 10.1146/annurev-control-091420-084139.

GigaWorld Team, Angen Ye, Angyuan Ma, Boyuan Wang, Chaojun Ni, Fangzheng Ye, Guan Huang, Guo Li, Guosheng Zhao, Haodong Yan, et al. GigaWorld-Policy-0.5: A faster and stronger WAM empowered by AutoResearch. *arXiv preprint arXiv:2607.13960*, 2026. URL https://arxiv.org/abs/2607.13960.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, pp. 1769–1782. PMLR, 2023. URL https://arxiv.org/abs/2207.05608.

Haoran Jiang, Jin Chen, Qingwen Bu, Li Chen, Modi Shi, Yanjie Zhang, Delong Li, Chuanzhe Suo, Chuang Wang, Zhihui Peng, and Hongyang Li. WholeBodyVLA: Towards unified latent VLA for whole-body loco-manipulation control. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2512.11047.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://arxiv.org/abs/2209.07753.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic /skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026. URL https://arxiv.org/abs/2607.00272.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Fernando Castañeda, Sirui Chen, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, et al. SONIC: Supersizing motion tracking for natural humanoid whole-body control. *Science Robotics*, 11(117):eaed4592, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317. URL https://arxiv.org/abs/2209.11302.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jin Wang, Arturo Laurenzi, and Nikos Tsagarakis. Autonomous behavior planning for humanoid loco-manipulation through grounded language model. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. URL https://arxiv.org/abs/2408.08282.

Taowen Wang, Zikang Xie, Bin Yang, Yunheng Wang, Zizhao Yuan, Yuetong Fang, Yixiao Feng, Yichi Wang, Xingyu Chen, Haodong Chen, et al. HumanoidArena: Benchmarking egocentric hierarchical whole-body learning. *arXiv preprint arXiv:2606.17833*, 2026. URL https://arxiv.org/abs/2606.17833.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026. URL https://arxiv.org/abs/2606.19980.

Haoru Xue, Xiaoyu Huang, Dantong Niu, Qiayuan Liao, Thomas Kragerud, Jan Tommy Gravdahl, Xue Bin Peng, Guanya Shi, Trevor Darrell, Koushil Sreenath, and Shankar Sastry. LeVERB: Humanoid whole-body control with latent vision-language instruction. *arXiv preprint arXiv:2506.13751*, 2025. URL https://arxiv.org/abs/2506.13751.

Haoqi Yuan, Yu Bai, Yuhui Fu, Bohan Zhou, Yicheng Feng, Xinrun Xu, Yi Zhan, Börje F. Karlsson, and Zongqing Lu. Being-0: A humanoid robotic agent with vision-language models and modular skills. *arXiv preprint arXiv:2503.12533*, 2025. URL https://arxiv.org/abs/2503.12533.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Jiaji Rao, Zhengru Fang, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026. URL https://arxiv.org/abs/2607.08448.

Zehui Zhao, Yuxuan Zhao, Gaojing Zhang, Chenxi Liu, Maolin Zheng, and Wenzhao Lian. HALOMI: Learning humanoid loco-manipulation with active perception from human demonstrations. *arXiv preprint arXiv:2606.18772*, 2026. URL https://arxiv.org/abs/2606.18772.

Jia Zheng, Teli Ma, Yudong Fan, Zifan Wang, Shuo Yang, and Junwei Liang. MotionWAM: Towards foundation world action models for real-time humanoid loco-manipulation. *arXiv preprint arXiv:2606.09215*, 2026. URL https://arxiv.org/abs/2606.09215.

A Implementation and Deployment Details

This appendix complements the agent–compiler–supervisor–runtime decomposition in Section 3.

A.1 Observation and Command Contracts

The observation interface exposes ego camera images and available proprioceptive channels: joint positions and velocities, body angular velocity, and projected gravity. Tilt and upright alignment can be derived from gravity. The task name and completion-criterion description state the objective; they do not supply an episode-specific outcome label. Checks, execution history, and memory respect the same information boundary.

The simulation adapter exposes backoff, left/right sidesteps, settling, turning to scan, slow advance, and hand actions subject to task permissions. Commands specify motion mode, speed, movement/facing vectors, and optional hand states. The robot-frame convention uses right, forward, and up axes. The compiler clips speed and injects execution bounds; the primitive's name does not guarantee its intended physical effect.

A.2 Validation and Platform Runtime

Admission checks enforce task identifiers, registered primitives, allowed motion/hand modes, finite arguments, and supported predicate inputs. Agent-authored planner documents cannot override harness-owned budget or safety fields. State-machine validation checks unique nodes, edge references, declared data types, structural reachability of a terminal/handoff, and bounded visits for cycles. The executable retains stage identifiers, conditions, and controller bindings. Compilation diagnostics can inform repair, but structural reachability does not establish physical termination or success.

Before dispatch, the runtime checks available uprightness signals against configured limits. The simulation observation adapter derives a stability flag from measured tilt. These checks do not guarantee collision avoidance or dynamic stability.

The G1 SONIC policy runner performs checkpoint assessment and supervisory stage selection. Its Body Only command runtime supports body motion and hand control and serializes learned-policy and support segments through a single publisher. Requests include command and observation identifiers; duplicates, unsettled preceding commands, stale action chunks, and budget violations are rejected. Failed source confirmation triggers hold. Source acknowledgement does not establish achieved displacement, completed manipulation, or task success. Hand commands are subject to the configured task permissions and validation rules.

A.3 Reflection and Memory Configuration

The reflection journal links evidence, a causal hypothesis, the proposed change, its expected effect, and a lesson to the behavior signature of the executed policy. Runtime events and camera checkpoints retain this provenance. A repeated failed policy requires a retry rationale. Assessed completion and runtime termination remain distinct from independently measured task success.

The host supplies a bounded task-memory snapshot. Retrieval matches exact task identity, scope, action contract, provenance, and schema version. It selects the most recent compatible records and presents them in chronological order; the agent cannot rewrite historical entries. The implementation defaults to six retrieved records and supports a maximum of 50. Each record retains a lesson, policy, observation, assessment, and the latest 20 actions. These are implementation defaults, not a reconstruction of the settings used for every reported experiment. Memory retains attempts, policies, observations, actions, assessments, and lessons, but no random seeds. Optional task-rule reuse links rules to reviewed execution identifiers and treats them as hypotheses without held-out validation. Evaluation split membership belongs to the experiment manifest, not memory.

A.4 SHARED EXECUTION PROTOCOL

Algorithm 1 separates policy editing, admission, supervisory transitions, and bounded actuation. An independent evaluator obtains outcome y_k and evidence E_k from the execution and task specification, $\mathcal{V}(\tau_k, g) \rightarrow (y_k, E_k)$. Its private labels are excluded from agent feedback and memory. A program terminal label, successful command acknowledgement, or visual goal judgment alone does not establish independently verified success.

Algorithm 1 Shared policy authoring and state-machine execution

Require: Human-provided task name and completion-criterion description g; Pretrained robot policy π_θ; support tools Π_{aux}; platform adapter D
Require: Execution and agent-call budgets B; permitted task memory H
Ensure: Evaluator outcome y, execution trace τ, and revision history H

Initialize D and budgets; load H; $\tau \leftarrow \varnothing$, $C \leftarrow \varnothing$
revise $\leftarrow$ true
while the configured execution and reasoning budgets permit continuation **do**
    Observe ego images and proprioceptive state o through D
    if policy termination or runtime fault **then break**
    if episode ended or attempt budget exhausted **then**
        Record agent-visible trace; **if** no permitted restart remains **then break**
        Restart through D; debit attempt budget; revise $\leftarrow$ true
        continue
    end if
    if revise **then**
        $C' \leftarrow \mathcal{A}(g, o, C, \tau, H)$ ▷ Edit the restricted policy
        $(\text{valid}, M, d) \leftarrow$ VALIDATEANDCOMPILE(C', D, B)
        if $\neg$valid **then** record diagnostics d; **continue**
        $C \leftarrow C'$; initialize the admitted machine state q; revise $\leftarrow$ false
    end if
    $(z, u, h, P_q) \leftarrow$ SUPERVISORDISPATCH(M, q)
    if q is an action state **then**
        Runtime executes the selected segment with horizon $\min(h, h_{\max}, B_{\text{remaining}})$
        Apply platform stop/hold semantics; debit resource usage
    end if
    Observe o' through D
    $e \leftarrow$ CHECK(P_q, o', τ) ▷ Tools, agent judgment, or both
    Append action, observations, check results, and their sources to τ
    $(q', \text{revise}) \leftarrow$ SUPERVISORTRANSITION(M, q, e)
    $q \leftarrow q'$; record reflections and agent-visible assessments in H
    if terminal state without a revision/restart request **then break**
end while
Stop/hold through D; save artifacts; obtain independent outcome y for evaluation only
return (y, τ, H) ▷ A state-machine terminal label is not proof of success

B Physical Experiments and Visual-Assessment Latency

B.1 Task Setup and Controller Responsibilities

Unitree G1 uses separately trained GR00T N1.7 VLAs for basket approach and cola grasping. Each VLA predicts SONIC tokens and hand opening/closing; the tokens condition the SONIC decoder motion-control model (Luo et al., 2026). Model weights remain fixed during harness development. GPT-5.6-sol reasons about tasks and revises the execution policy; the locomotion Planner supplies approach and support motions. The Body Only interface supports both body motion and hand control, as described in Appendix A.

Basket task. Direct GR00T N1.7 reaches the basket on a clear route but continues into an intervening box (Figure 5). With the harness, a box introduced during walking triggers takeover, leftward bypass, and resumed GR00T execution (Figure 6), illustrating a safety benefit of online supervision. Insertion is a task perturbation, not corrective guidance. The separate direct and harness recordings are not a matched-initial-state comparison.

Cola task. Starting away from the table, the agent identifies it as an intermediate goal, uses the Planner to approach and seek the cola, then hands off to GR00T N1.7 with the target in view. Figure 7 shows the progression from table approach to bottle grasping and lifting. This links long-horizon reasoning to precise VLA execution. Success requires reaching the table and picking up the cola.

Evaluation scope. Each task succeeds in ten final-system trials without policy edits, corrective guidance, or emergency stops (Table 3); development attempts are excluded. These counts assess the final system rather than the contribution of individual revisions. The direct-VLA trials illustrate nominal behavior and an obstacle-related failure mode.

Table 3: **Physical G1 evaluation after policy development.** Each task uses ten trials of the final system.

Task	Successes / trials	Success (%)
Walk to the basket (introduced obstacle)	10/10	100.0
Walk to the table and pick up the cola	10/10	100.0

B.2 Visual-Assessment Backend and Latency

We compare GPT-5.6-luna (low reasoning, priority service) with Qwen 3.8 27B deployed locally on an RTX 5090, using the same image frames and history inputs. Qwen is evaluated in standard and brief-answer modes; the latter is used in the final physical experiments. Latency runs from request dispatch to receipt of the complete assessment response (Table 4).

Table 4: **Visual-assessment latency with matched frame/history inputs.** Values are measured median request-to-complete-response times; they exclude subsequent robot motion.

Backend / response mode	Median latency (s)
GPT-5.6-luna / low reasoning, priority service	5.33
Qwen 3.8 27B / standard	3.03
Qwen 3.8 27B / brief answer (deployed)	1.69

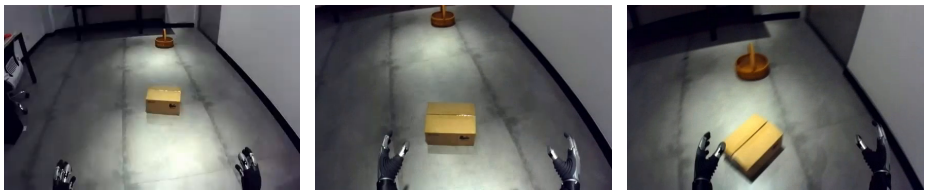
The backends give consistent judgments in our qualitative comparison; we deploy Qwen's brief-answer mode for its lower latency. The 1.69 s median excludes observation acquisition, checkpoint scheduling, and actuation. Online takeover denotes intervention during execution; it does not imply a hard real-time safety guarantee.

(a) GR00T N1.7 VLA: unobstructed basket approach



Start: 0.19 s | Approach: 2.52 s | Beside basket: 4.39 s

(b) GR00T N1.7 VLA: obstructed route



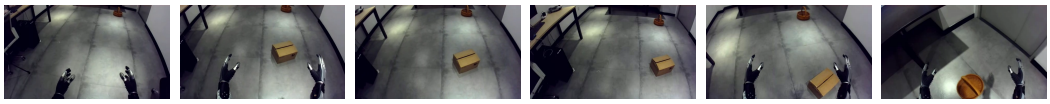
Box in path: 0.23 s | Continues forward: 3.06 s | Box displaced: 3.63 s

Figure 5: **Direct GR00T N1.7 execution on Unitree G1.** (a) The robot reaches the basket along an unobstructed path. (b) With a box blocking the path, the VLA continues walking forward, pushing the box with the robot's feet instead of avoiding it.

HumanoidHarness: obstacle introduction, bypass, and task resumption

1. Clear route | 2. Box appears | 3. Reposition | 4. Lateral move | 5. Pass box | 6. Reach basket

Egocentric observations



External views (presentation only)

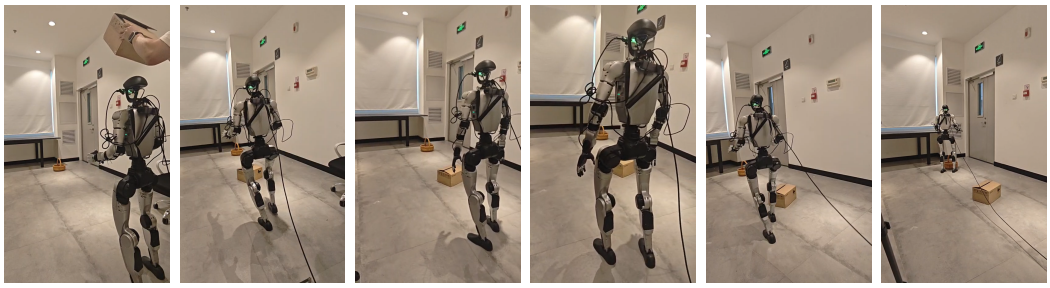


Figure 6: **Online takeover and recovery with HumanoidHarness.** Egocentric (top) and external (bottom) views show successive stages of the basket task. When a box is introduced during walking, HumanoidHarness switches from GR00T N1.7 to auxiliary locomotion, guides the robot around the obstacle, and resumes VLA execution to reach the basket.



(a) Approach table | (b) Target in view | (c) Grasp bottle | (d) Lift bottle

Figure 7: **Table approach and cola grasping with HumanoidHarness.** The egocentric sequence shows table approach, target acquisition, grasping, and lifting. The agent directs navigation through the Planner, then hands off to GR00T N1.7 for grasping with the cola in view.