A HIERARCHICAL AGENT FOR LONG-HORIZON MULTI-ROBOT COLLABORATION WITH FEEDBACK-DRIVEN TASK CONTINUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon multi-robot collaboration requires coordinating operations whose feasibility depends on the outcomes of other robots' actions. A central challenge is aligning high-level decisions with physical execution: an unmet prerequisite, a failed operation, or a changed scene state can invalidate a planned next step. We present a hierarchical agent that coordinates multiple robots through a common execution interface. A vision–language agent selects robot roles and skills from the task, observations, structured physical-state feedback, capabilities, and execution history. The interface validates requests, dispatches learned manipulation and supporting skills, and returns outcomes or unmet conditions for the next decision. It also supports replaceable coordination models and skill backends. We evaluate direct handover, object-dependent sorting, model configurations, and task continuation in a four-robot simulation. Across 75 sorting episodes, the agent selects every receiver and destination correctly and completes 62 deliveries (82.7%). Under 45 recoverable post-handover drops, feedback-driven coordination completes 15 deliveries, whereas a one-shot plan with the same execution components completes none. A paired 2×2 study shows both high-level models operating with both backends and selecting all destinations correctly, while delivery varies with low-level performance. The results show that aligning decisions with physical feedback supports continuous collaboration across robots and configurable components.

1 INTRODUCTION

Long-horizon multi-robot collaboration requires coordinating operations that depend on other robots' progress. A high-level plan specifies the next operation, but physical execution determines whether its prerequisites hold. We study this dependency through acquisition, direct handover, and delivery: a handover may leave the giver holding the object, the receiver may not be ready, or the object may fall after transfer. The central problem is to keep task-level coordination consistent with the evolving physical state so that collaboration can continue toward its goal. Figure 1 illustrates how the actual transfer outcome determines the next feasible operation.

Language and vision–language models ground task instructions in executable skills (Ahn et al., 2022), allocate work across robot teams (Kannan et al., 2023; Mandi et al., 2023), and use execution feedback in subsequent decisions (Huang et al., 2022; Guo et al., 2023; Li et al., 2026). In collaborative manipulation, feedback must also update which robot can act and which operation remains valid as possession and readiness change. We study this problem through direct handover and continued delivery, selecting the next robot and skill from the observed physical state and shared goal.

We present a hierarchical agent for long-horizon multi-robot collaboration. A vision–language coordinator selects robot roles and skills from the task, observations, structured physical-state feedback, capabilities, and execution history. A common interface validates requests, dispatches learned manipulation and supporting controllers, and returns outcomes or unmet conditions. This feedback updates the next decision while retaining the delivery goal. The interface also supports different coordinators and skill backends, separating collaborative decisions from continuous action generation.

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.

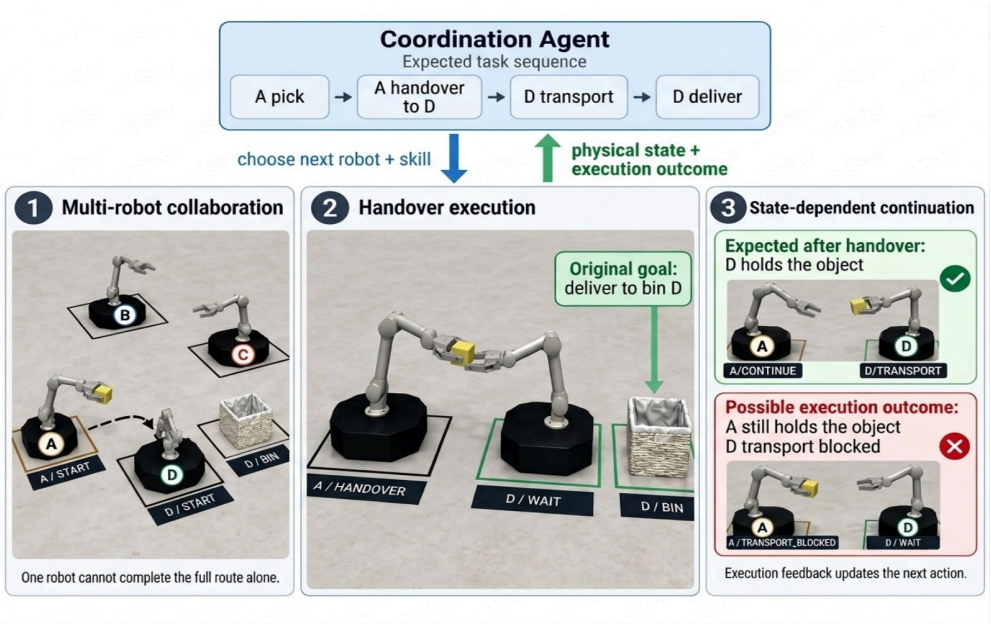

Figure 1: **Physical feedback determines the next feasible operation.** After attempted transfer from A to D, transport requires D to hold the object. A retained grasp instead requires an updated decision. Conceptual illustration.

We evaluate five objects in a four-robot simulation. End-to-end handover and sorting test whether one agent can organize acquisition, direct transfer, and delivery across three receiver routes; it completes 62/75 sorting episodes (82.7%) after selecting every receiver and destination correctly. Controlled interruptions and held-object starts test state-dependent continuation. Under 45 recoverable post-handover drops, feedback-driven coordination completes 15 deliveries, compared with none for a one-shot plan using the same execution components. A paired 2×2 study further tests the interface with two coordinators and two skill backends.

Our contributions are threefold:

- **A hierarchical agent framework for multi-robot collaboration.** The hierarchical agent uses a central coordinator to organize robot roles and learned skills across interdependent operations, demonstrated through acquisition, direct handover, and destination delivery.
- **A configurable multi-robot collaboration system.** A common execution interface separates high-level coordination from continuous skill execution and supports replacement of both the coordination model and the skill backend.
- **Physical-feedback-driven task continuation.** Structured physical-state feedback and execution outcomes update subsequent robot and skill decisions while retaining the original task objective, supporting continuation from interrupted or intermediate states.

2 Related Work

2.1 Multi-robot task coordination and manipulation

Language models organize robot teams through centralized or distributed decisions. SMART-LLM decomposes tasks, forms coalitions, and allocates work before invoking robot skills (Kannan et al., 2023). RoCo uses robot-specific language agents for dialogue and validation-guided planning, followed by centralized multi-arm execution (Mandi et al., 2023). COHERENT combines a central assigner with robot-specific language-model executors and adjusts proposals from execution feedback (Liu et al., 2024). These methods place assignment and execution reasoning at different points in the robot team.

Collaborative manipulation connects task structure to physical execution. RoboFactory generates demonstrations using vision–language-generated subgoals, motion primitives, and logical, spatial, and temporal checks (Qin et al., 2025). We study direct object transfer and subsequent receiver delivery in a persistent simulation world. Separate conditions test interruption recovery and destination selection. Holding execution components fixed across coordinator comparisons isolates task-level selection from the behavior of individual skills.

2.2 Language-based skill planning and execution feedback

Language-based planning grounds instructions in executable robot skills. SayCan combines task relevance with value-function estimates of skill feasibility (Ahn et al., 2022). Inner Monologue feeds success detection and scene descriptions into subsequent prompts (Huang et al., 2022). DoReMi generates constraints with a language model and checks them during execution using a vision–language model, triggering replanning when violations occur (Guo et al., 2023). These approaches motivate using execution feedback to choose subsequent operations.

Recent agents further structure the connection between reasoning and execution. RoboClaw uses a vision–language controller, memory, and tool interfaces to schedule learned policies and retries (Li et al., 2026). Vesta conditions subtasks on a goal, observation, and memory, with a separate actor (Bjorck et al., 2026); CaP-X examines primitive abstraction, interaction, and feedback in robot coding agents (Fu et al., 2026). We study how an interface-based coordinator connects state-dependent operations across robot roles, using direct handover and continued delivery as the evaluation setting.

2.3 Learned manipulation policies

Learned policies generate continuous actions beneath a task coordinator. Action Chunking with Transformers predicts action sequences (Zhao et al., 2023), while Diffusion Policy uses conditional denoising (Chi et al., 2024). ManiFlow learns manipulation through consistency flow training (Yan et al., 2025). We reuse ManiFlow weights across registered learned skills and active robot roles, conditioning execution on the operation selected by the coordinator.

Vision–language–action models such as OpenVLA, π_0, and $\pi_{0.5}$ integrate perception, language, and action generation (Kim et al., 2025; Black et al., 2025; Physical Intelligence et al., 2025). Flow matching also underlies π_0, so it is compatible with the VLA formulation. Our system separates task-level coordination at skill boundaries from continuous manipulation by the learned executor.

3 Hierarchical Agent for Feedback-Driven Collaboration

Our method addresses the mismatch between a planned collaborative sequence and the state actually produced by physical execution. As shown in Figure 2, a high-level vision–language coordinator selects the next robot and operation, a shared execution interface validates and dispatches the request, and a configurable execution layer carries it out in the persistent simulation world. The resulting object possession, receiver readiness, progress, and execution outcome return to the agent before its next decision. The following subsections formulate the collaborative task, describe agent-based coordination, explain feedback-driven continuation, and specify the interchangeable skill execution layer.

3.1 Collaborative Task Formulation

We consider collaborative manipulation in which operations performed by different robots depend on the outcomes of preceding actions. Our task instantiation involves a giver that acquires an object and transfers it to a receiver, which continues delivery to a designated bin. The task g specifies the object and either a destination or a complete object-to-bin sorting rule. In sorting, the coordinator selects the receiver and destination before manipulation and retains this choice throughout the episode. The delivery goal remains in effect when execution requires additional attempts or recovery.

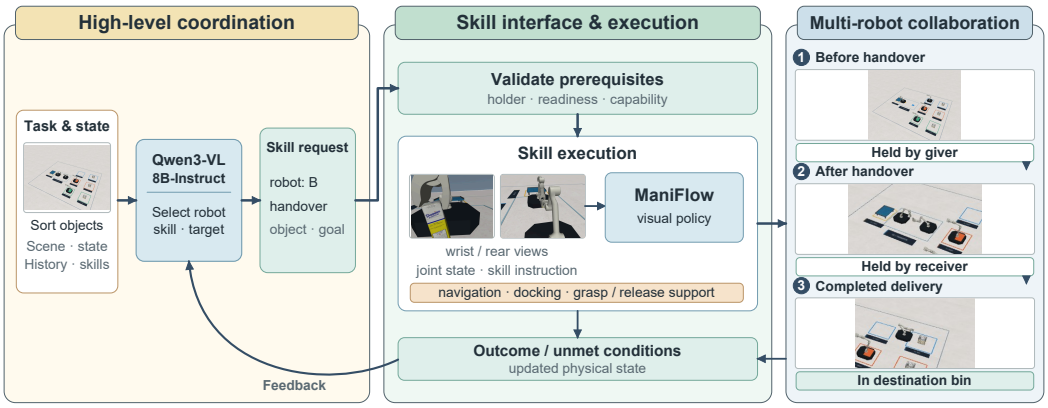

Figure 2: **Coordination and execution form a feedback loop.** The VLM selects robot, skill, and target; the interface validates and executes requests and returns outcomes. Camera inputs feed ManiFlow; archived views illustrate collaboration. Layout draws on COHERENT (Liu et al., 2024).

3.2 Agent-Based Robot and Skill Coordination

At decision round k, the coordinator receives the task g, an overview RGB image I_k, structured state facts f_k, the registered capability catalog $\mathcal{K}$, execution history H_k, and remaining budgets b_k. It produces a request

$$u_k = \mathcal{C}(g, I_k, f_k, \mathcal{K}, H_k, b_k), \tag{1}$$

where $\mathcal{C}$ denotes the VLM-based coordinator. The budgets bound coordination rounds, skill attempts, rejected requests, and wall-clock runtime. The index k counts coordinator requests rather than individual arm-control steps.

The capability catalog describes picking, inter-robot handover, transport, placement, and retrieval of dropped objects. Each entry specifies the executing robot, supported objects and destinations, availability, preconditions, and expected effect. The full registered catalog is supplied alongside the current state, leaving the coordinator to select a request. A skill request names the skill, robot, object, and target, with a short explanation. Sorting additionally uses a destination-selection request, and the coordinator can request completion or abort.

3.3 Physical Feedback and Task Continuation

Structured state describes the object location and possession, receiver readiness, and docking and delivery progress on which collaborative operations depend. In our simulation implementation, these facts are obtained from simulator geometry and executor state and supplied alongside the scene image to inform coordination decisions. The history records preceding requests and their outcomes, providing context for subsequent choices.

The execution interface checks each request against the task, capability specification, current state, and attempt budget. Handover requires giver possession and receiver readiness; transport requires receiver possession, and placement additionally requires docking. An accepted skill request invokes its executor in the current simulation world; a rejected request returns the unmet conditions and their observed and required values. For example, a transport request is rejected if the giver still holds the object after a handover attempt. These checks enforce execution conditions while leaving skill and destination selection to the coordinator.

After each non-terminal request, its outcome is added to the history, and a fresh observation is obtained before the next decision. The coordinator can select another skill, retry within the remaining budget, or retrieve an unheld object from the floor when that capability is available. Retrieval can restore receiver possession, allowing delivery to continue toward the retained destination. The coordinator may request completion or abort; run limits also bound execution.

3.4 CONFIGURABLE SKILL EXECUTION

We instantiate learned manipulation with ManiFlow (Yan et al., 2025). A skill invocation selects a registered task description ℓ_s and routes the active robot's wrist and rear-camera images and measured joint positions to the policy. At an action-generation step t, the deployed policy uses the two most recent observations to generate an action block:

$$A_t = \pi_\theta(I^{\text{wrist}}_{t-1:t}, I^{\text{rear}}_{t-1:t}, q_{t-1:t}, \ell_s; \epsilon) \in \mathbb{R}^{15\times 7}, \tag{2}$$

where q denotes measured joint positions and ϵ is sampling noise. Each of the 15 commands specifies target positions for six arm joints and the gripper.

The runtime executes the queued commands sequentially and requests another block when the queue is empty. At the start of each learned skill invocation, the runtime clears the observation history and any remaining commands, preventing the previous invocation's action queue from carrying over. The same loaded policy is reused across learned skill calls within an episode. Thus, the coordinator specifies the robot and skill, while the executor supplies the corresponding continuous arm and gripper commands.

Learned commands are integrated with wheel navigation, docking, grasp maintenance, and release assistance. During handover, the policy controls the receiver's arm and gripper, while execution logic manages giver presentation and coordinates attachment and release. Grasp maintenance preserves an object-to-gripper attachment once the configured grasp-acquisition conditions are met; the holder field is derived from this attachment state. The transfer implementation relaxes collision constraints on the giver's fingers. Receiver-specific routing retains the calibrated camera and control conventions. Retrieval of dropped objects follows learned execution paths for the medicine bottle and soup can, and a geometric grasp routine for the sugar box. These components translate skill requests into object transfers and subsequent manipulation within the same task.

We also use Multi-Task Action Chunking Transformer (MT-ACT), the instruction-conditioned ACT backend implemented in RoboManipBaselines following RoboAgent (Zhao et al., 2023; Bharadhwaj et al., 2024). It encodes the skill description and combines it with camera images and joint state to predict chunks of arm and gripper joint targets. Our demonstration-trained checkpoints comprise a multi-task policy and a specialist used only for cracker-box handover, each with its own normalization metadata. The same robot–skill–target request interface routes observations to the selected checkpoint and returns execution outcomes, leaving high-level coordination unchanged.

4 EXPERIMENTS

We evaluate the three capabilities introduced by the framework. End-to-end delivery and object-dependent sorting examine whether one agent can organize interdependent operations across multiple robot roles. Coordinator and skill-backend replacements test whether the common collaboration interface supports different model configurations. Controlled interruptions, unavailable capabilities, and held-object initializations examine whether physical feedback is reflected in subsequent decisions and whether the original task is continued or stopped appropriately. We report both stage-level collaboration outcomes and final task completion so that successful handover is distinguished from subsequent delivery. A frozen-log evaluation additionally measures whether subsequent requests match the supplied physical state, complementing end-to-end recovery outcomes.

4.1 SHARED SETUP AND EVALUATION PROTOCOL

Robots, workspace, and learned skills. The simulated workcell contains four Dingo mobile bases equipped with Z1 arms in MuJoCo (Todorov et al., 2012). Each episode involves robot A and one receiver operating on one object in a persistent world. The main configuration uses Qwen3-VL-8B-Instruct (Bai et al., 2025) with ManiFlow. The dataset contains 900 demonstrations covering 18 object–skill combinations: picking, handover, and placement for five objects, and floor pickup for three. Each combination contributes 45 training and five validation demonstrations. Training runs for 200 epochs, after which the validation-selected checkpoint is frozen. Evaluation retains the navigation and manipulation assistance described in the method; dropped-object retrieval uses learned execution for the medicine bottle and soup can, and a geometric routine for the sugar box.

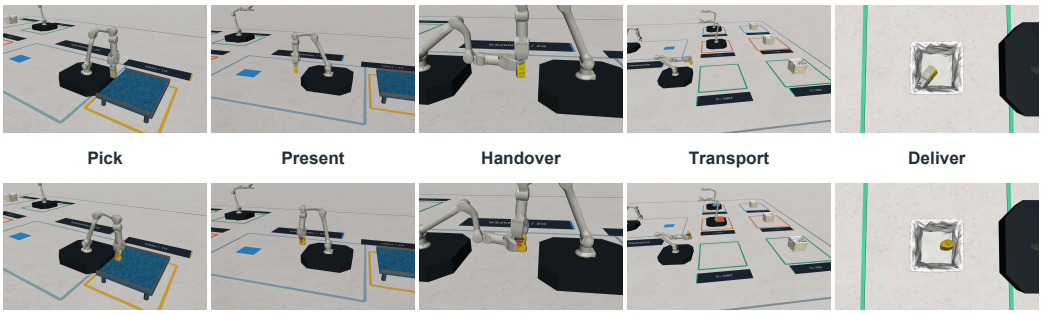

Figure 3: **Two geometries share a collaborative delivery sequence.** Archived views show sugar (top) and mustard (bottom) picked and presented by A, transferred to D, and delivered to bin D.

Controllers and model configurations. Controller comparisons hold the execution system fixed while changing how subsequent operations are selected. The closed-loop agent queries the VLM at each decision boundary, feedback rules select actions from the current state and capabilities, and the one-shot planner queries the same Qwen model once before executing without replanning. These controllers share the policy, execution assistance, request interface, and run budgets. A separate 2×2 study combines Qwen or GLM coordination with ManiFlow or MT-ACT execution through the same interface.

Initializations and statistical protocol. Paired initializations control local position variation across the main controller comparisons. The source object is translated within a 1 cm-radius XY disk around its calibrated position, with height and orientation fixed; compared runs share the physical state and policy seed. Each tested object has 15 initializations per condition, except held-object takeover, which has five. The coordination conditions contain 604 evaluated runs from 610 assignments, excluding six runs associated with three unavailable held-object initializations. Sorting adds 150 runs. We report Wilson 95% intervals for rates and object-balanced paired differences with within-object bootstrap intervals. The prespecified comparison is recoverable-drop performance against the one-shot planner; additional comparisons are descriptive.

Metrics and outcome definitions. An independent final-state audit and the controller's terminal request jointly determine delivery and stopping outcomes; handover is evaluated separately. Delivery requires the object to be unheld, inside and in contact with the correct bin for at least one second of simulation time, a satisfied work-region audit, and a finish request. Recoverable-drop success additionally requires actual fault exposure; earlier failures remain in the denominator. When retrieval is unavailable, success instead means an appropriate abort after exposure. Handover completion follows the assisted transfer criterion described below. Destination correctness is scored independently of the interface check for a supported robot–bin pairing.

4.2 Receiver Selection and End-to-End Collaborative Sorting

Sorting requires the coordinator to select the receiver and destination before manipulation. The supplied rule maps the cracker box to B, the medicine bottle and soup can to C, and the sugar box and mustard bottle to D. Each object has 15 paired initializations, with no injected drop and retrieval disabled. The agent chooses the correct destination in all 75 episodes and completes 62 deliveries (82.7%); feedback rules choose all destinations correctly and complete 61 (81.3%; Table 1). The selected destination is retained throughout each episode.

Delivery outcomes connect the selected route to completed collaboration. The agent delivers 8/15 cracker boxes, 15/15 medicine bottles, 15/15 soup cans, 11/15 sugar boxes, and 13/15 mustard bottles. All three receiver routes therefore contribute completed episodes. Figure 3 shows sugar-box and mustard-bottle sequences: the agent selects D and bin D, A acquires and presents the object, and D receives it and completes delivery. The handover and continuation analyses below separate the stages underlying these outcomes. An illustrative sorting workcell is shown in Figure 5.

Table 1: Receiver selection and end-to-end sorting.

Object	Receiver	Correct target (both)	Agent delivery	Rule delivery
Cracker box	B	15/15	8/15 (53.3)	8/15 (53.3)
Medicine bottle	C	15/15	15/15 (100.0)	15/15 (100.0)
Soup can	C	15/15	15/15 (100.0)	14/15 (93.3)
Sugar box	D	15/15	11/15 (73.3)	11/15 (73.3)
Mustard bottle	D	15/15	13/15 (86.7)	13/15 (86.7)
All objects	B/C/D	75/75	62/75 (82.7)	61/75 (81.3)

Both controllers select the correct destination on their first choice in every episode (Correct target). Delivery entries are successes/episodes (percentage). Each object has 15 paired starts.

4.3 Direct Handover Completion and Transfer Time

The agent completes 65 of 71 dispatched handovers in fixed-receiver delivery and 68 of 72 in sorting, corresponding to 91.5% and 94.4%. Both conditions begin with 75 episodes, so four and three episodes, respectively, do not reach handover. Completion requires receiver attachment, giver release, and retreat under the existing execution assistance. Final delivery reaches 59/75 and 62/75, respectively, separating transfer completion from subsequent task completion. Table 2 provides the per-object counts. The presentation and handover stages in Figure 3 show this change in possession.

Handover timing measures the transfer itself, excluding preparation and model waiting. Table 2 reports the median and first and third quartiles over all dispatched attempts, including failures. Per-object medians range from 7.87 to 11.39 s for fixed-receiver delivery and 6.09 to 8.38 s for sorting. Broader coordination and process timings, which include additional stages, are provided in Table 10.

Table 2: Object transfer and its execution cost.

Task	Object	N	n	k	Time (s)
Fixed B	Cracker box	15	13	13	10.40 [9.60, 10.67]
Fixed B	Medicine bottle	15	14	14	10.09 [9.28, 10.78]
Fixed B	Soup can	15	14	14	7.87 [7.57, 8.57]
Fixed B	Sugar box	15	15	13	8.19 [7.51, 8.82]
Fixed B	Mustard bottle	15	15	11	11.39 [9.76, 25.28]
Sorting	Cracker box	15	12	12	7.93 [7.75, 8.38]
Sorting	Medicine bottle	15	15	15	8.38 [7.92, 8.61]
Sorting	Soup can	15	15	15	6.09 [5.89, 6.99]
Sorting	Sugar box	15	15	13	6.12 [5.85, 6.40]
Sorting	Mustard bottle	15	15	13	7.93 [6.87, 11.08]

Agent results. N: evaluated episodes; n: dispatched handovers; k: completed handovers. Time is the median [first, third quartile] in wall-clock seconds over all n attempts, including failures; preparation and model waiting are excluded. Completion uses the existing assisted transfer criterion.

4.4 Replacing Coordinators and Skill Backends

The configuration study replaces both layers through the same collaboration interface. Qwen3-VL-8B-Instruct and GLM-4.1V-9B-Thinking (V Team et al., 2025) are each combined with ManiFlow and MT-ACT. All four groups use the same 25 physical initializations, five per object, for 100 runs under the sorting rules above. Injected drops and retrieval are disabled. Each comparison changes one layer while holding the other fixed.

Each backend retains its verified execution configuration, including the MT-ACT cracker-handover expert and its normalization. The comparison therefore evaluates complete backend configurations under a common coordination interface. We measure correct destination selection, handover completion, and final delivery. Model-specific request settings and paired successful-run timings are reported in the appendix; all task failures, including timeouts, remain in the success-rate denominators.

Both coordinators select every destination correctly, with similar delivery rates within each backend (Table 3). ManiFlow yields 20/25 deliveries with Qwen and 19/25 with GLM; both complete 22 of 23 attempted handovers. MT-ACT yields 9/25 and 10/25 deliveries, with 11 of 18 handovers completed for each coordinator. Thus, both coordinators operate through the common interface, while the deployed execution backend accounts for the larger observed difference. Tables 11 and 12 provide per-object outcomes and paired successful-run timings.

Table 3: Coordinator and skill-backend configurations.

Coordinator	Backend	Target	Handover	Delivery
Qwen	ManiFlow	25/25	22/23	20/25 (80.0)
GLM	ManiFlow	25/25	22/23	19/25 (76.0)
Qwen	MT-ACT	25/25	11/18	9/25 (36.0)
GLM	MT-ACT	25/25	11/18	10/25 (40.0)

Qwen denotes Qwen3-VL-8B-Instruct; GLM denotes GLM-4.1V-9B-Thinking. Target is first-choice correct/25, handover is completed/attempted, and delivery is successes/25 (percent). All cells use the same 25 starts, five per object. Six original GLM timeouts remain failures. Each backend retains its native execution assists.

4.5 State-Consistent Decisions and Interrupted-Task Continuation

State-consistent decisions. We score recorded requests against physical facts and execution contracts across 317 closed-loop episodes and 1,971 decisions. A request is state-consistent when its robot, operation, and target match the canonical next action derived from possession, support, readiness, docking, delivery state, available capabilities, and remaining attempts. Receiver possession permits transport, docking permits placement, and an unheld floor object requires retrieval or stopping according to capability availability. We separately score decisions following physical-state changes and deduplicated floor events. Appendix A.4 provides the scoring details.

The agent chooses a state-consistent next request after 1,323 of 1,384 physical-state transitions (95.6%). All 280 handover-ready decisions request handover, and all 176 stable-delivery decisions request completion. Across 106 floor events, the first response is correct in 69 cases (65.1%), and an eventual state-consistent response occurs in 104 (98.1%). When retrieval is available, these counts are 39/40 and 40/40; when stopping is required, they are 30/66 and 64/66. First-to-eventual improvement captures adjustment over subsequent decisions, rather than physical retrieval or delivery success. These decision-level counts complement the task outcomes below.

Feedback enables additional attempts when the preceding operation has not established the next action's conditions. In normal fixed-receiver delivery, the agent and feedback rules each complete 59/75 episodes, compared with 51/75 for the one-shot plan. The eight additional successes follow a second transport attempt after sugar-box docking falls outside tolerance. With a controlled drop during post-handover transport, the agent completes 15/45 deliveries, feedback rules 17/45, and the one-shot plan 0/45 (Table 4). The agent–one-shot difference is 33.3 percentage points, with a 95% paired interval of [22.2, 44.4] and exact McNemar $p = 6.10 \times 10^{-5}$ (Table 7).

Recovery outcomes depend on both invoking retrieval and completing the remaining delivery. The recoverable condition covers the sugar box, medicine bottle, and soup can. Of the agent's 45 episodes, 41 expose the fault, 40 invoke retrieval, 19 complete retrieval, 17 complete subsequent transport, and 15 finish delivery (Table 9). Disabling retrieval for these objects changes the outcome to appropriate stopping: the agent achieves 40/45 and rules 41/45. For mustard and cracker, which lack retrieval, the corresponding counts are 23/30 and 24/30. Here, all 24 exposed agent cases first request unsupported retrieval; 23 terminate after rejection. Figure 4 traces one successful soup-can recovery: the drop feedback prompts retrieval, followed by renewed transport and placement into the original bin B.

Held-object initializations test continuation from progress established before coordination begins. Frozen physical prefixes construct either A-held or B-held states. The agent and rules each complete 22/23 A-held episodes; from B-held states they complete 18/24 and 23/24, respectively. Successful receiver-held episodes proceed to transport and placement without repeating acquisition. The

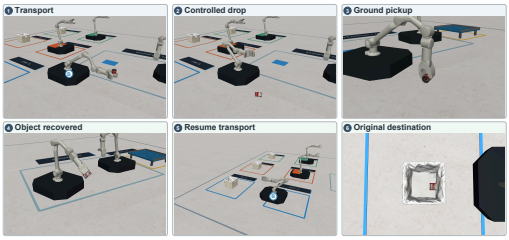

Figure 4: **Recovery preserves the original destination.** In one recorded episode, B drops a soup can after handover, retrieves it, resumes transport, and delivers it to bin B. All views retain the same execution assistance.

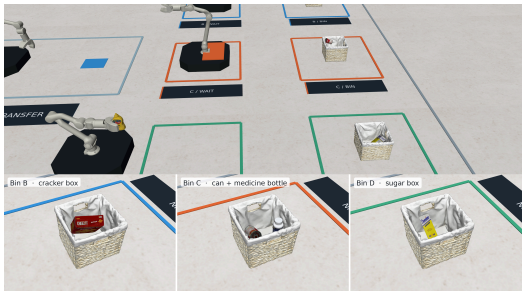

Figure 5: **Object-dependent destinations in the sorting workcell.** The illustrative scene shows cracker in B, medicine and soup in C, and sugar and mustard in D.

five paired failures unique to the agent involve state interpretation or subsequent action selection, identifying where continuation decisions affect the remaining task.

Table 4: Coordination outcomes. Entries are successes/evaluated episodes (percentage).

Condition	Closed-loop agent	Feedback rules	One-shot plan
Normal delivery	59/75 (78.7)	59/75 (78.7)	51/75 (68.0)
Recoverable drop	15/45 (33.3)	17/45 (37.8)	0/45 (0.0)
Recovery disabled (stop)	40/45 (88.9)	41/45 (91.1)	–
No recovery skill (stop)	23/30 (76.7)	24/30 (80.0)	–
A-held takeover	22/23 (95.7)	22/23 (95.7)	–
B-held takeover	18/24 (75.0)	23/24 (95.8)	–

Success means completed delivery, except in the two capability-limited conditions, where it means correct stopping after fault exposure. A-held and B-held takeover exclude two and one unavailable initializations per method, respectively; – denotes a method not evaluated.

5 Conclusion

We presented a hierarchical agent for interdependent multi-robot manipulation. A vision–language coordinator selects receivers and skills through a common interface, using execution feedback to connect acquisition, direct handover, and delivery. Sorting and controlled-drop experiments demonstrate destination selection and continuation of interrupted tasks. Coordinator and backend replacements further show operation through the same interface across model configurations. The framework organizes learned skills into collaborative tasks; broader task settings and physical robot teams remain directions for future work.

Reproducibility Statement

Sections 3 and 4 specify the observation and action interfaces, training data split, execution assistance, controller budgets, paired initialization protocol, and outcome definitions. Appendix A provides per-object results, uncertainty estimates, execution-process statistics, and the separate resource-measurement protocol. The underlying records retain frozen configurations, initialization hashes, policy seeds, model requests, skill outcomes, and per-call timings. These records support recomputation of the reported results and distinguish unavailable initializations from evaluated episodes.

Statement of AI Use

AI tools assisted with software development, experimental orchestration and analysis, literature search, drafting, and translation. The authors are responsible for the final content and source verification.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, 2022. URL https://arxiv.org/abs/2204.01691v2.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025. URL https://arxiv.org/abs/2511.21631v2.

Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking. In *IEEE International Conference on Robotics and Automation*, 2024. doi: 10.1109/ICRA57147.2024.10611293. URL https://arxiv.org/abs/2309.01918.

Johan Bjorck, Zhiqi Li, Yunze Man, Jing Wang, An-Chieh Cheng, Sifei Liu, Shihao Wang, Zhiding Yu, Abhishek Badki, Stan Birchfield, Valts Blukis, Yevgen Chebotar, Siyi Chen, Sicong Leng, Yu-Cheng Chou, Tianli Ding, Boyi Li, Zhengyi Luo, Hang Su, Jonathan Tremblay, Tingwu Wang, Bowen Wen, Jimmy Wu, Xianghui Xie, Hanrong Ye, Hongxu Yin, K. R. Zentner, Liangyan Gui, Yu-Xiong Wang, Yuke Zhu, Linxi "Jim" Fan, and Jan Kautz. Vesta: A Generalist Embodied Reasoning Model, 2026. URL https://arxiv.org/abs/2606.20905v1.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *The International Journal of Robotics Research*, 2024. URL https://arxiv.org/abs/2303.04137v5.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi "Jim" Fan. CaP-X: A Framework for Benchmarking and Improving Coding Agents for Robot Manipulation, 2026. URL https://arxiv.org/abs/2603.22435v1.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding Language Model by Detecting and Recovering from Plan-Execution Misalignment, 2023. URL https://arxiv.org/abs/2307.00329v4.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models, 2022. URL https://arxiv.org/abs/2207.05608v1.

Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models, 2023. URL https://arxiv.org/abs/2309.10062v2.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R 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. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, Jiayao Ma, Xin He, Yongjian Shen, Yang Yang, Guanghui Ren, Maoqing Yao, Wenhao Wang, and Yao Mu. RoboClaw: An Agentic Framework for Scalable Long-Horizon Robotic Tasks, 2026. URL https://arxiv.org/abs/2603.11558v3.

Kehui Liu, Zixin Tang, Dong Wang, Zhigang Wang, Bin Zhao, and Xuelong Li. COHERENT: Collaboration of Heterogeneous Multi-Robot System with Large Language Models, 2024. URL https://arxiv.org/abs/2409.15146v1.

Zhao Mandi, Shreeya Jain, and Shuran Song. RoCo: Dialectic Multi-Robot Collaboration with Large Language Models, 2023. URL https://arxiv.org/abs/2307.04738v1.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization, 2025. URL https://arxiv.org/abs/2504.16054v1.

Yiran Qin, Li Kang, Xiufeng Song, Zhenfei Yin, Xiaohong Liu, Xihui Liu, Ruimao Zhang, and Lei Bai. RoboFactory: Exploring Embodied Agent Collaboration with Compositional Constraints, 2025. URL https://arxiv.org/abs/2503.16408v1.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng,

Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, Baoxu Wang, Bin Chen, Boyan Shi, Changyu Pang, Chenhui Zhang, Da Yin, Fan Yang, Guoqing Chen, Jiazheng Xu, Jiale Zhu, Jiali Chen, Jing Chen, Jinhao Chen, Jinghao Lin, Jinjiang Wang, Junjie Chen, Leqi Lei, Letian Gong, Leyi Pan, Mingdao Liu, Mingde Xu, Mingzhi Zhang, Qinkai Zheng, Sheng Yang, Shi Zhong, Shiyu Huang, Shuyuan Zhao, Siyan Xue, Shangqin Tu, Shengbiao Meng, Tianshu Zhang, Tianwei Luo, Tianxiang Hao, Tianyu Tong, Wenkai Li, Wei Jia, Xiao Liu, Xiaohan Zhang, Xin Lyu, Xinyue Fan, Xuancheng Huang, Yanling Wang, Yadong Xue, Yanfeng Wang, Yanzi Wang, Yifan An, Yifan Du, Yiming Shi, Yiheng Huang, Yilin Niu, Yuan Wang, Yuanchang Yue, Yuchen Li, Yutao Zhang, Yuting Wang, Yu Wang, Yuxuan Zhang, Zhao Xue, Zhenyu Hou, Zhengxiao Du, Zihan Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2507.01006.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, and Dieter Fox. ManiFlow: A General Robot Manipulation Policy via Consistency Flow Training. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 2268–2293. PMLR, 2025. URL https://proceedings.mlr.press/v305/yan25a.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.html.

A Detailed evaluation results

A.1 Success rates and paired comparisons

Table 5: Condition-level success rates and Wilson 95% confidence intervals (percent).

Condition	Agent	Rules	One-shot
E1-N	78.7 [68.1, 86.4]	78.7 [68.1, 86.4]	68.0 [56.8, 77.5]
E1-R	33.3 [21.4, 47.9]	37.8 [25.1, 52.4]	0.0 [0.0, 7.9]
E2-D	88.9 [76.5, 95.2]	91.1 [79.3, 96.5]	–
E2-U	76.7 [59.1, 88.2]	80.0 [62.7, 90.5]	–
E3-A	95.7 [79.0, 99.2]	95.7 [79.0, 99.2]	–
E3-B	75.0 [55.1, 88.0]	95.8 [79.8, 99.3]	–

E1-N: normal delivery; E1-R: recoverable drop; E2-D: recovery disabled; E2-U: no native recovery skill; E3-A/B: takeover with A/B holding the object. Denominators are given in Table 4.

Table 6: Per-object coordination outcomes.

Condition	Object	Agent	Rules	One-shot
E1-N	Cracker box	8/15 (53.3)	8/15 (53.3)	8/15 (53.3)
E1-N	Medicine bottle	14/15 (93.3)	14/15 (93.3)	14/15 (93.3)
E1-N	Soup can	13/15 (86.7)	13/15 (86.7)	13/15 (86.7)
E1-N	Sugar box	13/15 (86.7)	13/15 (86.7)	5/15 (33.3)
E1-N	Mustard bottle	11/15 (73.3)	11/15 (73.3)	11/15 (73.3)
E1-R	Medicine bottle	1/15 (6.7)	1/15 (6.7)	0/15 (0.0)
E1-R	Soup can	10/15 (66.7)	12/15 (80.0)	0/15 (0.0)
E1-R	Sugar box	4/15 (26.7)	4/15 (26.7)	0/15 (0.0)
E2-D	Medicine bottle	12/15 (80.0)	12/15 (80.0)	–
E2-D	Soup can	15/15 (100.0)	15/15 (100.0)	–
E2-D	Sugar box	13/15 (86.7)	14/15 (93.3)	–
E2-U	Cracker box	12/15 (80.0)	13/15 (86.7)	–
E2-U	Mustard bottle	11/15 (73.3)	11/15 (73.3)	–
E3-A	Cracker box	4/5 (80.0)	4/5 (80.0)	–
E3-A	Medicine bottle	3/3 (100.0)	3/3 (100.0)	–
E3-A	Soup can	5/5 (100.0)	5/5 (100.0)	–
E3-A	Sugar box	5/5 (100.0)	5/5 (100.0)	–
E3-A	Mustard bottle	5/5 (100.0)	5/5 (100.0)	–
E3-B	Cracker box	4/5 (80.0)	4/5 (80.0)	–
E3-B	Medicine bottle	3/5 (60.0)	5/5 (100.0)	–
E3-B	Soup can	5/5 (100.0)	5/5 (100.0)	–
E3-B	Sugar box	2/5 (40.0)	5/5 (100.0)	–
E3-B	Mustard bottle	4/4 (100.0)	4/4 (100.0)	–

Entries are successes/evaluated episodes (percentage). Condition definitions and success criteria follow Tables 4 and 5.

Table 7: Paired controller comparisons.

Condition	Comparison	n	Only 1st/2nd	Macro Δ (pp)	95% CI (pp)	p
E1-N	Agent / One-shot	75	8/0	+10.7	[+5.3, +16.0]	0.0078
E1-N	Agent / Rules	75	0/0	+0.0	[+0.0, +0.0]	1.0000
E1-N	Rules / One-shot	75	8/0	+10.7	[+5.3, +16.0]	0.0078
E1-R	Agent / One-shot	45	15/0	+33.3	[+22.2, +44.4]	6.10×10^{-5}
E1-R	Agent / Rules	45	1/3	-4.4	[-13.3, +4.4]	0.6250
E1-R	Rules / One-shot	45	17/0	+37.8	[+26.7, +48.9]	1.53×10^{-5}
E2-D	Agent / Rules	45	0/1	-2.2	[-6.7, +0.0]	1.0000
E2-U	Agent / Rules	30	0/1	-3.3	[-10.0, +0.0]	1.0000
E3-A	Agent / Rules	23	0/0	+0.0	[+0.0, +0.0]	1.0000
E3-B	Agent / Rules	24	0/5	-20.0	[-32.0, -8.0]	0.0625
Sorting	Agent / Rules	75	1/0	+1.3	[+0.0, +4.0]	1.0000

Differences equally average the first-minus-second success-rate difference over applicable objects; Only 1st/2nd counts discordant successes. Percentile intervals use 10,000 paired bootstrap resamples within each object, preserving its sample size and equal object weights; p-values use two-sided exact McNemar tests. The primary comparison is Agent / One-shot in E1-R. All others are descriptive. A zero-width bootstrap interval indicates no observed discordance, not population equivalence.

Table 8: Sorting delivery rates and Wilson 95% confidence intervals (percent).

Object	Agent	Rules
Cracker box	53.3 [30.1, 75.2]	53.3 [30.1, 75.2]
Medicine bottle	100.0 [79.6, 100.0]	100.0 [79.6, 100.0]
Soup can	100.0 [79.6, 100.0]	93.3 [70.2, 98.8]
Sugar box	73.3 [48.0, 89.1]	73.3 [48.0, 89.1]
Mustard bottle	86.7 [62.1, 96.3]	86.7 [62.1, 96.3]
All objects	82.7 [72.6, 89.6]	81.3 [71.1, 88.5]

There are 15 episodes per object and controller, and 75 per controller overall. Destination accuracy is reported separately from delivery.

A.2 EXECUTION PROCESS AND COORDINATION COST

Table 9: Recoverable-drop execution funnel (episode counts).

Controller	Evaluated	Drop exposed	Recovery called	Recovery complete	Transport complete	Delivered
Agent	45	41	40	19	17	15
Rules	45	41	39	20	19	17
One-shot	45	40	0	0	0	0

Recovery complete is executor-reported completion. All endpoint rates retain the 45-episode denominator. In E2-D, 41/45 episodes per controller expose the fault and correct stops are 40 for Agent and 41 for Rules. In E2-U, exposure is 24/30, with 23 and 24 correct stops, respectively.

Table 10: Mean controller calls and elapsed times per evaluated episode.

Condition	Controller	n	LLM calls	Skill calls	Retries	LLM wait (s)	Coord. (s)	Process (s)
E1-N	Agent	75	6.08	4.73	0.12	69.07	227.37	266.72
E1-N	Rules	75	0.00	4.73	0.12	0.00	157.04	194.54
E1-N	One-shot	75	1.00	4.51	0.00	24.49	175.80	213.77
E1-R	Agent	45	8.07	5.53	0.04	95.62	277.73	317.48
E1-R	Rules	45	0.00	5.60	0.04	0.00	184.31	222.80
E1-R	One-shot	45	1.00	3.84	0.00	24.43	152.28	189.92
E2-D	Agent	45	5.53	3.89	0.04	65.32	193.44	233.27
E2-D	Rules	45	0.00	3.89	0.04	0.00	126.36	165.09
E2-U	Agent	30	5.90	3.67	0.00	66.73	194.83	233.70
E2-U	Rules	30	0.00	3.67	0.00	0.00	128.48	167.70
E3-A	Agent	23	5.13	4.09	0.09	52.74	190.12	249.58
E3-A	Rules	23	0.00	4.09	0.09	0.00	134.82	192.65
E3-B	Agent	24	3.00	1.67	0.00	35.67	94.15	241.97
E3-B	Rules	24	0.00	2.08	0.08	0.00	62.98	199.26
Sorting	Agent	75	7.15	4.75	0.03	146.79	264.15	300.61
Sorting	Rules	75	0.00	4.75	0.04	0.00	114.70	152.86

All evaluated episodes contribute, including failures. Skill calls count accepted dispatches, excluding rejected and non-skill requests. Retries repeat a failed skill, excluding resumptions after interruption. Timings are nested: Coord. includes LLM wait and skill execution; Process also includes initialization, held-object prefix preparation, shutdown, and result collection. The columns must not be added. LLM wait includes queuing, communication, and generation; Process is per episode, not batch wall-clock time.

A.3 MODEL CONFIGURATIONS AND MATCHED-SUCCESS TIMING

Table 11: Per-object configuration outcomes.

Coordinator	Backend	Object	Handover	Delivery
Qwen	ManiFlow	Cracker box	3/3	3/5 (60.0)
Qwen	ManiFlow	Medicine bottle	5/5	5/5 (100.0)
Qwen	ManiFlow	Soup can	5/5	5/5 (100.0)
Qwen	ManiFlow	Sugar box	4/5	2/5 (40.0)
Qwen	ManiFlow	Mustard bottle	5/5	5/5 (100.0)
GLM	ManiFlow	Cracker box	3/3	3/5 (60.0)
GLM	ManiFlow	Medicine bottle	5/5	5/5 (100.0)
GLM	ManiFlow	Soup can	5/5	4/5 (80.0)
GLM	ManiFlow	Sugar box	4/5	2/5 (40.0)
GLM	ManiFlow	Mustard bottle	5/5	5/5 (100.0)
Qwen	MT-ACT	Cracker box	0/4	0/5 (0.0)
Qwen	MT-ACT	Medicine bottle	3/3	2/5 (40.0)
Qwen	MT-ACT	Soup can	3/4	2/5 (40.0)
Qwen	MT-ACT	Sugar box	3/5	3/5 (60.0)
Qwen	MT-ACT	Mustard bottle	2/2	2/5 (40.0)
GLM	MT-ACT	Cracker box	0/4	0/5 (0.0)
GLM	MT-ACT	Medicine bottle	3/3	2/5 (40.0)
GLM	MT-ACT	Soup can	3/4	3/5 (60.0)
GLM	MT-ACT	Sugar box	3/5	3/5 (60.0)
GLM	MT-ACT	Mustard bottle	2/2	2/5 (40.0)

Each object has five paired starts per configuration; every first-choice target is correct (5/5). Handover is completed/attempted; delivery is successes/5 (percent). Model names and timeout handling follow Table 3. Handover completion is a stage outcome and does not imply final delivery.

Table 12: Mean runtime on paired successful configuration episodes (seconds).

Backend	Coordinator	Paired n	Process (s)	Coord. (s)	LLM wait (s)
ManiFlow	Qwen	19	214.55	178.48	56.36
ManiFlow	GLM	19	704.34	652.35	530.55
MT-ACT	Qwen	9	186.70	160.42	57.32
MT-ACT	GLM	9	701.40	660.74	555.68

Within each backend, both coordinators use exactly the same fixtures on which both delivered successfully (19/25 ManiFlow pairs; 9/25 MT-ACT pairs). Every displayed metric covers all selected pairs. The six GLM 1,200-s coordination-budget timeouts are excluded from this success-conditioned timing subset but remain failures in Table 3; their zero LLM-wait placeholders are not treated as measurements. Process includes initialization and export; Coord. includes skill execution and LLM waiting; LLM wait includes queuing, communication, and generation. These nested columns must not be added. Results describe the evaluated implementations and do not isolate model inference speed.

A.4 FROZEN-LOG STATE–ACTION EVALUATION

The offline state–action audit uses original requests and observations from 317 valid closed-loop episodes: 195 delivery and interruption episodes, 47 held-object takeovers, and 75 sorting episodes. It makes no additional model or simulation calls. Requests are compared with the canonical action determined by the frozen execution contract and remaining attempt budget. A physical transition changes possession, support, region, receiver readiness or docking, or stable-bin state between adjacent decisions. A floor event spans consecutive floor-state decisions; its eventual response is scored before the object leaves that state. This is a behavioral evaluation of supplied feedback; no separate latent belief is recorded.

Table 13 separates decision and event denominators. Required input fields are complete in 1,971/1,971 decisions; all 57,315 visible-field comparisons agree with recorded physical facts. These check feedback propagation. After rejected requests, 96/200 next decisions are immediately state-consistent and 127/200 avoid repeating the same rejected request. Within-episode decisions are dependent, so the pooled counts are descriptive rather than independent-trial significance tests.

Table 13: **Recorded decisions reflect physical-state feedback.** Decision-level and deduplicated event-level counts are descriptive; within-episode observations are dependent.

Metric	Correct / total	Rate
All decisions	1,751/1,971	88.8%
After physical-state change	1,323/1,384	95.6%
Handover-ready decisions	280/280	100.0%
Stable-delivery decisions	176/176	100.0%
After rejection: state-consistent	96/200	48.0%
After rejection: request changed	127/200	63.5%
Floor events: first response	69/106	65.1%
Floor events: eventual response	104/106	98.1%

A.5 Implementation details and executor resource measurements

Policy inference uses 224×224 RGB images from two cameras and ten flow-integration steps. The Qwen coordinator receives a 512×384 overview image together with the structured inputs described in the method. In the primary controller comparisons, each episode permits at most 15 decisions, 12 skill dispatches, four rejected requests, and 1,200 s of coordination time. Each skill permits one outer execution attempt, except transport, which permits two. Individual model requests have a 180 s timeout, and the outer process watchdog allows 1,800 s. Frozen prompts, camera settings, normalization, and execution assistance are retained across these comparisons; model-specific settings for the configuration study are recorded separately.

The four-configuration study retains the shared 1,200 s coordination budget, 15-decision limit, 12-skill limit, and four-rejection limit. Qwen uses a 1,024-token output limit and a 180 s model-request timeout; GLM uses a 2,048-token limit and a 360 s timeout with its final-answer adapter. The outer watchdog is 1,500 s. Six original GLM task-deadline failures are retained as unsuccessful evaluated episodes, with their raw records unchanged. Only previously unattempted fixtures were continued after correcting Python exception handling. All four groups share the same 25 physical initializations; configuration and initialization hashes are archived.

We measure ManiFlow and the official OpenPI `pi05_droid` implementation of $\pi_{0.5}$ (Physical Intelligence et al., 2025) sequentially on the same Tesla V100-SXM2-32GB, with batch size one and FP32. Each configuration uses three fresh processes, each with 20 warm-up calls followed by 1,000 timed calls over ten fixed native inputs. We evaluate eager and officially compiled $\pi_{0.5}$, yielding 9,000 measured calls in total. Both policies use ten flow steps and return 15-action blocks, with their native seven- and eight-dimensional actions. Full-call latency includes input transformations, host/device transfer, synchronized policy sampling, and output transformations; robot execution and disk decoding are excluded. We record parameter counts, sampled process GPU-memory peaks, and median and 95th-percentile latency. This is a native-workload resource comparison; $\pi_{0.5}$ is not adapted to the Dingo task, so it does not provide a task-success comparison.

Table 14 reports the measured parameter counts, sampled GPU-memory peaks, and full-call latency for the native low-level inference workloads. These measurements characterize the executor deployment configuration; task-level coordination performance is evaluated separately in Section 4.

Table 14: Native-workload low-level inference cost on one V100-SXM2-32GB.

Policy configuration	Parameters (M)	GPU peak (MiB)	p50 (ms)	p95 (ms)
ManiFlow	241.70	1,434	226.98	230.42
$\pi_{0.5}$, eager	3,353.43	14,766	627.97	635.53
$\pi_{0.5}$, compiled	3,353.43	14,754	459.28	461.90

Batch size one, FP32, ten flow steps, and 15-action output blocks. Full-call latency percentiles pool 3,000 timed calls per configuration across three processes. GPU memory is the maximum sampled process peak across those runs, including loading and warm-up. Inference-relevant parameter counts exclude the unused $\pi_{0.5}$ language head, which remains in memory. Each policy uses its native inputs and action interface.

The official $\pi_{0.5}$ wrapper instantiates 3,616.76 million parameters, including a 263.32-million-parameter language head unused during action inference. We exclude that head from the inference-relevant parameter count while retaining its actual memory footprint. The three process-level latency medians are 223.73–227.88 ms for ManiFlow and 459.11–459.52 ms for compiled $\pi_{0.5}$. In a separate successful deployment probe, the Qwen–ManiFlow system contains one ManiFlow instance, approximately 9.009 billion total parameters, and a simultaneous two-GPU device-memory peak of 24.581 GiB. Together, the measurements quantify the compact executor's cost and the additional resources of the deployed coordinator.