SYNERMIND: TOWARDS A LONG-HORIZON MULTI-AGENT SYSTEM FOR EMBODIED COLLABORATION

Anonymous authors
Paper under double-blind review



Figure 1: Overview of SynerMind operating in simulated and real-world environments. The radar charts on the right compare the performance of SynerMind with existing methods across different task categories and execution metrics. SynerMind achieves the best performance in all task categories and across all execution metrics.

ABSTRACT

Multi-robot collaboration systems are essential for complex robotic tasks. While existing systems can coordinate multiple robots in short-horizon scenes, their predefined static task allocation makes them difficult to adapt to dynamic long-horizon execution. Addressing this challenge, we introduce **SynerMind**, a unified system for long-horizon collaboration among heterogeneous robots. It introduces **SynerMind-Brain**, a collaborative agentic model. The brain maintains global task awareness throughout dynamic long-horizon task execution by continuously aligning evolving task progress with heterogeneous robot capabilities. Based on this, the multi-agent framework **SynerMind-Flow** organizes perception, planning, execution, and reflection into a closed-loop collaboration process. Meanwhile, we introduce **Collaborative-State Mining Group Relative Policy Optimization** to mine difficult collaborative states from group reward statistics. It prioritizes unresolved coordination failures during policy optimization, thereby improving long-horizon collaboration. Notably, we construct ***SynerAwareness-20k*** and ***SynerAgentic-102k***, two specialized real-world datasets for long-horizon heterogeneous collaboration. Extensive experiments demonstrate that SynerMind exhibits excellent performance in simulation and achieves a ***1.53***$\times$ higher success rate than existing methods in dynamic real-world environments.

1 INTRODUCTION

Recent advances in embodied models (Zitkovich et al., 2023; Zhen et al., 2024; Mu et al., 2023; Li et al., 2025; Liang et al., 2023) have enhanced the ability of robots to understand environments, interpret instructions, and execute goal-directed actions. However, many real-world tasks (Zhang et al., 2024b; Mandi et al., 2024; Liu et al., 2025a) cannot be completed by a single robot . In long-horizon embodied collaboration (Chen et al., 2025; Chang et al., 2025; Shi et al., 2025), the system must maintain shared task awareness across multiple heterogeneous robots and coordinate decisions

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.

among different agents during execution. Therefore, building a multi-agent embodied system is not a simple extension of single-agent policies. It requires a unified collaborative intelligence that can transform global task understanding into coordinated embodied actions.

Current multi-robot systems commonly coordinate agents through communication, task planning, and role assignment. Communication-based systems (Zhang et al., 2024b;a; Mandi et al., 2024) allow agents to exchange messages after local reasoning, but the execution responsibility is usually tied to predefined agent roles. Planning-based systems (Lin et al., 2023; Hazra et al., 2024; Gupta et al., 2024; Cen et al., 2024) first decompose a high-level instruction into a sequence of subtasks, then assign these subtasks according to the initial task structure. Role-assignment systems (Chen et al., 2025; Liu et al., 2025a; Peng et al., 2026) further match agents with subtasks based on predefined capability descriptions. Although these systems (Szot et al., 2023; Chang et al., 2025) can achieve multi-robot cooperation in short-horizon tasks, they usually assume that the assigned roles and subtasks remain suitable throughout execution. However, long-horizon embodied collaboration requires dynamic task re-allocation in highly changing scenes. Existing systems usually rely on the initial static assignment, and do not fully account for changes in the scene and task state. This predefined division of labor (Liu et al., 2025a; Liang et al., 2026) limits their adaptability during execution, eventually leading to accumulated execution errors.

To address this problem, we present **SynerMind**, a long-horizon multi-agent embodied collaboration system. To accommodate the long-horizon task execution in highly dynamic environments, SynerMind continuously updates task allocation according to evolving collaborative states. We further design **SynerMind-Brain**, a shared collaborative brain. This brain maintains global task awareness across heterogeneous robots, including the states of heterogeneous robots and shared multi-view observations. Built upon this shared cognition, **SynerMind-Flow** organizes collaboration through a closed loop of planning, execution, and reflection, dynamically revising robot-skill assignments based on execution feedback. Furthermore, we propose a two-stage training paradigm. We first adopt **Awareness-to-Agency Cold Start** to establish basic collaborative awareness and agentic decision-making. We then perform **Collaborative-State Mining Group Relative Policy Optimization (CSM-GRPO)**. It estimates collaborative-state difficulty from group reward statistics, then prioritizes difficult states during subsequent policy updates. This difficulty-mining process enables SynerMind to better adapt task allocation to achieve more reliable long-horizon collaboration. Our main contributions are summarized as follows:

- We introduce **SynerMind**, a unified system for long-horizon embodied collaboration that couples the globally shared cognition of **SynerMind-Brain** with the closed-loop orchestration of **SynerMind-Flow**.
- We propose a rollout-driven **Collaborative-State Mining Training Paradigm**. It dynamically adapts the training curriculum based on mining capability of the model, enabling improvements in multi-agent reasoning, coordination, and long-horizon decision-making.
- We establish ***SynerAwareness-20k*** and ***SynerAgentic-102k***, two specialized real-world datasets to support embodied perception and task allocation training in heterogeneous multi-robot collaboration.
- Extensive experiments demonstrate the effectiveness of SynerMind across both simulated and real-world benchmarks. Our analysis highlights the contribution of the multi-agent architecture to long-horizon task execution.

2 Related Work

2.1 Embodied Systems

VLA models (Li et al., 2024a; Wang et al., 2024; Huang et al., 2024; Ma et al., 2024; Li et al., 2024b; Liu et al., 2023; Li et al., 2024d; Lin et al., 2025; Chebotar et al., 2023) have achieved promising performance in single-robot tasks. RT-2 (Zitkovich et al., 2023) transfers knowledge from vision and language to robot control through action tokens. OpenVLA (Kim et al., 2025) is introduced as an open-source model, while RDT (Liu et al., 2025b) is developed for generalizable bimanual manipulation. These advances have laid the foundation for subsequent work (Huang et al., 2023; Zhu et al., 2024; Lv et al., 2024; Ni et al., 2024; Madaan et al., 2023). However, these models mainly focus on the single robot and provide limited support for the shared task understanding and coordinated decision making required in long-horizon collaboration among multiple robots.

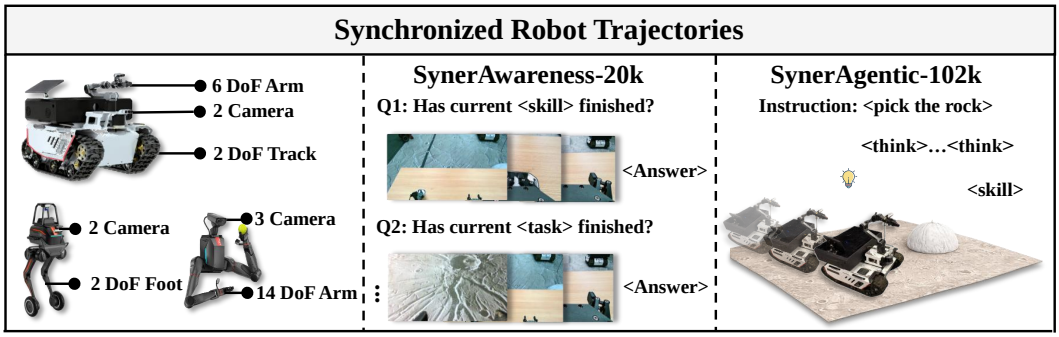


Figure 2: Overview of the real-world robots and dataset construction. *SynerAwareness-20k* is built from synchronized trajectories, while *SynerAgentic-102k* is collected from long-horizon collaboration episodes.

2.2 Multi-Robot Systems

Recent multi-robot systems (Choi et al., 2024) based on communication, task planning, and role assignment have made notable progress in complex embodied tasks. COHERENT (Liu et al., 2025a) performs centralized task planning and adjusts task allocation through the PEFA mechanism. CoELA (Zhang et al., 2024b) supports communication-based decentralized collaboration through modular perception, communication, planning, and execution. EMOS (Chen et al., 2025) performs embodiment-aware role assignment according to heterogeneous robot capabilities. Despite this progress, recent systems remain limited in developing an effective understanding of task scenes and current states of robots, hindering precise instruction generation and appropriate task allocation. Moreover, once established, static global plans cannot adapt to evolving robot states, scene conditions, or execution progress. Motivated by these limitations, SynerMind integrates a state-aware brain with an adaptive closed-loop flow for robust multi-robot collaboration in complex embodied environments.

3 SynerMind Datasets and Benchmark

To better support state awareness and task allocation in real-world long-horizon collaboration, we construct two multimodal datasets from three robots with distinct embodiments (O'Neill et al., 2024; Jiang et al., 2025), including a navigation robot, a mobile manipulation robot, and a fixed dual-arm robot, as shown in Figure 2. We further establish a long-horizon benchmark to evaluate the heterogeneous collaboration capability of embodied systems.

SynerAwareness-20k. Unified task awareness and scene understanding are essential for heterogeneous robot collaboration. Accordingly, we define four task-awareness states throughout long-horizon execution, which comprise (1) pre-execution, (2) in-execution, (3) task success, and (4) task failure. We collect multi-view observations from robots during task execution and construct more than 20***k*** samples to establish unified task-state awareness. *SynerAwareness-20**k*** organizes visual supervision around the complete collaborative workflow. Each representative image is paired with workflow context covering the global instruction, verified execution history, and current task description. Multiple binary visual questions are then generated from complementary workflow perspectives. This supervision provides the perceptual basis for collaboration planning and skill execution while supporting state verification throughout the workflow.

SynerAgentic-102k. Effective heterogeneous robot collaboration requires not just unified task awareness, but also capability-aware robot allocation under dynamic environmental changes. However, existing methods lack the capability for dynamic adjusting during long-horizon task execution. To address this, we construct an agentic reasoning dataset from real-world execution episodes. The callable skills are organized into three categories, covering (1) navigation, (2) perception, and (3) manipulation. By structuring the decision process as combinations of these skills, the dataset improves efficient collaboration while reducing invalid skill invocations. To further strengthen planning capability, we expand the reasoning distribution across different combinations of robot capabilities. Overall, SynerAgentic-102***k*** comprises a large-scale dataset of diverse decision samples, each containing an intermediate reasoning process and a final sequence of skill invocations. Construction details are provided in Appendix A.

SynerMind-Benchmark. To evaluate long-horizon collaboration among heterogeneous robots (Li et al., 2024c; Khanna et al., 2024) in real-world environments, we establish SynerMind-Benchmark. Our benchmark consists of 24 tasks categorized into four groups, including Coarse-Grained, Instruction-Constrained, Pipelined Multi-Target, and Novel Scene-Target. Each task provides multi-view visual observations and involves multiple heterogeneous robots. Additionally, we introduce a multi-dimensional evaluation protocol to assess planning consistency and task success rate.

4 SYNERMIND SYSTEM

4.1 SYSTEM OVERVIEW

SynerMind follows a skill-grounded hierarchy for long-horizon embodied collaboration. The trained SynerMind-Brain provides shared task-level cognition, while robot-specific skills support physical execution through standardized skill contracts. SynerMind instantiates the shared brain as three role-conditioned agents: the planner agent, execution agent, and reflection agent, which together form SynerMind-Flow. Formally,

$$\begin{aligned}(\mathcal{A}_\theta^P, \mathcal{A}_\theta^E, \mathcal{A}_\theta^R) &= \mathrm{RoleCondition}\big(\mathcal{B}_\theta; \psi^P, \psi^E, \psi^R\big),\\ (\Pi_t, a_t, \hat{p}_t) &= \mathcal{A}_\theta^P(I, s_t),\\ e_t &= \mathcal{A}_\theta^E(a_t, O_t^{\mathcal{R}}, s_t),\\ (q_t, y_t, h_t) &= \mathcal{A}_\theta^R(\tau_t, \hat{p}_t, s_t, \tilde{s}_t, \ell_t),\end{aligned} \tag{1}$$

where $\mathcal{B}_\theta$ is the shared brain; ψ^P, ψ^E, and ψ^R are the role-conditioning prompts; Π_t, a_t, and $\hat{p}_t$ are the workflow, robot-skill assignment, and expected post-condition; $O_t^{\mathcal{R}}$ is the multimodal observation; e_t is the execution output; $\tilde{s}_t$ and ℓ_t are the candidate post-execution state and runtime-error flag; and q_t, y_t, and h_t are the verification result, responsible agent, and corrective feedback.

4.2 SYNERMIND-BRAIN

4.2.1 SHARED BRAIN.

SynerMind-Brain serves as the shared cognitive backbone of SynerMind-Flow, maintaining identical model parameters and a unified collaborative state across the Planner, Execution, and Reflection. Role-specific prompts guide the shared brain toward distinct reasoning functions while preserving a unified collaborative policy.

Across SynerMind, the shared brain reasons from the current instruction and the latest collaborative state. Its reasoning focus evolves with workflow progression, shifting from plan formation to execution validation before reaching post-execution diagnosis. To equip this shared brain with the multimodal state understanding and adaptive decision-making capabilities required across these roles, we adopt the two-stage training paradigm described next.

4.3 COLLABORATIVE-STATE MINING TRAINING PARADIGM

To equip SynerMind-Brain with the multimodal state understanding and adaptive decision-making capabilities required throughout SynerMind-Flow, we adopt a two-stage training pipeline consisting of Awareness-to-Agency Multimodal Cold Start and CSM-GRPO.

4.3.1 AWARENESS-TO-AGENCY MULTIMODAL COLD START.

We choose Qwen3.5-9B as the base model of SynerMind-Brain to balance multimodal reasoning capability and training efficiency. The cold-start stage jointly establishes perception and decision priors through instruction tuning with SynerAwareness-20k and the verifier-curated traces in *SynerAgentic-102k*. *SynerAwareness-20k* provides binary supervision for scene understanding, robot-state recognition, and workflow-state verification, whereas SynerAgentic-102k supervises structured chain-of-thought reasoning and robot-skill-level decisions. We optimize

$$\mathcal{L}_{\mathrm{cold}} = \mathcal{L}_{\mathrm{aware}} + \lambda \mathcal{L}_{\mathrm{agent}}, \tag{2}$$

where both terms follow the standard autoregressive instruction-tuning objective. We keep the backbone largely frozen and update the visual-language projection module together with selected full-

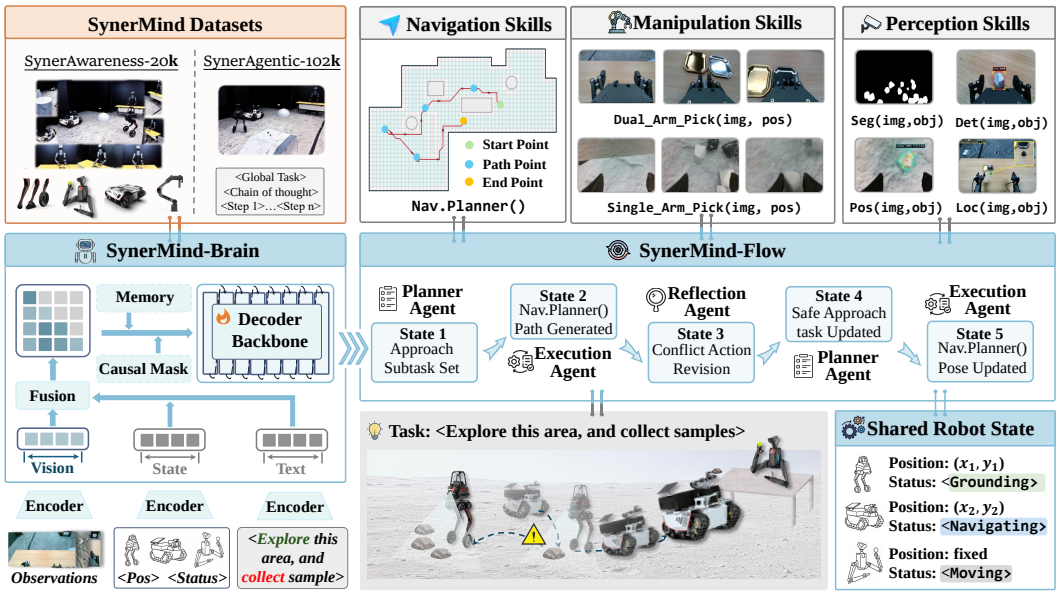


Figure 3: Overall framework of SynerMind. SynerMind-Brain fuses multimodal observations, robot states, and task instructions to maintain shared collaborative memory and generate role-conditioned decisions. SynerMind-Flow coordinates the Planner, Execution, and Reflection Agents in a closed loop, grounding plans into navigation, manipulation, and perception skills while continuously updating the shared robot state.

attention layers, enabling embodied perception alignment and workflow-aware decision reasoning without full-parameter fine-tuning.

4.3.2 CSM-GRPO

We propose CSM-GRPO to improve policy learning for long-horizon embodied collaboration. Building on the state-conditioned agentic supervision of *SynerAgentic-102k*, CSM-GRPO jointly optimizes structured reasoning and robot-skill-level decisions. It further identifies challenging collaborative states from rollout feedback and assigns them higher sampling probabilities in subsequent updates, thereby directing the shared-brain policy toward unresolved complex reasoning and persistent coordination failures.

Preliminary. We optimize the shared-brain policy at the agentic response level via CSM-GRPO. For each collaborative state s, a group of K candidate responses $\mathcal{X}(s) = \{x_i\}_{i=1}^{K}$ is sampled from the local old policy π_{old}. Each candidate x_i is evaluated by a verifier-guided reward $r_i = R_{\text{agent}}(s, x_i)$. The group-relative advantage is computed as

$$\mu_s = \frac{1}{K}\sum_{j=1}^{K} r_j, \quad \sigma_s = \sqrt{\frac{1}{K}\sum_{j=1}^{K}(r_j - \mu_s)^2},$$

$$A_i = \frac{r_i - \mu_s}{\sigma_s + \epsilon_{\text{std}}}. \tag{3}$$

The policy parameter θ is then updated by minimizing the clipped group-relative objective:

$$\begin{aligned}\mathcal{L}_{\text{CSM-GRPO}}(\theta) = -\,\mathbb{E}_{s,\, x_i \in \mathcal{X}(s)}\Big[\min\big(&\omega_i A_i,\\ &\text{clip}(\omega_i, 1-\epsilon, 1+\epsilon)A_i\big)\\ &-\beta D_{\text{KL}}\big(\pi_\theta(\cdot \mid s) \,\|\, \pi_{\text{ref}}(\cdot \mid s)\big)\Big],\end{aligned} \tag{4}$$

where $\omega_i = \pi_\theta(x_i \mid s)/\pi_{\text{old}}(x_i \mid s)$, and π_{ref} denotes the cold-start reference policy.

Reward Design for Embodied Collaboration. CSM-GRPO evaluates each structured agentic response through a joint criterion over reasoning consistency and history-grounded decision quality. For $x_i = (z_i, d_i)$, consisting of a reasoning trace z_i and a structured decision d_i, we define

$$R_{\text{agent}}(s, x_i) = \mathcal{F}_i \mathcal{U}_i^{1-\alpha} \mathcal{P}_i^{\alpha} - \lambda_{\text{f}}(1 - \mathcal{F}_i) - \lambda_{\text{r}} \mathcal{C}_i, \tag{5}$$

where all four components are computed using fixed deterministic rules. Specifically, $\mathcal{P}_i \in [0, 1]$ is the normalized proportion of satisfied consistency checks, evaluating whether the reasoning is grounded in the observed scene and robot states, respects workflow dependencies, and supports the final robot-skill assignment. $\mathcal{F}_i \in \{0, 1\}$ denotes decision feasibility and is set to one only when the structured decision is correctly parsed, the selected robots and skills are valid, and their execution preconditions are satisfied. $\mathcal{U}_i \in [0, 1]$ measures contribution to the current task frontier, defined by unfinished workflow nodes whose predecessors have been verified as complete. $\mathcal{C}_i \in [0, 1]$ penalizes contradictions with verified history, such as repeating completed subtasks or conflicting with confirmed robot and object states.

The multiplicative term jointly rewards responses that are feasible, task-relevant, and reasoning-consistent. When $\mathcal{F}_i = 1$, increasing either $\mathcal{P}_i$ or $\mathcal{U}_i$ directly raises the reward and the corresponding group-relative advantage, while decreasing either suppresses the positive term. When $\mathcal{F}_i = 0$, the positive term vanishes and the infeasibility penalty applies. Here, $\mathcal{F}_i$ evaluates immediate executability, whereas $\mathcal{C}_i$ captures temporal inconsistency across execution rounds. All signals are computed before execution from the collaborative state, workflow, skill contracts, and verified history. Malformed or incomplete responses receive $\mathcal{P}_i = \mathcal{U}_i = \mathcal{F}_i = 0$. The scoring rules remain fixed throughout training, with their exact definitions provided in Appendix B.

Collaborative-State Mining. CSM-GRPO prioritizes collaborative states for which the current policy produces poor or unstable reasoning-and-decision responses. For each candidate group $\mathcal{X}(s)$, we summarize its normalized reward distribution as

$$\begin{aligned} \overline{R}(s) &= \frac{1}{K} \sum_{i=1}^{K} \widetilde{R}_{\text{agent}}(s, x_i), \\ \sigma_R(s) &= \sqrt{\frac{1}{K} \sum_{i=1}^{K} \left(\widetilde{R}_{\text{agent}}(s, x_i) - \overline{R}(s) \right)^2}, \end{aligned} \tag{6}$$

where $\widetilde{R}_{\text{agent}} \in [0, 1]$ is the normalized reward in Eq. 5. The resulting state difficulty is

$$\delta(s) = \text{clip}\left(1 - \overline{R}(s) + \kappa \sigma_R(s), 0, 1\right). \tag{7}$$

The first term captures states that remain difficult on average, whereas the second identifies unstable policy behavior across sampled responses. We sample subsequent training states from

$$p_{\text{train}}(s) = (1 - \rho) p_{\mathcal{B}}(s) + \rho \frac{\exp(\delta(s)/\tau_{\text{temp}})}{\sum_{s' \in \mathcal{B}} \exp(\delta(s')/\tau_{\text{temp}})}, \tag{8}$$

where ρ is the mining-mixture coefficient, τ_{temp} is the sampling temperature, and $\mathcal{B}$ is the state buffer. This distribution preserves state-buffer coverage while concentrating later policy updates on unresolved collaborative states.

4.4 SynerMind-Flow

Within SynerMind-Flow, the Planner Agent maps task requirements and current collaborative states into robot-skill assignments under the corresponding skill constraints. Building on these assignments, the Execution Agent grounds high-level decisions into executable skill invocations, bridging strategic decisions with embodied physical execution. The Reflection Agent verifies the resulting state transitions and generates corrective signals for subsequent decisions. Together, the three agents assume complementary roles around a unified skill representation, allowing the high-level collaboration mechanism to adapt across different robot embodiments and skill repertoires.

4.4.1 Planner Agent.

To transform coarse user instructions into clear and structured collaboration workflows, we formulate the Planner Agent as a state-conditioned planning module with adaptive revision capability.

Table 1: Simulation results across five simulation scenes. The best results are highlighted in bold.

Method	S1		S2		S3		S4		S5		Average	
	SR(%)	AS	SR(%)	AS	SR(%)	AS	SR(%)	AS	SR(%)	AS	SR(%)	AS
PEFA (Liu et al., 2025a)	35.0	17.95	25.0	19.50	35.0	17.55	15.0	19.25	0.0	18.00	22.0	18.45
Prim-MCTS (Browne et al., 2012)	30.0	16.00	15.0	19.90	5.0	16.35	5.0	18.35	0.0	18.00	11.0	17.72
LLM-MCTS (Zhao et al., 2023)	45.0	14.85	45.0	16.75	40.0	14.90	35.0	16.30	45.0	15.25	42.0	15.61
CMRS (Huang et al., 2022)	75.0	12.80	80.0	13.05	80.0	9.75	70.0	12.80	65.0	13.50	74.0	12.38
DMRS-1D (Zhang et al., 2024b)	30.0	15.95	35.0	18.20	35.0	14.15	40.0	16.00	10.0	17.55	30.0	16.37
DMRS-2D (Zhang et al., 2024b)	30.0	16.40	25.0	19.15	40.0	13.70	35.0	16.15	15.0	17.05	29.0	16.49
SynerMind	**80.0**	**11.35**	**100.0**	**10.05**	**100.0**	**8.10**	**100.0**	**9.05**	**80.0**	**12.60**	**92.0**	**10.23**

Table 2: Success rates of different methods across 24 real-world heterogeneous robot collaboration tasks. Each entry reports the number of successful trials over the total number of trials. Class I–IV represent four task categories.

Class I: Coarse-Grained Tasks								Class II: Instruction-Constrained Tasks						
Method	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**Avg. (%)**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**Avg. (%)**
PEFA	11/20	7/20	6/20	8/20	10/20	10/20	43.3	9/20	7/20	10/20	8/20	8/20	8/20	41.7
CMRS	15/20	11/20	10/20	10/20	16/20	16/20	65.0	13/20	13/20	14/20	12/20	12/20	10/20	61.7
SynerMind	**20/20**	**17/20**	**17/20**	**16/20**	**20/20**	**20/20**	**91.7**	**19/20**	**19/20**	**20/20**	**20/20**	**18/20**	**18/20**	**95.0**
Class III: Pipelined Multi-Target Tasks								**Class IV: Novel Scene-Target Tasks**						
Method	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**Avg. (%)**	**T19**	**T20**	**T21**	**T22**	**T23**	**T24**	**Avg. (%)**
PEFA	9/20	7/20	8/20	8/20	6/20	8/20	38.3	7/20	7/20	8/20	4/20	8/20	8/20	35.0
CMRS	13/20	11/20	12/20	12/20	10/20	12/20	58.3	11/20	11/20	12/20	8/20	12/20	12/20	55.0
SynerMind	**20/20**	**17/20**	**18/20**	**19/20**	**16/20**	**20/20**	**91.7**	**20/20**	**17/20**	**17/20**	**14/20**	**20/20**	**18/20**	**88.3**

Given the current feasible skill space, the Planner Agent predicts and continuously updates a structured workflow Π_t conditioned on the user instruction I and collaborative state s_t, with reflective feedback. Each subtask is decoded as a robot-skill-level assignment. The unified mechanism supports both workflow initialization and runtime revision: it generates an initial workflow Π_0 in the absence of a prior plan, while error-specific feedback from the Reflection Agent guides selective updates to erroneous robot-skill assignments without altering verified progress or unaffected subtasks.

4.4.2 Execution Agent.

To ensure that planned instructions are faithfully translated into physical robot execution, we formulate the Execution Agent as an action grounding module between SynerMind-Flow and the heterogeneous robots. Receiving the current task assignment in workflow Π_t, the Execution Agent parses the assigned subtask to identify the designated robot and skill, then predicts structured execution parameters. The resulting skill call is grounded into a robot-executable command c_t through a unified skill adapter and dispatched to the corresponding robot. The executable skill library comprises three categories: navigation, perception, and manipulation, with detailed skill descriptions provided in Appendix D. Furthermore, multimodal execution evidence and runtime feedback are integrated with the Planner assignment into an execution trace τ_t for subsequent reflection and state updates. The complete pseudocode is provided in Appendix C.

4.4.3 Reflection Agent.

To maintain consistency between skill-grounded assignments and physical execution throughout long-horizon collaboration, we formulate the Reflection Agent as a verifier-guided state assessment module in SynerMind-Flow. Given the execution trace τ_t, the Reflection Agent infers collaborative-state transitions from execution evidence and detects execution anomalies, treating timeouts as failures. Once either condition is identified, the agent verifies the resulting execution state.

Let $\tilde{s}_t$ denote the post-execution collaborative state inferred from τ_t, $\hat{p}_t$ the expected postcondition of the invoked skill, and ℓ_t the runtime error indicator. The verification result q_t is defined as

$$q_t = \mathbb{I}\left[\tilde{s}_t \models \hat{p}_t \wedge \ell_t = 0\right]. \tag{9}$$

When $q_t = 1$, the verified progress is integrated into s_{t+1}. Otherwise, failed verification activates failure attribution to identify the responsible agent $y_t \in \{\mathrm{P}, \mathrm{E}\}$ and the corresponding failure cause ξ_t. Conditioned on y_t and ξ_t, the Reflection Agent generates a targeted corrective signal h_t, prompting the responsible agent to re-examine and correct its previous action. The diagnosis and corrective instruction are then incorporated into the collaborative context, while subsequent verified evidence updates the shared state, completing the iterative loop across planning, execution, and reflection.

5 Experiments

5.1 Implementation Details

We conduct SynerMind training and inference in simulation and real-world settings on an NVIDIA A100 GPU. For simulation evaluation, we report Success Rate (SR) and Average Steps (AS) to measure task completion reliability and execution efficiency. For SynerMind-Brain evaluation, we report ROUGE-L (R-L) and structured agentic decision accuracy (Acc.). R-L measures similarity between decisions and reference responses, while Acc. measures executable decision correctness.

5.2 System Performance

5.2.1 Performance on Simulation Tasks.

We first evaluate the skill-grounded collaboration mechanism of SynerMind-Flow in simulation. To isolate the workflow contribution from brain-specific training, we adopt an officially multimodal LLM as the shared SynerMind-Brain and evaluate our workflow on the COHERENT simulation benchmark. We compare SynerMind with communication, task planning, and role assignment baselines. Details of the compared methods are provided in Appendix F.

Experimental Results. As reported in Table 1, SynerMind attains the highest average SR of *92.0%* and the lowest average AS of *10.23* across the five simulation scenes. Relative to CMRS, the strongest baseline, SynerMind improves SR by *18.0%* while reducing the average execution length by *2.15* steps. It further achieves a *100.0%* success rate in S2–S4 and maintains *80.0%* in S1 and S5. The consistent improvements in both metrics demonstrate that SynerMind-Flow enhances task completion reliability while reducing redundant execution.

5.2.2 Performance on Real-World Tasks.

We evaluate the complete SynerMind system on *24* real-world heterogeneous robot collaboration tasks. Each task requires joint reasoning over robot capabilities, task dependencies, and execution feedback to maintain a coherent long-horizon workflow. We conduct extensive real-robot trials for meta-action execution and long-horizon collaboration, with detailed task definitions provided in Appendix G.

Experimental Results. We select CMRS and PEFA as representative baselines for centralized multi-robot planning and role-assignment hierarchical collaboration, respectively. As reported in Table 2, SynerMind achieves an overall success rate of *91.7%*, substantially exceeding CMRS and PEFA by *31.7%* and *52.1%*. Its category-level success rates remain within *88.3%–95.0%*, indicating only limited performance degradation as execution constraints or scene uncertainty increase. The gains remain pronounced on instruction-constrained and pipelined multi-target tasks, where successful completion requires persistent maintenance of workflow dependencies and collaborative states across successive robot stages. These results demonstrate the effectiveness of shared task cognition and reflective correction for long-horizon physical collaboration.

5.3 Ablation Study

5.3.1 Ablation Results of SynerMind-Brain.

To validate the effectiveness of our two-stage training strategy, we compare representative multimodal baselines (Lu et al., 2024) with different training variants of SynerMind-Brain. As shown in Table 4, all baseline models (Lu et al., 2024; Qwen Team, 2026), including DeepSeekVL-7B and Gemini3.0-Flash, perform clearly below the cold-start model in decision accuracy. The cold-start gains reflect progressive improvements in state understanding, workflow verification and collaborative decision-making. Starting from the same cold-start model, standard GRPO improves the

Table 3: Ablation results of different reward variants under the same CSM-GRPO setting.

Reward Variant	Class I		Class II		Class III		Class IV	
	R-L	Acc.	R-L	Acc.	R-L	Acc.	R-L	Acc.
w/o $\mathcal{P}_i$	68.4	79.6	67.9	78.8	65.8	75.9	63.0	73.7
w/o $\mathcal{U}_i$	69.5	80.4	65.7	76.8	63.2	73.5	65.1	75.2
w/o $\mathcal{F}_i$	65.3	75.8	63.8	74.1	60.7	70.2	58.9	68.7
w/o $\mathcal{C}_i$	70.7	81.6	68.8	79.7	64.9	75.0	65.4	75.8
SynerMind-Reward	**72.5**	**84.0**	**71.8**	**82.5**	**70.1**	**80.8**	**69.4**	**79.6**

Table 4: Ablation results of the training strategy for SynerMind-Brain. "+ Cold Start" denotes the first training stage, while standard GRPO and CSM-GRPO are alternative second-stage variants initialized from the cold-start model.

Model	Class I		Class II		Class III		Class IV	
	R-L	Acc.	R-L	Acc.	R-L	Acc.	R-L	Acc.
DeepSeekVL-7B	39.8	46.7	38.1	44.9	34.8	41.0	32.6	38.5
Gemini3.0-Flash	44.1	51.6	42.7	49.5	39.8	46.1	37.5	43.7
Qwen3.5-9B	43.2	50.5	41.8	48.6	38.9	45.0	36.7	42.8
+ Cold Start	58.6	66.8	57.1	65.4	54.0	62.7	52.4	60.3
+ GRPO	64.7	74.2	63.8	73.1	60.5	70.6	59.1	69.2
+ CSM-GRPO	**72.5**	**84.0**	**71.8**	**82.5**	**70.1**	**80.8**	**69.4**	**79.6**

Table 5: Ablation results of the complete workflow on representative real-world heterogeneous robot collaboration tasks, where selected tasks cover different task settings.

Method	Task1	Task2	Task7	Task8
Base-Flow	9/20	8/20	11/20	12/20
w/o Reflection	15/20	12/20	15/20	16/20
SynerMind-Flow	**20/20**	**17/20**	**19/20**	**19/20**
Method	**Task13**	**Task14**	**Task19**	**Task20**
Base-Flow	7/20	7/20	4/20	6/20
w/o Reflection	12/20	11/20	9/20	8/20
SynerMind-Flow	**20/20**	**17/20**	**20/20**	**17/20**

average decision accuracy by *8.0%*, validating group-relative optimization under the joint process-decision reward. CSM-GRPO achieves the best overall performance, outperforming standard GRPO by *9.9%*. This further gain indicates that difficulty-aware resampling based on group reward statistics improves task allocation capability in complex collaborative states.

5.3.2 Ablation Results of Reward-Design.

To evaluate the contribution of each reward component, we remove $\mathcal{P}_i, \mathcal{U}_i, \mathcal{F}_i$, or $\mathcal{C}_i$ while retaining the remaining CSM-GRPO settings. As shown in Table 3, SynerMind-Reward consistently achieves the best R-L and accuracy across all four categories. Removing $\mathcal{F}_i$ causes the most pronounced degradation, indicating the importance of feasibility-aware supervision. Detailed removal strategies are provided in the Appendix B. These results demonstrate the effectiveness of our coupled reward.

5.3.3 Ablation Results of SynerMind-Flow.

To evaluate the contribution of the complete workflow and the Reflection Agent, we compare SynerMind-Flow with Base-Flow and w/o Reflection. Base-Flow follows a hierarchical brain-cerebellum paradigm, in which a high-level brain issues abstract commands to low-level embodied executors. Details are provided in Appendix H. As shown in Table 5, SynerMind-Flow achieves an overall success rate of *93.1%*, exceeding Base-Flow and w/o Reflection by *53.1* and *31.9* percentage points. The improvement is more pronounced on pipelined multi-target tasks and novel scene and target tasks, where execution dependencies and state uncertainty can propagate across successive robot stages. These results demonstrate that the Planner and Execution Agents strengthen task execution and allocation, while targeted corrective feedback suppresses error accumulation and improves recovery from local failures during long-horizon heterogeneous robot collaboration.

6 Conclusion

We present SynerMind, a multi-agent system for long-horizon collaboration, including SynerMind-Brain and SynerMind-Flow. By refining dynamic task allocation through reflective feedback, SynerMind demonstrates superior performance in dynamic environments. Besides, we construct SynerAwareness-20k and SynerAgentic-102k, two specialized real-world datasets for heterogeneous robot collaboration. Furthermore, We propose CSM-GRPO, dynamically adapting the training curriculum based on group reward statistics. Real-world and simulation evaluations demonstrate SynerMind achieves superior performance, providing reliable guarantees for long-horizon collaborative tasks.

AI USE STATEMENT

Generative AI tools were used for language polishing, grammatical revision, and LaTeX formatting during manuscript preparation. Large language models also assisted in dataset construction by refining question wording, generating candidate answers and intermediate reasoning text, and producing synthetic reasoning–skill-sequence samples, as detailed in Appendix A. Human experts reviewed the generated content and corrected identified errors. The authors take full responsibility for the datasets, reported results, and final manuscript.

REFERENCES

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games (TCIAIG)*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

Jun Cen, Chenfei Wu, Xiao Liu, Shengming Yin, Yixuan Pei, Jinglong Yang, Qifeng Chen, Nan Duan, and Jianguo Zhang. Using left and right brains together: Towards vision and language planning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 5982–6001, 2024. URL https://proceedings.mlr.press/v235/cen24a.html.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, et al. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 65205–65268, 2025.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning (CoRL)*, pp. 3909–3928, 2023.

Junting Chen, Checheng Yu, Xunzhe Zhou, Tianqi Xu, Yao Mu, Mengkang Hu, Wenqi Shao, Yikai Wang, Guohao Li, and Lin Shao. Emos: Embodiment-aware heterogeneous multi-robot operating system with llm agents. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 56670–56690, 2025.

Wonje Choi, Woo Kyung Kim, Minjong Yoo, and Honguk Woo. Embodied CoT distillation from LLM to off-the-shelf agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 8702–8721, 2024. URL https://proceedings.mlr.press/v235/choi24d.html.

Jigyasa Gupta, Shreya Sharma, Shreshth Tuli, Rohan Paul, et al. Goalnet: Interleaving neural goal predicate inference with classical planning for generalization in robot instruction following. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 20113–20122, 2024.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. Saycanpay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 20123–20133, 2024.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3D world. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 20413–20451, 2024. URL https://proceedings.mlr.press/v235/huang24ae.html.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/huang22a.html.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. Grounded decoding: Guiding text generation with grounded models for embodied agents. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:59636–59661, 2023.

Guangqi Jiang, Yifei Sun, Tao Huang, Huanyu Li, Yongyuan Liang, and Huazhe Xu. Robots pre-train robots: Manipulation-centric robotic representation from large-scale robot datasets. In *The Thirteenth International Conference on Learning Representations (ICLR)*, volume 2025, pp. 81885–81905, 2025.

Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. Goat-bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16373–16383, 2024.

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 (CoRL)*, pp. 2679–2713, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Daixun Li, Yusi Zhang, Mingxiang Cao, Donglai Liu, Weiying Xie, Tianlin Hui, Lunkai Lin, Zhiqiang Xie, and Yunsong Li. Towards long-horizon vision-language-action system: Reasoning, acting and memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6839–6848, 2025.

Jiachen Li, Qiaozi Gao, Michael Johnston, Xiaofeng Gao, Xuehai He, Hangjie Shi, Suhaila Shakiah, Reza Ghanadan, and William Yang Wang. Mastering robot manipulation with multimodal prompts through pretraining and multi-task fine-tuning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 27822–27845, 2024a. URL https://proceedings.mlr.press/v235/li24x.html.

Jianxiong Li, Jinliang Zheng, Yinan Zheng, Liyuan Mao, Xiao Hu, Sijie Cheng, Haoyi Niu, Jihao Liu, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. DecisionNCE: Embodied multimodal representations via implicit preference learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 29461–29488, 2024b. URL https://proceedings.mlr.press/v235/li24cr.html.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 100428–100534, 2024c.

Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18061–18070, 2024d.

Dingkang Liang, Cheng Zhang, Xiaopeng Xu, Jianzhong Ju, Zhenbo Luo, and Xiang Bai. Cook and clean together: Teaching embodied agents for parallel task execution. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 18415–18424, 2026.

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. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. On grounded planning for embodied tasks with language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pp. 13192–13200, 2023.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 54877–54910, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:34892–34916, 2023.

Kehui Liu, Zixin Tang, Dong Wang, Zhigang Wang, Xuelong Li, and Bin Zhao. Coherent: Collaboration of heterogeneous multi-robot system with large language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10208–10214, 2025a.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 29982–30009, 2025b.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024.

Qi Lv, Hao Li, Xiang Deng, Rui Shao, Michael Y Wang, and Liqiang Nie. RoboMP^2: A robotic multimodal perception-planning framework with multimodal large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 33558–33574, 2024. URL https://proceedings.mlr.press/v235/lv24a.html.

Xiao Ma, Sumit Patidar, Iain Haughton, and Stephen James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18081–18090, June 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:46534–46594, 2023.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299, 2024.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:25081–25094, 2023.

Fei Ni, Jianye Hao, Shiguang Wu, Longxin Kou, Jiashun Liu, Yan Zheng, Bin Wang, and Yuzheng Zhuang. Generate subgoal images before act: Unlocking the chain-of-thought reasoning in diffusion model for robot manipulation with multimodal prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13991–14000, June 2024.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024.

Bo Peng, Pi Bu, Keyu Pan, Xinrun Xu, Yingxiu Zhao, Miao Chen, Yang Du, Lin Li, Jun Song, and Tong Xu. How foundational skills influence vlm-based embodied agents: A native perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 8322–8330, 2026.

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

Haojun Shi, Suyu Ye, Xinyu Fang, Chuanyang Jin, Leyla Isik, Yen-Ling Kuo, and Tianmin Shu. Muma-tom: Multi-modal multi-agent theory of mind. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, pp. 1510–1519, 2025.

Andrew Szot, Unnat Jain, Dhruv Batra, Zsolt Kira, Ruta Desai, and Akshara Rai. Adaptive coordination in social embodied rearrangement. In *International Conference on Machine Learning (ICML)*, pp. 33365–33380, 2023.

Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, et al. Embodiedscan: A holistic multi-modal 3d perception suite towards embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19757–19767, 2024.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. Proagent: Building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 17591–17599, 2024a.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 19373–19401, 2024b.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36: 31967–31987, 2023.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 61229–61245, 2024. URL https://proceedings.mlr.press/v235/zhen24a.html.

Yichen Zhu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Retrieval-augmented embodied agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17985–17995, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, pp. 2165–2183, 2023.

A Dataset Construction

Both SynerAwareness-20k and SynerAgentic-102k are constructed from the same collection of real-world robot execution trajectories. During collection, human operators selected skills, such as navigation and grasping, which were executed by the robots' underlying controllers. Multimodal sensor data, including LiDAR point clouds, RGB-D images, and IMU measurements, were recorded at 1 Hz.

For SynerAwareness-20k, human annotators designed question categories and templates for embodied scene understanding and task-state assessment. GPT-5.2 refined the question wording and generated binary answers conditioned on a single RGB image, the global task instruction, the current task description, and task descriptions from several preceding steps retrieved from execution logs. Every question-answer pair was manually checked for question validity and consistency between the answer and the observations, and erroneous annotations were corrected. The resulting dataset contains 20k question-answer pairs.

SynerAgentic-102k combines samples organized from recorded execution trajectories with synthetic samples that extend the coverage of robot capability combinations. For trajectory-derived samples, reference skill sequences were first organized from the recorded executions and then provided to GPT-5.2 to generate intermediate reasoning text. For synthetic samples, GPT-5.2 jointly generated reasoning text and skill sequences under expanded robot capability configurations, conditioned on images, task instructions, and execution history. The dataset contains 102k samples after expansion. Annotation quality control for both datasets comprised programmatic format checks, followed by model-based answer review and subsequent human review. Human experts evaluated GPT-generated answers, reasoning text, and skill sequences against the available observations, task context, and robot capabilities, and corrected identified errors.

B Reward Design Details

For a collaborative state s, each sampled response is written as $x_i = (z_i, d_i)$, where z_i is the reasoning trace and d_i is the structured robot-skill decision. The reward is computed before execution using the current observations, robot states, workflow, skill contracts, and verified history. Let $\mathbb{I}[\cdot]$ denote the indicator function.

Reasoning consistency. Let $\mathcal{Q}_i^P$ be the applicable consistency checks for x_i. These checks determine whether the reasoning is grounded in the observed scene and robot states, respects workflow dependencies and task constraints, and supports the final robot-skill assignment. We define

$$P_i = \frac{1}{|\mathcal{Q}_i^P|} \sum_{q \in \mathcal{Q}_i^P} \mathbb{I}[q(s, z_i, d_i)],$$

so $P_i \in [0, 1]$ is the proportion of satisfied checks.

Decision feasibility. Let $r(d_i)$, $k(d_i)$, and $\eta(d_i)$ denote the selected robot, skill, and arguments. Feasibility is a hard gate,

$$F_i = \mathbb{I}[\mathrm{Parse}(d_i)]\mathbb{I}[r(d_i) \in \mathcal{R}]\mathbb{I}[k(d_i) \in \mathcal{K}_{r(d_i)}]\mathbb{I}[\mathrm{Valid}(\eta(d_i))]\mathbb{I}[s \models \mathrm{Pre}_{k(d_i)}].$$

Thus, $F_i = 1$ only when the decision is correctly parsed, the robot-skill pair and arguments are valid, and the corresponding skill preconditions are satisfied.

Task-frontier contribution. Let $\Pi_t = (V, E)$ be the current workflow and $\chi_t(v)$ the verified status of node v. The task frontier is

$$\mathcal{V}_t^{\mathrm{front}} = \{v \in V \mid \chi_t(v) = \mathrm{unfinished},\ \forall u \in \mathrm{Pred}(v),\ \chi_t(u) = \mathrm{verified}\}.$$

For each frontier node v, let $\mathcal{G}(v)$ contain its required subtask attributes and postconditions, including the target, destination, and ordering constraints when applicable. We compute

$$U_i = \max_{v \in \mathcal{V}_t^{\mathrm{front}}} \frac{1}{|\mathcal{G}(v)|} \sum_{g \in \mathcal{G}(v)} \mathbb{I}[d_i \models g].$$

Therefore, $U_i \in [0, 1]$ measures how well the decision advances a currently executable workflow node. We set $U_i = 0$ when no frontier node is matched.

Verified-history contradiction. Let $\mathcal{Q}_i^C$ contain checks for contradictions with verified history, including repeating completed subtasks, violating a confirmed execution order, and conflicting with confirmed robot or object states. We define

$$C_i = \frac{1}{|\mathcal{Q}_i^C|} \sum_{q \in \mathcal{Q}_i^C} \mathbb{I}[q \text{ is contradicted by } x_i],$$

with $C_i = 0$ when no contradiction is present.

Combining the four components, the response reward is

$$R_{\text{agent}}(s, x_i) = F_i U_i^{1-\alpha} P_i^{\alpha} - \lambda_f (1 - F_i) - \lambda_r C_i.$$

The geometric coupling gives positive credit only when the response is both reasoning-consistent and relevant to the current task frontier. The feasibility gate removes positive credit from invalid decisions, while C_i penalizes temporal inconsistency across execution rounds. Malformed or incomplete responses are assigned $P_i = U_i = F_i = 0$. For collaborative-state mining, the raw reward is mapped to $[0, 1]$ using its fixed range:

$$\widetilde{R}_{\text{agent}}(s, x_i) = \text{clip}\left(\frac{R_{\text{agent}}(s, x_i) + \lambda_f + \lambda_r}{1 + \lambda_f + \lambda_r}, 0, 1 \right).$$

Reward ablations. All variants in Table 3 of the main paper retain the same CSM-GRPO setting, cold-start checkpoint, training data, rollout group size, optimization schedule, and inference settings. Only the reward formulation is changed:

$$\begin{aligned}
R_{\text{w/o } P} &= F_i U_i - \lambda_f (1 - F_i) - \lambda_r C_i, \\
R_{\text{w/o } U} &= F_i P_i - \lambda_f (1 - F_i) - \lambda_r C_i, \\
R_{\text{w/o } F} &= U_i^{1-\alpha} P_i^{\alpha} - \lambda_r C_i, \\
R_{\text{w/o } C} &= F_i U_i^{1-\alpha} P_i^{\alpha} - \lambda_f (1 - F_i).
\end{aligned}$$

These variants respectively remove state-grounded reasoning consistency, task-frontier contribution, decision feasibility, and verified-history consistency.

C Complete Process of SynerMind-Flow

The complete planning, execution, and reflection procedure of SynerMind-Flow is summarized in Algorithm 1.

Algorithm 1: Complete Process of SynerMind-Flow

Require: Instruction I, initial state s_0, robots $\mathcal{R}$, skills $\mathcal{K}$, maximum planning rounds B
Ensure: Task status and execution history $\mathcal{T}$

$s \leftarrow s_0$, $\Pi \leftarrow \emptyset$, $h \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$
for $t = 1$ to B **do**
  if $\text{GoalReached}(s)$ **then return** success, $\mathcal{T}$
  Planner: $(\Pi, a, \hat{p}) \leftarrow \mathcal{A}_\theta^P(I, s, \Pi, h)$; $h \leftarrow \emptyset$
  repeat
    Execution: $c \leftarrow \mathcal{G}(a, s, \mathcal{K}, \mathcal{R}, h)$
    Dispatch c and accumulate execution trace τ
    repeat Read candidate state $\tilde{s}$ and flag ℓ from τ **until** $\text{Changed}(s, \tilde{s}) \vee \ell = 1$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau\}$; **Reflection:** $(q, y, h) \leftarrow \mathcal{A}_\theta^R(\tau, \hat{p}, s, \tilde{s}, \ell)$
    if $q = 1$ **then**
      $s \leftarrow \tilde{s}$ and mark a as verified in Π; $h \leftarrow \emptyset$
    else
      Route feedback h to $y \in \{\text{P}, \text{E}\}$
    end if
  until $q = 1 \vee y = \text{P}$
end for
return failure, $\mathcal{T}$

D ROBOT SKILL LIBRARY

SynerMind coordinates three heterogeneous embodiments through a unified robot-skill interface, including a navigation robot, a mobile manipulation robot, and a fixed dual-arm robot. For robot r, its executable task-skill set is denoted by

$$\mathcal{K}_r = \mathcal{K}_r^{\text{nav}} \cup \mathcal{K}_r^{\text{per}} \cup \mathcal{K}_r^{\text{man}}, \tag{10}$$

where the three subsets contain the task-oriented skills for navigation, perception, and manipulation, respectively. Initialization, posture transition, handover preparation, reset, and execution recovery are controller-level routines invoked internally during skill execution. They are not exposed as independent Planner-callable skills and are not additional members of $\mathcal{K}_r$.

Each skill is represented by a structured contract

$$k = \langle \text{Pre}_k, \text{Post}_k, \text{Cost}_k \rangle , \tag{11}$$

where Pre_k and Post_k denote its required and expected collaborative states, while Cost_k provides an execution-cost estimate. The structured arguments α_k are supplied when the skill is invoked rather than stored as part of the skill contract. The Planner Agent selects a robot-skill assignment from the corresponding admissible set, and the Execution Agent grounds the selected skill and arguments into the associated robot controller.

The primary skills are grouped according to their functional role and the collaborative-state transition they produce. Internal sensing, localization, or control modules do not alter the primary category of a skill. For example, autonomous exploration uses multimodal observations but is categorized as navigation because its primary output is an updated traversable map. Object collection invokes close-range visual grounding but is categorized as manipulation because its expected postcondition requires the robot to hold the assigned target.

D.1 NAVIGATION SKILLS

Navigation skills modify the spatial state of a robot or construct the shared spatial representation required for subsequent collaboration. They are instantiated on the navigation robot and the mobile manipulation robot.

Autonomous Exploration and Mapping. This skill is deployed on the navigation robot to construct the shared spatial representation of the workspace. Starting from the current robot pose p_t, the robot observes a local exploration region

$$\mathcal{B}_t = \{p \mid \|p - p_t\|_2 \leq 5\,\text{m}\} . \tag{12}$$

LiDAR and RGB-D observations are integrated to distinguish free, occupied, and unexplored space. The candidate frontier set is defined as

$$\mathcal{F}_t = \partial\mathcal{M}_t^{\text{free}} \cap \partial\mathcal{M}_t^{\text{unknown}}, \tag{13}$$

where each frontier connects the currently traversable region with unexplored space. The next exploration target is selected by

$$f_t^* = \arg\max_{f \in \mathcal{F}_t} \left[G(f) - \eta d(p_t, f) \right] , \tag{14}$$

where $G(f)$ measures the expected observable area around frontier f, while $d(p_t, f)$ denotes the collision-free path length from the current pose.

After reaching f_t^*, the robot updates the local map and continues exploration from the new pose. The process terminates when no reachable frontier remains. The accumulated spatial representation is converted into a two-dimensional occupancy grid $\mathcal{M}$ and shared with the mobile manipulation robot. The postcondition requires the explored region to be updated and a valid occupancy grid to be available in the collaborative state.

Goal-Conditioned Navigation. This skill enables the navigation robot to move toward a relative destination. Given $\Delta p = (\Delta x, \Delta y, \Delta\theta)$, the target pose in the map frame is

$$p_g = \hat{p}_t \oplus \Delta p, \tag{15}$$

where $\hat{p}_t$ denotes the current AMCL pose estimate and $\oplus$ denotes planar pose composition. AMCL estimates the localization posterior

$$p\left(p_t \mid z_{1:t}, u_{1:t}, \mathcal{M}\right) \tag{16}$$

from the sensor observations, motion inputs, and occupancy grid. The target pose is then published through ROS, while the `move_base` stack generates a collision-aware trajectory. The postcondition is satisfied when the robot reaches the target region within the predefined position and orientation tolerances.

Map-Conditioned Navigation. This skill is deployed on the mobile manipulation robot. The robot receives the occupancy grid $\mathcal{M}$ constructed by the navigation robot and localizes itself in the shared coordinate system through AMCL. Given a collection, waypoint, handover, or delivery pose p_g, `move_base` generates a collision-free trajectory over the shared map.

When the robot is carrying an object, the manipulator maintains the predefined posture q_{carry} throughout base motion. The postcondition requires the mobile manipulation robot to reach the assigned region while preserving its current object-carrying state. For instruction-constrained tasks, the specified map-frame waypoint is treated as an intermediate postcondition before the subsequent navigation skill is activated.

D.2 Perception Skills

Perception skills convert multimodal observations and textual target descriptions into spatial information required by subsequent navigation or manipulation. They provide coarse map-level localization and close-range manipulation grounding.

Open-Vocabulary Target Localization. This skill is deployed on the navigation robot to estimate the coarse location of a target described by language. YOLO-World detects the target according to its textual description. Given the detected image coordinate (u, v) and the associated depth $d(u, v)$, the target point in the camera frame is obtained by

$${}^{c}p = d(u,v)K^{-1}\begin{bmatrix} u & v & 1 \end{bmatrix}^{\top}, \tag{17}$$

where K denotes the camera intrinsic matrix. The corresponding point in the map frame is

$$\begin{bmatrix} {}^{m}p \\ 1 \end{bmatrix} = {}^{m}T_c \begin{bmatrix} {}^{c}p \\ 1 \end{bmatrix}, \tag{18}$$

where ${}^{a}T_b$ transforms a point from frame b to frame a.

This skill provides a coarse map-level target region rather than a manipulation-level pose. Its postcondition requires the target identity and estimated map-frame region to be written into the shared collaborative state.

Close-Range Object Grounding. This skill is deployed on the mobile manipulation robot and is invoked after the robot reaches the coarse target region. Its eye-in-hand RGB-D camera first acquires a close-range observation. Given a textual target description, SAM 3 extracts the corresponding object mask $\mathcal{S}_o$. Depth observations inside the mask are back-projected into the target point cloud

$${}^{c}\mathcal{P}_o = \left\{ d(u,v)K^{-1}\begin{bmatrix} u & v & 1 \end{bmatrix}^{\top} \;\middle|\; (u,v) \in \mathcal{S}_o \right\}. \tag{19}$$

The target pose estimated in the camera frame is transformed into the robot base frame through

$${}^{b}\hat{T}_o = {}^{b}T_e \, {}^{e}T_c \, {}^{c}\hat{T}_o, \tag{20}$$

where ${}^{b}T_e$ denotes the current end-effector pose, ${}^{e}T_c$ is obtained from eye-in-hand calibration, and ${}^{c}\hat{T}_o$ is estimated from the segmented RGB-D observation.

For destination grounding, SAM 3 extracts the target surface, container, or designated placement region according to its textual description. The corresponding point cloud is obtained through the same back-projection process, from which the desired placement pose ${}^{b}\hat{T}_d$ is estimated. The postcondition requires a valid target or destination pose to be available for the subsequent manipulation skill.

D.3 MANIPULATION SKILLS

Manipulation skills change object states through grasping, placement, handover, or table-level processing. They are instantiated on the mobile manipulation robot and the fixed dual-arm robot.

Object Collection. This skill is deployed on the mobile manipulation robot. It receives the target pose ${}^{b}\hat{T}_o$ produced by close-range object grounding and computes a feasible joint configuration through inverse kinematics,

$$q^* = \mathrm{IK}\left({}^{b}\hat{T}_o; q_{\mathrm{ref}}\right), \tag{21}$$

subject to the robot joint limits, workspace constraints, and collision constraints. Here q_{ref} denotes a predefined reference configuration used to select a stable inverse-kinematics solution.

The manipulator approaches the target, closes the gripper, and returns to the carrying posture. The expected postcondition requires the gripper to hold the assigned object and the object identity to remain associated with the mobile manipulation robot.

Object Placement. This skill receives the desired destination pose ${}^{b}\hat{T}_d$ produced by close-range grounding. The corresponding joint configuration is calculated by

$$q_d^* = \mathrm{IK}\left({}^{b}\hat{T}_d; q_{\mathrm{ref}}\right). \tag{22}$$

The mobile manipulator approaches the destination region, releases the object, and returns to a safe posture. The expected postcondition requires the object to lie within the designated region and the gripper to return to an unloaded state.

VLA-Based Dual-Arm Manipulation. This skill is deployed on the fixed dual-arm robot for table-level object processing. It is exposed as a callable manipulation skill in $\mathcal{K}_r^{\mathrm{man}}$ and can be selected directly by the Planner Agent. GR00T N1.5 serves as the underlying vision-language-action policy that grounds the structured skill arguments into coordinated dual-arm actions.

Given the current visual observation o_t, dual-arm state s_t^m, manipulation instruction I_t^m, and structured arguments α_t^m, the skill predicts an action segment

$$a_{t:t+H} = \pi_{\mathrm{VLA}}\left(o_t, s_t^m, I_t^m, \alpha_t^m\right), \tag{23}$$

where $a_{t:t+H}$ contains the coordinated actions of both arms over horizon H. The generated segment is executed before the next visual observation and robot state are acquired.

The skill supports designated tabletop placement and category-conditioned sorting. For designated placement, the structured arguments specify the target object and final tabletop region. For category-conditioned sorting, the Planner Agent provides the object category c_o and its associated destination region $\mathcal{R}(c_o)$. The manipulation instruction can be written as

$$I_{\mathrm{sort}} = \langle o, c_o, \mathcal{R}(c_o) \rangle. \tag{24}$$

The fixed dual-arm robot then grasps, transfers, and releases the object into $\mathcal{R}(c_o)$.

The classification tasks in the benchmark therefore refer to category-conditioned physical sorting rather than an independent visual-category recognition module. The expected postcondition requires the processed object to lie within the designated tabletop region while maintaining collision-free dual-arm execution.

D.4 CONTROLLER-LEVEL AUXILIARY ROUTINES

Controller-level auxiliary routines support the three task-oriented skill categories. They are invoked internally by the Execution Agent or the associated robot controller when required by a selected task-oriented skill. They are not independent Planner-callable skills. These routines establish valid controller states, safe robot configurations, and handover conditions.

Navigation Robot Initialization. This routine activates the LiDAR, RGB-D cameras, coordinate transformations, localization modules, and the `move_base` navigation stack. Its postcondition requires valid sensor observations, available coordinate transformations, and an operational navigation interface before navigation or perception skills are invoked.

Mobile Manipulation Robot Initialization. This routine activates the mobile base, LiDAR, RGB-D cameras, AMCL, `move_base`, robotic arm, gripper, and the required coordinate transformations. The manipulator is simultaneously moved to a predefined safe configuration before navigation begins. Its postcondition requires both the mobile base and manipulator interfaces to be available.

Preset Posture Control. The mobile manipulation robot uses the posture library $\mathcal{Q} = \{q_{\text{init}}, q_{\text{obs}}, q_{\text{pre-grasp}}, q_{\text{carry}}, q_{\text{pre-place}}, q_{\text{recover}}\}$ for initialization, close-range observation, grasp preparation, transportation, placement, and recovery. Given a posture identifier j, this routine executes

$$q \leftarrow q_j. \tag{25}$$

These predefined configurations provide repeatable intermediate states and reduce unnecessary arm motion during transitions between navigation and manipulation.

Fixed Dual-Arm Robot Initialization. This routine activates both robotic arms, grippers, RGB cameras, the VLA inference module, and the required coordinate transformations. The two arms are then moved to collision-free initial configurations q_L^0 and q_R^0. Its postcondition requires both arms and the VLA inference interface to be available.

Dual-Arm Receiving Pose. This routine moves the fixed dual-arm robot to the predefined receiving configuration (q_L^r, q_R^r) before object delivery. The receiving pose preserves the handover workspace and prepares the dual-arm robot for the subsequent VLA-based manipulation skill.

Dual-Arm Reset. After table-level manipulation, this routine returns both arms to their predefined initial configurations,

$$(q_L, q_R) \leftarrow (q_L^0, q_R^0). \tag{26}$$

Its postcondition requires the operating workspace to be released and the fixed dual-arm robot to be ready for the next object.

D.5 Examples of Composed Execution Flows

The following examples illustrate how SynerMind combines multiple task-oriented skills with controller-level routines during long-horizon collaboration. These examples are workflow-level execution sequences rather than independent skills. They are not included as additional entries in any robot-specific skill set. In the sequences below, initialization, posture, receiving-pose, and reset symbols denote internal controller routines nested within execution rather than independent Planner choices.

Exploration-Guided Collection. A typical exploration-guided collection process first initializes the navigation robot, constructs the occupancy grid, and estimates the coarse target region. The mobile manipulation robot is then initialized, navigates to the shared target region, performs close-range grounding, and collects the object. The resulting execution sequence can be expressed as

$$\begin{aligned}\tau_{\text{collect}} = k_{\text{init}}^{\text{nav}} &\rightarrow k_{\text{explore}} \rightarrow k_{\text{localize}} \\ &\rightarrow k_{\text{init}}^{\text{mobile}} \rightarrow k_{\text{navigate}} \rightarrow k_{\text{ground}} \rightarrow k_{\text{collect}}.\end{aligned} \tag{27}$$

The final workflow state requires the target to be localized and held by the mobile manipulation robot.

Transportation and Delivery. After object collection, the mobile manipulation robot switches to the carrying posture, navigates toward the assigned destination, grounds the final placement region, and releases the object. A representative sequence is

$$\begin{aligned}\tau_{\text{delivery}} = k_{\text{posture}}(q_{\text{carry}}) &\rightarrow k_{\text{navigate}} \\ &\rightarrow k_{\text{ground}} \rightarrow k_{\text{place}}.\end{aligned} \tag{28}$$

For waypoint-constrained tasks, the specified map-frame waypoint is inserted as an intermediate navigation postcondition. For obstacle-constrained tasks, the same sequence is executed under the corresponding route constraint.

Cross-Robot Handover and Tabletop Manipulation. For tasks involving all three embodiments, the fixed dual-arm robot first enters the receiving pose. The mobile manipulation robot then reaches the operating area and places the target in the handover region. The fixed dual-arm robot invokes the

Hyperparameter	Value	Hyperparameter	Value
Reward balance α	0.5	GRPO clipping ratio	0.2
Feasibility penalty λ_f	1.0	KL coefficient	0.01
History penalty λ_r	0.25	Warmup ratio	0.03
Rollout group size K	8	Learning-rate schedule	cosine
GRPO stability constant ϵ_{std}	10^{-6}	Maximum gradient norm	1.0
CSM coefficient κ	1.0	Prompt batch size	8
CSM mixing ratio ρ	0.5	Optimization epochs per rollout batch	1
State-sampling temperature τ_{temp}	1.0	Rollout temperature / top-p	$1.0/0.95$
Optimizer	AdamW	Maximum completion length	1024 tokens
Learning rate	1×10^{-6}	Numerical precision	bfloat16
AdamW coefficients (β_1, β_2)	$(0.9, 0.95)$	Random seed	42
Weight decay	0.1		

Table 6: Hyperparameters used for CSM-GRPO training.

Component	Base-Flow	w/o Reflection	Full Flow
Stepwise Observation Update	✓	✓	✓
Runtime Status Feedback	✓	✓	✓
Structured Workflow	×	✓	✓
Execution-Agent Grounding	×	✓	✓
Postcondition Verification	×	×	✓
Failure Attribution	×	×	✓
Targeted Correction	×	×	✓

Table 7: Component comparison among the three Flow variants. All variants use the same shared brain, robot skills, and execution settings.

VLA-based skill for designated placement or category-conditioned sorting, followed by dual-arm reset:

$$\begin{aligned}\tau_{\text{handover}} = k_{\text{receive}} \rightarrow k_{\text{navigate}} \rightarrow k_{\text{place}} \\ \rightarrow k_{\text{VLA}} \rightarrow k_{\text{reset}}.\end{aligned} \tag{29}$$

This sequence illustrates a coordinated workflow across different robots rather than a single compound skill.

Sequential Multi-Target Processing. For multi-target tasks, SynerMind repeatedly instantiates the collection, delivery, and tabletop-manipulation sequences for each target. After processing object o_i, the Reflection Agent verifies the resulting collaborative state before the workflow advances to o_{i+1}:

$$\mathcal{W}_i : \quad \tau_{\text{collect}}(o_i) \rightarrow \tau_{\text{delivery}}(o_i) \rightarrow \tau_{\text{handover}}(o_i) \rightarrow \text{Verify}(o_i). \tag{30}$$

Here, $\text{Verify}(\cdot)$ denotes a Flow-level state-verification stage rather than a robot skill. The next target is activated only after the current object state and required processing order have been verified.

E PARAMETER

Table 6 reports the CSM-GRPO-stage hyperparameters used by all applicable SynerMind variants. We set $\alpha = 0.5$ to weight reasoning consistency and task-frontier contribution equally, and use $\lambda_f = 1.0$ and $\lambda_r = 0.25$. All applicable checks within P_i, U_i, and C_i are uniformly weighted, while F_i remains a binary feasibility gate. For GRPO, raw rewards are standardized within each rollout group, with $\epsilon_{\text{std}} = 10^{-6}$ added to the standard-deviation denominator.

F Details of Compared Methods

We compare SynerMind with representative methods through communication, task planning, and role assignment. PEFA and CMRS represent centralized task planning and robot assignment, while DMRS-1D and DMRS-2D represent communication-based decentralized collaboration. Primitive MCTS and LLM-MCTS are included as search-based planning baselines. All methods use the same simulation tasks, robot embodiments, and admissible skill library. For a fair comparison, all LLM-based methods are instantiated with the same Qwen backbone. GT is included only as an expert reference.

PEFA. We implement PEFA as a feedback-guided task-assignment baseline. A centralized task assigner receives the global instruction, robot capabilities, current observations, and execution history. It decomposes the task into subtasks and assigns the current subtask to a suitable robot. The corresponding robot executor selects an admissible skill and returns execution feedback. The task assigner then updates the remaining task proposal. This proposal–execution–feedback–adjustment cycle continues until task completion or the maximum step budget is reached.

Primitive MCTS. Primitive MCTS is implemented as a search-based planner without language-model guidance. Each search node represents the current collaborative state, while each branch corresponds to an admissible robot-skill action. The method performs selection, expansion, rollout, and backpropagation directly over the shared skill space. The action with the highest estimated value at the root node is executed before replanning from the updated state.

LLM-MCTS. LLM-MCTS extends Primitive MCTS with an LLM-derived policy prior. Given the task instruction, current observations, and execution history, the LLM estimates the preferences of admissible robot-skill actions. These preferences guide action selection in PUCT, concentrating the search on more task-relevant branches. The remaining tree expansion, rollout, and backpropagation procedures follow Primitive MCTS.

CMRS. CMRS is implemented as a centralized task-planning and robot-assignment baseline. A single LLM receives the global instruction, shared robot states, capability descriptions, execution history, and admissible skill lists. It directly selects the next robot together with the corresponding skill and arguments. After execution, the updated global state is added to the context for the next planning step. CMRS performs centralized allocation without inter-robot dialogue.

DMRS-1D. DMRS-1D represents communication-based decentralized collaboration. Each robot is associated with an independent language agent that receives its local observation, embodiment capabilities, task progress, and admissible skills. The agents exchange one round of messages describing their observations, capabilities, and intended actions. The next robot-skill decision is generated from the aggregated dialogue before execution.

DMRS-2D. DMRS-2D extends DMRS-1D with two rounds of inter-agent communication. During the first round, each agent shares its local task state and preliminary decision. During the second round, the agents revise their proposals according to the messages received from other robots. The final robot-skill decision is then selected from the updated dialogue context. All other settings remain identical to DMRS-1D.

G Task Definition

To comprehensively evaluate SynerMind in long-horizon heterogeneous robot collaboration, we construct a real-world benchmark containing 24 tasks across four categories, as summarized in Table 8. The four categories are Coarse-Grained Tasks, Instruction-Constrained Tasks, Pipelined Multi-Target Tasks, and Novel Scene-Target Tasks. Each task requires the collaboration of at least two heterogeneous robots and consists of multiple dependent execution stages, including exploration, target localization, navigation, collection, transportation, placement, or tabletop operation.

Tasks within the same group follow a shared task template while varying target objects, intermediate waypoints, obstacle layouts, processing orders, or environmental configurations. For Tasks 7–9, each intermediate waypoint is represented by a predefined map-frame pose $w_i = (x_i, y_i, \theta_i)$. The waypoint coordinates remain fixed across all compared methods. A task is considered successful only

Task ID	Description
CLS. I: Coarse-Grained Tasks	
Task 1–3	Exploration-guided collection of a bottle, cube, or rock and placement on a cylindrical platform
Task 4–6	Exploration-guided transportation of a bottle, cube, or rock followed by designated tabletop placement
CLS. II: Instruction-Constrained Tasks	
Task 7–9	Object collection and cylindrical-platform placement through specified map-frame waypoints
Task 10–12	Object collection and cylindrical-platform placement while avoiding a designated obstacle
CLS. III: Pipelined Multi-Target Tasks	
Task 13–15	Ordered collection, sequential delivery, and tabletop classification of two target objects
Task 16–18	Sequential removal of two obstacle screens from different workspace regions
CLS. IV: Novel Scene-Target Tasks	
Task 19–21	Collection of novel target objects and placement on a cylindrical platform
Task 22–24	Full-chain transportation and tabletop placement under changed terrain configurations

Table 8: Summary of the 24 real-world heterogeneous robot collaboration tasks. Tasks within each range share the same execution template while varying target objects, spatial constraints, processing orders, or environmental configurations.

when all required intermediate transitions and final task conditions are completed in the specified order.

G.1 Coarse-Grained Tasks

This category evaluates whether SynerMind can expand a high-level instruction into a complete heterogeneous collaboration workflow. The instructions specify only the target object and final destination without explicitly assigning robot roles or intermediate skills. The system must determine the required exploration, target localization, collection, transportation, and placement stages according to the current collaborative state.

Tasks 1–3 require the navigation robot to explore the workspace and estimate the coarse position of the target object. The mobile manipulation robot then receives the shared occupancy grid and target location, approaches the object, performs close-range collection, and places it on the cylindrical platform.

Tasks 4–6 extend this workflow to three-robot collaboration. After the target object is located and transported to the operating area, the fixed dual-arm robot receives the object and places it at the tabletop position specified by the upper-level instruction.

G.2 Instruction-Constrained Tasks

This category evaluates long-horizon collaboration under explicit spatial and route constraints. In addition to completing the final task goal, the robot team must satisfy specified intermediate conditions during execution.

Tasks 7–9 require the mobile manipulation robot to pass through a predefined map-frame waypoint before reaching the cylindrical platform. The corresponding waypoint $w_i = (x_i, y_i, \theta_i)$ is treated as an intermediate postcondition, and subsequent transportation is activated only after the waypoint has been reached.

Tasks 10–12 introduce obstacle-specific navigation constraints. The navigation robot identifies the target obstacle and estimates a feasible route, while the mobile manipulation robot transports the object without colliding with the specified obstacle and places it on the cylindrical platform.

G.3 Pipelined Multi-Target Tasks

This category evaluates whether SynerMind can preserve object identity, processing order, execution progress, and robot handover states across repeated collaboration cycles.

Tasks 13–15 require two objects to be collected and delivered in a specified order. While the mobile manipulation robot retrieves the next object, the fixed dual-arm robot processes the previously delivered object, forming a continuous transportation-and-manipulation pipeline. Successful execution requires both objects to be delivered and classified into their corresponding tabletop regions without reversing their required order.

Tasks 16–18 require the navigation robot to locate two obstacle screens in a specified workspace region. The mobile manipulation robot then removes the two screens sequentially. The shared collaborative state is updated after each removal, and the task is completed only when both screens have been cleared from the target region.

G.4 Novel Scene-Target Tasks

This category evaluates collaboration under unseen target appearances and changed environmental configurations.

Tasks 19–21 preserve the exploration-and-collection workflow of Tasks 1–3 while replacing the original objects with novel targets. The destination remains the cylindrical platform. These tasks evaluate open-vocabulary target localization and manipulation under target appearance variations.

Tasks 22–24 preserve the three-robot workflow of Tasks 4–6 while changing the terrain and obstacle configuration. The navigation robot must reconstruct the occupancy grid, after which the mobile manipulation robot collects and transports the target object under the updated spatial layout. The fixed dual-arm robot subsequently completes the designated tabletop placement.

H Flow Details

Common Settings. All Flow variants use the same SynerMind-Brain checkpoint, multimodal observations, robot platforms, low-level skill library, and robot controllers. They also share the same decoding settings, maximum execution budget, task-success criteria, and termination conditions. The variants differ only in how high-level decisions are organized, grounded, and updated during task execution.

Base-Flow. It follows a conventional brain-cerebellum hierarchy. At the beginning of each task, the shared brain decomposes the global instruction into a coarse sequence of task-level steps. The corresponding robot controllers execute the low-level skills required by the current step. After each execution, the latest observations and runtime status are returned to the shared brain. The next task-level decision is generated from the original instruction, the remaining task steps, and the updated observations.

Base-Flow therefore supports stepwise replanning rather than open-loop execution. However, it does not instantiate role-conditioned Planner, Execution, or Reflection Agents. It also does not construct a structured robot-skill-level workflow or verify execution against explicit postconditions. Runtime information is used only as context for the next planning step. Independent state verification, failure attribution, and targeted corrective feedback are excluded.

SynerMind-Flow without Reflection. The w/o Reflection variant retains the Planner Agent and Execution Agent without modification. The Planner Agent constructs the structured workflow, selects the robot-skill assignment, and predicts the expected postcondition. The Execution Agent grounds the assignment into an executable skill call under the same robot capabilities and skill constraints used by the complete SynerMind-Flow.

After skill execution, the latest observations and runtime status are directly appended to the collaborative context for the next planning round. The expected postcondition remains part of the Planner output, but no independent agent compares it with the observed state transition. This variant therefore removes postcondition verification, failure attribution, and targeted corrective feedback. All remaining components and evaluation settings are identical to the complete SynerMind-Flow.

I Cross-Method Simulation Execution Logs

To complement the aggregate simulation results, we report normalized execution traces for all evaluated methods on the same representative task. The selected case requires a robot arm to place a plate into the basket carried by a quadrotor. The quadrotor must subsequently transport the plate from the living room to the kitchen, approach the designated high table, and complete the final landing. Successful execution therefore depends on ordered manipulation, robot-to-robot handover, inter-room navigation, and landing under explicit preconditions.

All methods are evaluated on the same task instance, initial scene state, robot embodiments, admissible action space, and termination budget. We report step-level or phase-level normalized traces instead of verbatim terminal logs. Each record retains the executed robot-skill action, the resulting state update, and the feedback that changes the subsequent decision. Repeated scene descriptions, complete candidate-action lists, accumulated dialogue histories, local file paths, launch commands, and duplicated debugging messages are omitted.

I.1 SIMULATION ACTION-SELECTION PROMPT

All LLM-based methods use the same base prompt, with modifications limited to method-specific requirements. The following is a condensed version of this base prompt, retaining robot capabilities, task inputs, and the output format.

```
You are a housework helper coordinating three types of robots in an
    apartment, each with different capabilities and action spaces. When
    one robot cannot complete a task alone, other robots can assist to
    accomplish long-horizon tasks. Given the robots' capabilities,
    current observations, action history, available actions, and task
    goal, decide which robot should execute what action next.

Robot capabilities:

Quadrotor:
The quadrotor can take off, fly, and land. It transports objects in a
    basket but cannot pick up or place objects itself. It can travel
    between rooms only through open doors. Other robots can access its
    basket only when it has landed within their reach. To reach another
    landing surface, it must take off, move toward the destination, and
    land on a surface marked LANDABLE.

Robot dog:
The robot dog has a robotic arm and can manipulate objects on the floor
    or low surfaces, but cannot reach high surfaces. It can move between
    rooms through open doors and open or close accessible doors and
    containers when its hand is empty. Before manipulation, it must
    approach the relevant object, container, or surface. To access the
    quadrotor's basket, it must approach the basket after the quadrotor
    has landed within reach.

Robot arm:
The robot arm is fixed on a table or platform. It can manipulate objects
    and containers within reach, including objects in the basket of a
    quadrotor that has landed on the same surface. It cannot reach
    objects on other tables and must have an empty hand to open or close
    containers.

Task goal:
#TASK_GOAL#

Number of robots:
#NUMBER_ROBOTS#

Current observations:
#ROBOT_OBSERVATIONS#

Available actions:
#ACTION_LIST#

Action history:
#ACTION_HISTORY#

Select exactly one action from ACTION_LIST that advances the task goal.
    Consider all robots' capabilities and observations, satisfy the
    action's prerequisites, and use the action history to avoid
    repeating completed actions. Coordinate multiple robots when needed.

Output only the selected action in the following format, preserving the
    lowercase names and identifiers in ACTION_LIST:
<robot>(id): [action] <class name>(id)
```

I.2 Task Setting and Overall Outcomes

To provide a controlled comparison of cross-method execution behavior, we select a representative long-horizon collaboration task that requires coordinated object transfer, cross-room transport, and target landing. As summarized in Table 9, successful completion requires both the plate to be placed inside the basket and the quadrotor to land on the high kitchen table. Table 10 reports the corresponding execution outcomes. Among the compared methods, only DMRS-2D and SynerMind satisfy both goal predicates, while SynerMind completes the task in fewer steps through reflection-guided workflow corrections.

Item	Description
Task instance	Environment 0, Task 9.
Global goal	The quadrotor lands on the high kitchen table with the plate inside its basket.
Required collaboration	The robot arm performs object transfer, while the quadrotor performs cross-room transport and target landing.
Goal predicates	The plate is inside the basket; the quadrotor is on the high kitchen table.
Expert reference	The manually designed reference trajectory contains nine environment steps.

Table 9: Configuration of the representative simulation task.

Method	Result	Steps/Budget	Termination
PEFA	**Failure**	19	The step budget is reached before both goal predicates are satisfied.
Primitive MCTS	**Failure**	19	The search executes unrelated or weakly valued actions and fails to complete the handover.
LLM-MCTS	**Timeout**	900 s	The runtime budget is exhausted before a complete task trajectory is produced.
CMRS	**Failure**	19	The plate remains outside the basket when the step budget is reached.
DMRS-1D	**Failure**	19	One-round dialogue does not stabilize an executable handover sequence.
DMRS-2D	**Success**	16	Both goal predicates are satisfied.
SynerMind	**Success**	15	Both goal predicates are satisfied after reflection-guided corrections.

Table 10: Execution outcomes on the selected simulation task.

I.3 SynerMind Execution Trace

The SynerMind trace retains corrective events because they alter the executed workflow. A suffix "R" denotes the corrected retry generated within the same planning round.

Table 11: Normalized execution trace of SynerMind.

Step	Selected Robot-Skill Action	Observed State Update	Reflection / Workflow Update
0	`<robot arm>(24): [grab] <plate>(51)`	The robot arm holds the plate.	The object-transfer stage begins.
1	`<robot arm>(24): [puton] <plate>(51) on <dining table>(13)`	The plate is placed on the dining table.	The quadrotor must approach the handover surface.
2	`<quadrotor>(22): [takeoff_from] <livingroom floor>(5)`	The quadrotor becomes airborne.	The dining table is selected as the intermediate target.
3	`<quadrotor>(22): [movetowards] <dining table>(13)`	The quadrotor reaches the dining-table region.	Normal workflow progress.
4	`<quadrotor>(22): [land_on] <dining table>(13)`	The quadrotor lands beside the robot arm.	The basket becomes accessible for handover.

Continued on the next page

Table 11 continued from the previous page

Step	Selected Robot-Skill Action	Observed State Update	Reflection / Workflow Update
5	Proposed <robot arm>(24): [putinto] <plate>(51) into <basket>(25)	No executable state transition is produced.	Reflection identifies the missing grasp precondition and requests a grasp action first.
6-R	<robot arm>(24): [grab] <plate>(51)	The robot arm holds the plate.	The corrected action is accepted.
7	<robot arm>(24): [putinto] <plate>(51) into <basket>(25)	The plate enters the basket.	The first goal predicate is satisfied.
8	<quadrotor>(22): [takeoff_from] <dining table>(13)	The quadrotor becomes airborne with the plate.	The transport stage begins.
9	Proposed direct movement toward the final table	The target action is not executable from the current local state.	Reflection replaces the direct transition with accessible intermediate navigation actions.
10	<quadrotor>(22): [movetowards] <livingroom floor>(5)	The quadrotor reaches an accessible intermediate region.	The kitchen becomes the next navigation target.
11	<quadrotor>(22): [movetowards] <kitchen>(6)	The quadrotor enters the kitchen.	The destination table becomes locally reachable.
12	Proposed <quadrotor>(22): [land_on] <high kitchen table>(35)	The quadrotor is not yet positioned above the landing surface.	Reflection requests target approach before landing.
13-R	<quadrotor>(22): [movetowards] <high kitchen table>(35)	The quadrotor reaches the target-table region.	The landing precondition is satisfied.
14	<quadrotor>(22): [land_on] <high kitchen table>(35)	The quadrotor lands with the plate inside its basket.	All goal predicates are satisfied.

I.4 PEFA Execution Trace

PEFA repeatedly assigns the plate-transfer subtask to the robot arm. The normalized trace groups consecutive repetitions while preserving the observed action pattern and the final termination state.

Table 12: Normalized execution trace of PEFA.

Step(s)	Selected Action Pattern	Execution Result / State Update
0	<robot dog>(23): [movetowards] <fridge>(14)	The action does not advance either goal predicate.
1	<robot arm>(24): [grab] <plate>(51)	The robot arm holds the plate.
2	<robot arm>(24): [puton] <plate>(51) on <dining table>(13)	The plate is returned to the dining table rather than placed in the basket.
3–15	Alternating [grab] <plate>(51) and [puton] <plate>(51) on <dining table>(13)	The method repeatedly changes the local plate state without completing the handover.
16	No executable action is parsed.	No visible state change is produced.
17–18	The robot arm again alternates placement and grasp actions.	The plate remains outside the basket when the budget is exhausted.
Final	Termination at 19 steps.	Both goal predicates are not jointly satisfied; the episode is counted as failure.

I.5 Primitive MCTS Execution Trace

Primitive MCTS performs planning directly over the admissible robot-skill space without language-model guidance. The trace below reports only the root actions selected by the search and executed in the simulator. Internal selection, expansion, rollout, and backpropagation records are omitted because they do not correspond to environment actions.

Table 13: Normalized execution trace of Primitive MCTS.

Step(s)	Executed Root Action(s)	Execution Result / State Update
1–5	<robot dog>(23): [movetowards] <fridge>(14); <quadrotor>(22): [takeoff_from] <livingroom floor>(5);<robot dog>(23): [movetowards] <door>(10);<quadrotor>(22): [movetowards] <dining table>(13);<robot dog>(23): [movetowards] <fridge>(14)	Navigation progresses without establishing the required plate handover.

Continued on the next page

Table 13 continued from the previous page

Step(s)	Executed Root Action(s)	Execution Result / State Update
6–10	<robot arm>(50): [grab] <battery>(39);<robot arm>(24): [grab] <plate>(51);<robot dog>(23): [movetowards] <kitchen table>(32); <quadrotor>(22): [movetowards] <kitchen>(6); <robot dog>(23): [movetowards] <livingroom>(4)	Several selected actions concern robots or objects unrelated to the active handover dependency.
11–15	<robot dog>(23): [movetowards] <knife>(54);<robot dog>(23): [movetowards] <sofa>(11);<robot arm>(50): [puton] <battery>(39) on <high table>(33);<robot dog>(23): [movetowards] <coffee table>(12);<robot dog>(23): [movetowards] <door>(8)	The search continues to explore task-irrelevant branches.
16–19	<robot dog>(23): [movetowards] <door>(10);<robot arm>(50): [grab] <battery>(39);<robot arm>(50): [puton] <battery>(39) on <high table>(33);<robot arm>(24): [puton] <plate>(51) on <dining table>(13)	The plate never enters the basket before the step budget is exhausted.
Final	Termination at 19 steps.	Both goal predicates are not jointly satisfied; the episode is counted as failure.

I.6 LLM-MCTS Execution Trace

LLM-MCTS augments tree search with an LLM-derived policy prior. As with Primitive MCTS, only root actions that are selected and executed in the simulator are retained. Internal tree-search trajectories are omitted. The runtime terminates when the 900-second search budget is reached.

Table 14: Normalized execution trace of LLM-MCTS.

Step	Executed Root Action	Execution Result / State Update
1	<quadrotor>(22): [takeoff_from] <livingroom floor>(5)	The quadrotor becomes airborne.
2	<robot arm>(24): [grab] <plate>(51)	The robot arm holds the plate.
3	<quadrotor>(22): [movetowards] <kitchen>(6)	The quadrotor leaves the handover region before the plate enters the basket.
4	<quadrotor>(22): [movetowards] <high kitchen table>(35)	The destination is approached while the plate-handover predicate remains unsatisfied.
Final	No complete executable continuation is returned within the search budget.	The 900-second runtime budget is exhausted; the episode is counted as timeout and failure.

I.7 CMRS Execution Trace

CMRS receives the shared state through a centralized prompt and replans after each environment transition. The raw run does not expose a separate verification record, so repeated or unparsable decisions are represented as normalized phase-level entries.

Table 15: Normalized execution trace of CMRS.

Phase	Observed Decision Pattern	Execution Result / State Update
1	The robot arm grasps the plate, followed by repeated placement on the dining table.	The plate is manipulated locally, but it is not transferred into the basket.
2	The quadrotor takes off and receives movement or landing actions toward the dining-table and kitchen regions.	Navigation progresses independently of the unresolved plate-handover predicate.
3	Several model outputs cannot be mapped to an available action.	These rounds produce no executable state transition.
4	The quadrotor reaches or leaves the high kitchen table while the robot arm continues to place the plate on the dining table.	The landing predicate and basket predicate are not satisfied simultaneously.
5	The same local manipulation and navigation decisions recur until the maximum step count.	No independent mechanism redirects the method to the missing handover action.
Final	Termination at 19 steps.	The plate remains outside the basket; the episode is counted as failure.

I.8 DMRS-1D Execution Trace

DMRS-1D performs one communication round before selecting the next action. The trace below reports the final parsed action of each environment step.

Table 16: Normalized execution trace of DMRS-1D.

Step(s)	Final Parsed Action	Execution Result / State Update
0–1	Repeated <robot arm>(24): [grab] <plate>(51)	The plate is grasped, but the second decision does not advance the handover.
2–3	Repeated <quadrotor>(22): [takeoff_from] <livingroom floor>(5)	The quadrotor is already airborne after the first valid takeoff.
4–9	No executable action is parsed.	Six consecutive rounds produce no task progress.
10	<robot dog>(23): [movetowards] <livingroom>(4)	The robot dog moves, but neither target predicate is completed.
11–15	No executable action is parsed.	The collaborative state remains unresolved.
16	<robot arm>(24): [puton] <plate>(51) on <dining table>(13)	The plate is returned to the table rather than placed in the basket.
17–18	Repeated <quadrotor>(22): [takeoff_from] <livingroom floor>(5)	The repeated takeoff command does not complete the missing handover.
Final	Termination at 19 steps.	Both goal predicates are not jointly satisfied; the episode fails.

I.9 DMRS-2D Execution Trace

DMRS-2D introduces a second communication round. This additional exchange eventually produces a valid handover and landing sequence, although several repeated or unrelated actions remain in the trajectory.

Table 17: Normalized execution trace of DMRS-2D.

Step	Final Parsed Action	Execution Result / State Update
0	<robot arm>(24): [grab] <plate>(51)	The robot arm holds the plate.
1	<quadrotor>(22): [takeoff_from] <livingroom floor>(5)	The quadrotor becomes airborne.
2	No executable action is parsed.	No visible state change.
3	<robot dog>(23): [movetowards] <livingroom>(4)	The robot dog moves toward the collaboration region.
4	<quadrotor>(22): [takeoff_from] <livingroom floor>(5)	The takeoff action is repeated.
5	<robot dog>(23): [movetowards] <door>(10)	The robot dog changes location without completing a target predicate.
6	<quadrotor>(22): [takeoff_from] <livingroom floor>(5)	Another repeated takeoff decision is produced.
7	<quadrotor>(22): [movetowards] <dining table>(13)	The quadrotor reaches the handover region.
8	<quadrotor>(22): [land_on] <dining table>(13)	The basket becomes accessible to the robot arm.
9	<robot arm>(24): [putinto] <plate>(51) into <basket>(25)	The first goal predicate is satisfied.
10–11	Repeated <quadrotor>(22): [takeoff_from] <dining table>(13)	The quadrotor starts the transport stage after one redundant decision.
12–13	Repeated <quadrotor>(22): [movetowards] <kitchen>(6)	The quadrotor reaches the destination room after one repeated move.
14	<quadrotor>(22): [movetowards] <high kitchen table>(35)	The quadrotor reaches the target-table region.
15	<quadrotor>(22): [land_on] <high kitchen table>(35)	The second goal predicate is satisfied; the task succeeds.

I.10 Case Analysis

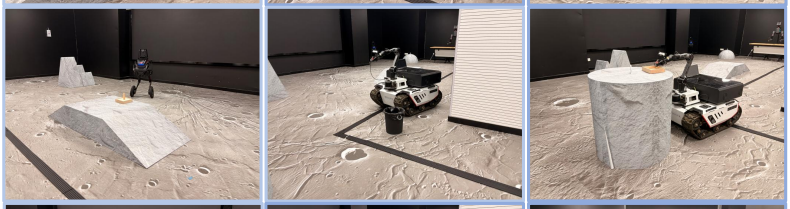
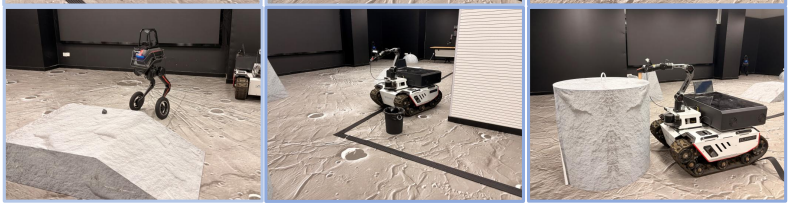
The traces expose distinct failure modes under the same collaborative dependencies. Primitive MCTS spends most of its budget on weakly valued or task-irrelevant branches, while LLM-MCTS reaches the destination side before resolving the plate handover and subsequently exceeds the runtime budget. PEFA, CMRS, and DMRS-1D repeatedly revisit locally plausible actions without restoring the missing cross-robot dependency. DMRS-2D succeeds after additional dialogue, but its trajectory contains repeated takeoff and navigation decisions. SynerMind completes the task in fewer steps by preserving the verified handover state and issuing targeted corrections when a proposed action violates manipulation, navigation, or landing preconditions. This case illustrates how explicit execution verification limits the propagation of local errors across successive robot stages.



Figure 4: Execution examples for coarse-grained Tasks 1–6 with exploration-guided collection and placement.

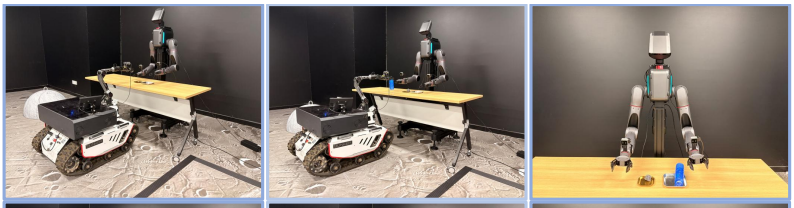
J Execution Demonstration

We provide execution demonstrations for all 24 real-world tasks. Each demonstration is paired with its corresponding instruction and representative observations from the heterogeneous collaboration process. The demonstrations cover exploration and target localization, mobile collection and transportation, object handover, tabletop manipulation, multi-target processing, and collaborative execution under changed environments.

Task 7: Collect the bottle, pass through the specified waypoint, and place it on the cylindrical platform.	
Task 8: Collect the cube, pass through the specified waypoint, and place it on the cylindrical platform.	
Task 9: Collect the rock, pass through the specified waypoint, and place it on the cylindrical platform.	
Task 10: Collect the bottle, avoid the black obstacle, and place it at the designated cylindrical platform.	
Task 11: Collect the cube, avoid the black obstacle, and place it at the designated cylindrical platform.	
Task 12: Collect the rock, avoid the black obstacle, and place it at the designated cylindrical platform.	

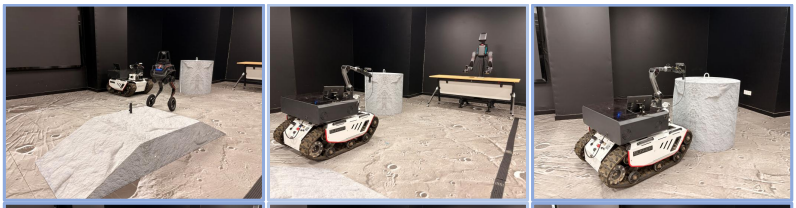
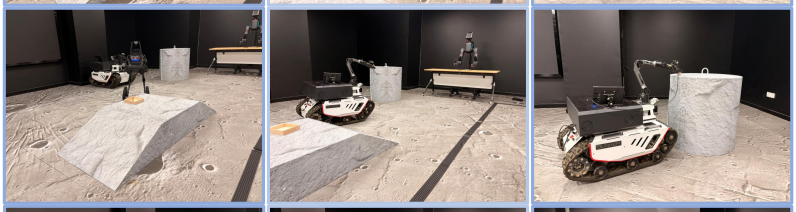
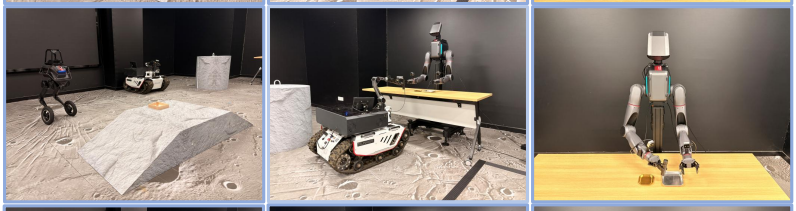
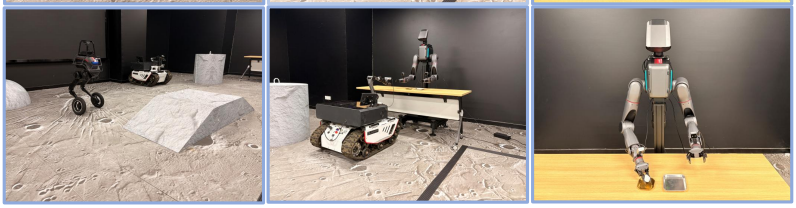
Instruction-Constrained Tasks (6-12)

Figure 5: Execution examples for instruction-constrained Tasks 7–12 with waypoint and obstacle constraints.

Task 13: Collect the bottle and the rock, deliver them sequentially to the operating table, and classify them into their designated regions.	
Task 14: Collect the bottle and the cube, deliver them sequentially to the operating table, and classify them into their designated regions.	
Task 15: Collect the rock and the cube, deliver them sequentially to the operating table, and classify them into their designated regions.	
Task 16: Locate and remove the two obstacle screens on the right side of the slope, then verify that the area has been cleared.	
Task 17: Locate and remove the two obstacle screens on the left side of the slope, then verify that the area has been cleared.	
Task 18: Locate and remove the two obstacle screens in front of the operating table, then verify that the operating area has been cleared.	

Pipelined Multi-Target Tasks (13-18)

Figure 6: Execution examples for pipelined multi-target Tasks 13–18 with ordered delivery, sorting, and obstacle removal.

Task 19: Search the workspace for the beacon, collect it, and place it on the cylindrical platform.	
Task 20: Search the workspace for the cylindrical object, collect it, and place it on the cylindrical platform.	
Task 21: Search the workspace for the white rock, collect it, and place it on the cylindrical platform.	
Task 22: Under the changed terrain configuration, locate the cylindrical object, transfer it to the operating table, and place it at the designated tabletop position.	
Task 23: Under the changed terrain configuration, locate the beacon, transfer it to the operating table, and place it at the designated tabletop position.	
Task 24: Under the changed terrain configuration, locate the white rock, transfer it to the operating table, and place it at the designated tabletop position.	

Novel Scene-Target Tasks(19-24)

Figure 7: Execution examples for Novel Scene-Target Tasks 19–24 with unseen objects and changed terrain.