
CLiMRS: COOPERATIVE LARGE-LANGUAGE-MODEL-DRIVEN HETEROGENEOUS MULTI-ROBOT SYSTEM

Anonymous authors

Paper under double-blind review

ABSTRACT

Cooperative multi-robot tasks often require heterogeneous agents to collaborate over long horizons while managing spatial constraints and execution uncertainties. Although large language models (LLMs) excel at reasoning and planning, their potential for coordinated control in heterogeneous multi-robot teams has not been fully explored. We present **CLiMRS**, an adaptive negotiation framework inspired by human teamwork. The framework pairs each robot with an independent LLM agent and dynamically forms subgroups to facilitate perception-driven discussions and collaborative planning under long-horizon uncertainty. Within each group, local oracle planners lead parallel discussions to synchronize actions, while agents provide feedback to refine plans. This grouping–planning–feedback–execution loop enables efficient long-horizon planning and robust execution. To evaluate these capabilities, we introduce **CLiMBench**, a heterogeneous multi-robot benchmark of challenging assembly tasks with diverse robot types and skill libraries. Across both **CLiMBench** and a simpler benchmark, **CLiMRS** surpasses the best baseline, boosting success rates and improving efficiency by over 40% on complex tasks while maintaining very high success on simpler tasks. Our results demonstrate that leveraging human-inspired group formation and negotiation principles markedly enhances the efficiency of heterogeneous multi-robot collaboration.

1 INTRODUCTION

Addressing real-world, everyday tasks often requires collaboration to enhance the efficiency of long-horizon, complex planning and perception. Meanwhile, the development of intelligent agents that can assist embodiments in accomplishing such tasks remains an open challenge, particularly regarding how these agents can effectively help humans and other robots execute such intricate operations. Inspired by human teamwork, incorporating principles of human teaming into multi-agent systems, where sub-groups coordinate planning and perception through shared observations and information, offers a promising yet challenging path to improving efficiency and robustness Zhang et al. (2024b).

At the same time, large language models (LLMs) have exhibited outstanding performance across various dimensions, including natural language question answering Rein et al. (2024), code generation Jain et al. (2024), and logical reasoning Plaat et al. (2024). In recent years, numerous studies have integrated LLMs into robotic planning scenarios Song et al. (2023); Zhang et al. (2024a); Mower et al. (2024); Salimpour et al. (2025); Liang et al. (2025), with some extending their application to multi-robot collaborative planning tasks Zhang et al. (2024b); Mandi et al. (2024); Liu et al. (2025).

However, earlier explorations of robot collaboration largely center on homogeneous agents, which restricts the range of capabilities that can be demonstrated Liu et al. (2024a). Furthermore, works on heterogeneous teams typically assume ideal operating conditions Liu et al. (2025), while such assumptions ignore the cumulative errors that escalate over long horizons, driving up communication costs and undermining cooperative efficiency. While these advances show the promise of LLM-driven multi-robot collaboration, important gaps persist when the setting involves heterogeneous agents, long-horizon objectives, and the practical constraints of real-world operation.

To address these limitations, we propose **CLiMRS** (**C**ooperative **L**arge-**L**anguage-**M**odel-**D**riven **H**eterogeneous **M**ulti-**R**obot **S**ystem), a human-team-inspired LLM-driven adaptive-negotiation framework that orchestrates heterogeneous robots through dynamic sub-group formation and cooperative planning, supporting robust long-horizon collaboration in uncertain environments.

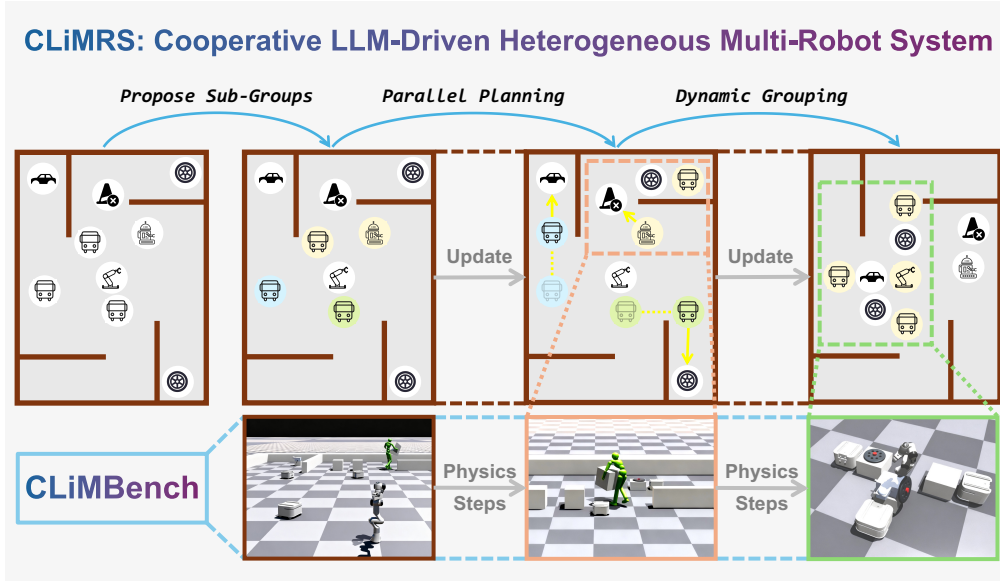


Figure 1: **Overview.** We present **CLiMRS**, a human-team-inspired negotiation paradigm for heterogeneous multi-robot systems that dynamically forms perception-driven discussion sub-groups, and **CLiMBench**, a heterogeneous multi-robot benchmark with challenging assembly tasks.

In this framework, each robot is guided by an independent LLM agent that communicates with peers to accomplish complex, long-horizon tasks. To strengthen collaborative effectiveness, the system leverages the broad world knowledge of LLMs and explicitly models inter-agent dependencies through a carefully designed grouping–planning–feedback–execution loop.

With **CLiMRS**, we further explore its applicability to challenging industrial scenarios, where heterogeneous robots must handle unpredictable execution errors. To evaluate this, we introduce **CLiMBench**, a benchmark for heterogeneous multi-robot collaboration. It features five robotic devices across three types of heterogeneous robots, equipped for transportation, conveyance, and assembly. Tasks of varying difficulty simulate material-handling and assembly processes with diverse skill usage, designed to test the planning and perception capabilities of LLM-based frameworks.

We evaluated our proposed framework in two distinct environments: **CLiMBench** and another heterogeneous robot collaboration benchmark Liu et al. (2025). Our experiments show that **CLiMRS** outperforms the best baseline, increasing success rates and improving efficiency by over 40% on complex tasks while maintaining high success on simpler ones. These results demonstrate that incorporating human-inspired group formation and negotiation principles substantially enhances the efficiency of heterogeneous multi-robot collaboration. To summarize, our main contributions are:

- We present **CLiMRS**, a multi-LLM cooperation framework for heterogeneous multi-robot collaboration which can perform long-horizon planning and efficient perception in complex tasks.
- We propose **CLiMBench**, a benchmark evaluating heterogeneous multi-robot collaboration in industrial assembly scenarios, featuring varied skill sets and a realistic simulation environment.
- We demonstrate through extensive experiments that **CLiMRS** achieves significant efficiency improvements via dynamic group formation and cooperative long-horizon planning.

2 RELATED WORK

2.1 EMBODIED SKILLS TRAINING ACROSS DIVERSE SCENARIOS

Embodied Agent Skill Training. Approaches to train embodied skills for task execution generally follow two primary paradigms: rule-based and learning-driven methods. Traditional embodiment controllers optimize joint movements through the resolution of robotic kinematics, aiming to im-

prove motion robustness and generate smoother, more precise trajectories Kashyap & Parhi (2021); Katayama et al. (2023). In recent years, with the advances of reinforcement learning and imitation learning in robotic motion control, spanning domains such as dexterous manipulation Rajeswaran et al. (2017); Zhu et al. (2019); Chen et al. (2022); Luo et al. (2025), bipedal locomotion Li et al. (2025); Zhang et al. (2024c); Serifi et al. (2024), and quadrupedal navigation Bellegarda et al. (2024); Shi et al. (2024), embodied perception has progressively learned to coordinate actions in a cerebellum-like fashion, enabling increasingly complex tasks in diverse environments. Overall, as tasks and environments grow more complex, embodied intelligence is shifting from traditional low-level planning toward more integrated, end-to-end perception and control.

Multi-agent Skill Training. Originally developed in game AI Kurach et al. (2020); Perolat et al. (2022), multi-agent skill training has since extended to industrial fields such as robotics Wang et al. (2024); Lai et al. (2025) and autonomous driving Li et al. (2022), where many of the coordination and credit-assignment strategies first pioneered in games remain fundamental. Despite these advances, current methodologies for multi-agent embodied tasks remain underdeveloped, particularly in light of the exponential state-space challenges introduced by an increasing number of robotic agents. Although certain researchers have explored mean-field approximations to alleviate these challenges Yang et al. (2018), robust generalization across heterogeneous robots has yet to be realized.

To further this goal, we design a set of generalizable robotic skills in **CLiMBench** to support heterogeneous multi-agent collaboration, leveraging robots’ low-level control capabilities for high success rates and reducing the impact of execution failures on higher-level task planning.

2.2 TASK PLANNING WITH LLMs IN ROBOTICS

LLM Planner for Robotics. The rapid progress of LLMs in generalization and commonsense reasoning has fueled growing interest in robotics, as their strong few-shot Brown et al. (2020); Madaan et al. (2022) and zero-shot Huang et al. (2022); Kojima et al. (2022) learning capabilities make them well-suited as task planners for robots. Reliable code-generation abilities further allow LLMs to synthesize precise, executable instructions for robotic control Liang et al. (2023a); Singh et al. (2023); Wang et al. (2023); Wu et al. (2023a), and value-function-based approaches Lin et al. (2023); Ahn et al. (2022) leverage these models to select robust, skill-level commands for robotic agents. Recent improvements in context-driven prompting strategies Zhang et al. (2024b); Mandi et al. (2023); Liu et al. (2025); Wu et al. (2023b) have strengthened LLM-based task planning even further. Moreover, some studies Mandi et al. (2023) demonstrate that LLMs can reason and plan directly in 3D joint space, enabling the generation of fine-grained and precise task instructions.

Multi-LLM Task Planning. A promising way to overcome the limits of a single LLM in complex reasoning is to use multiple LLMs with cooperation, employing strategies such as round-table discussion Chen et al. (2023a), mutual debate Liang et al. (2023b), and role assignment Hong et al. (2024) to divide labor and improve output reliability. In embodied tasks, many studies emphasize the use of feedback Mandi et al. (2023); Liu et al. (2025) and memory modules Zhang et al. (2024b); Mandi et al. (2023); Liu et al. (2025); Wang et al. (2023) to enhance multi-LLM perception and planning. These modules allow LLMs to generate execution-level feedback and refine planning decisions using the rich context stored in well-designed memory components.

Decision Paradigms in Multi-Robot Collaboration. Two primary decision-making paradigms have emerged for complex multi-robot tasks: centralized and decentralized approaches. In decentralized schemes, multiple models or agents communicate, exchange intermediate plans, and iteratively refine their decisions through structured dialogue Mandi et al. (2023); Zhang et al. (2024b); Liu et al. (2024b), while centralized methods typically rely on a single, large-scale LLM to decompose global objectives and allocate tasks when planning Kannan et al. (2023); Liu et al. (2025). A recent comparative study conducted across four diverse multi-agent 2D scenarios Chen et al. (2023b) further reports that centralized communication consistently achieves higher success rates and markedly greater token efficiency, highlighting its strong potential for scalable real-world deployment.

To enhance collaboration in multi-robot scenarios, we propose a multi-LLM cooperation framework inspired by human teamwork. Robots are organized into dynamic subgroups for specific sub-tasks, reducing communication overhead while enabling concurrent discussions, plan refinement, and parallel action execution to improve efficiency and maintain a high success rate.

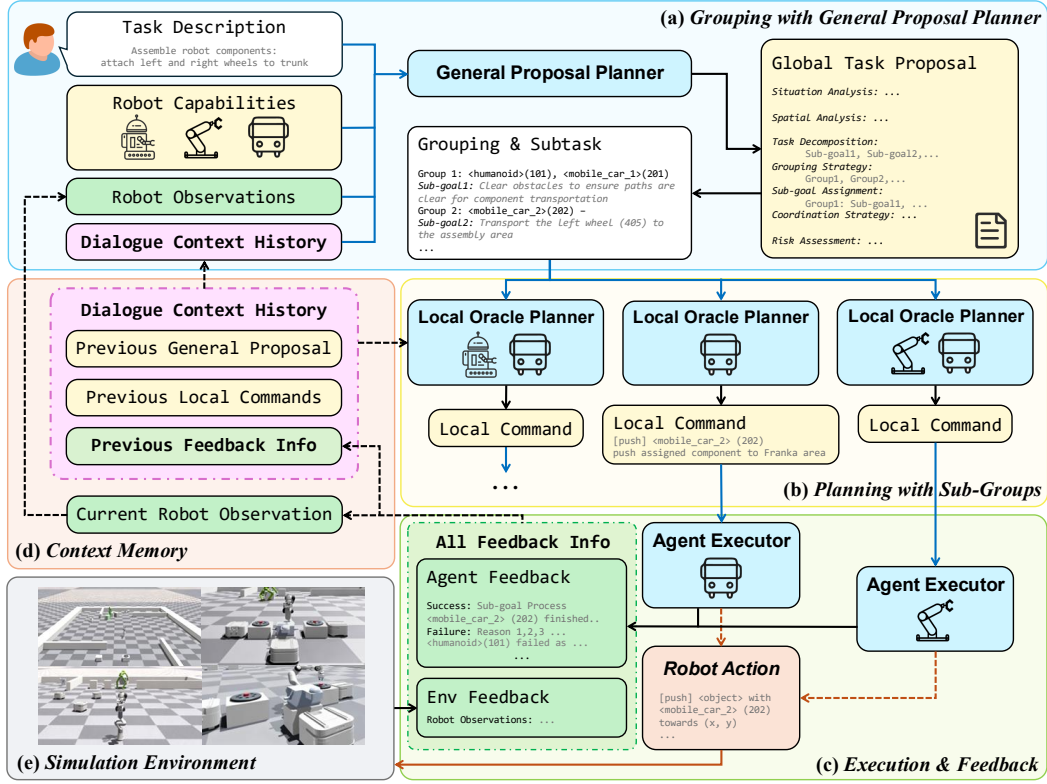


Figure 2: **CLiMRS Framework.** To employ our grouping–planning–feedback–execution cycle, **CLiMRS** comprises (a) a general grouping module, (b) multiple local planners, (c) multiple agent execution and feedback modules, (d) a context memory module, and (e) a simulation environment.

3 METHOD

In this section, we present **CLiMRS**, an adaptive negotiation-driven multi-LLMs cooperation framework for heterogeneous robot systems. Inspired by human teamwork, our approach forms dynamic agent sub-groups that facilitate centralized discussions on robot perception in parallel, with each robot paired with an individual LLM agent to give feedback to these discussions, resulting in a dynamic grouping–planning–feedback–execution cycle. As illustrated in Fig. 2, **CLiMRS** comprises five core modules: (a) a general grouping module that forms dynamic agent groups, (b) multiple local planners that generate agent commands, (c) agent execution and feedback modules that produce robot skills and return execution feedback, (d) a context memory module that records all inter-agent dialogues, and (e) a simulation environment for real-time interaction.

3.1 GROUPING WITH GENERAL PROPOSAL PLANNER

The first stage of our grouping–planning–feedback–execution cycle is to dynamically partition the agents into sub-groups, each responsible for different aspects of the overall task. To achieve this, we use a *general proposal planner* to augment the task instructions and orchestrate the grouping process.

General Proposal Planner. As illustrated in Fig. 2(a), the *general proposal planner* generates a global task proposal that organizes all agents into sub-task-oriented teams. Given the overall task instruction, this prompted LLM incorporates robot capabilities, current observations, and the dialogue history through a structured prompt. It outputs a well-defined plan designed to facilitate systematic reasoning: (1) *Situation Analysis*, assessing the environment and the current progress of the task; (2) *Spatial Analysis*, accounting for the locations of agents and known objects, as well as spatial constraints; (3) *Task Decomposition*, breaking the objective into executable sub-tasks; (4) *Grouping Strategy*, deciding how to cluster agents for concurrent or parallel work while minimizing interference;

(5) *Sub-goal Assignment*, specifying the objective of each group; (6) *Coordination Strategy*, outlining inter-group synchronization and execution order; and (7) *Risk Assessment*, identifying potential conflicts and corresponding mitigation plans. The resulting mapping from agent groups to their designated sub-tasks is then extracted and passed to the perception and execution modules.

3.2 PLANNING WITH SUB-GROUP LOCAL PLANNERS

Given the agent groupings and their designated sub-tasks, the second stage of our cycle issues precise commands to individual robots according to their capabilities and current observations. Because these sub-tasks are mutually independent, multiple *local oracle planners* operate in parallel to generate commands for different robots simultaneously, which is shown in Fig. 2(b).

Local Oracle Planner. The *local oracle planner* facilitates a centralized discussion among robots in a sub-group to determine precise commands for completing their assigned sub-tasks. This discussion leverages prior agent feedback stored in the dialogue context history. Similar to the *general proposal planner*, the *local oracle planner* takes into account sub-task instructions, robot capabilities, partial observations, and historical dialogue as context, but operates within a narrower scope to make fine-grained decisions focused on individual agents executing specific skills.

3.3 AGENT EXECUTION AND FEEDBACK

With commands issued to the robots, the final two stages of our cycle require them to evaluate these commands, determine appropriate actions, and provide feedback to refine future planning while ensuring safe execution. The *agent executor* LLM verifies the feasibility of its command and issues the corresponding action only when the command is deemed executable. The feedback then consolidates outcomes from both the LLMs and the simulator, gathering information to guide subsequent planning cycles and thereby closing the loop of negotiation among the LLMs.

Agent Execution with Feedback. Shown in Fig. 2(c), the *agent executors* verify and execute commands from *local oracle planner* while providing feedback. Each *agent executor* LLM considers its robot’s capabilities, current observations, and available actions. The executor first checks its feasibility against the robot’s physical constraints and observations. If feasible, the action is executed using the robot’s skills; otherwise, the robot remains idle in this loop. Simultaneously, the executor produces feedback based on its evaluation, which is sent to the feedback module to inform future planning. Execution failures are categorized as (1) *improper grouping*: no robot in the group can complete the sub-task; (2) *incorrect agent selection*: a valid sub-task is assigned to an unsuitable robot; and (3) *state inconsistency*: missing information or unmet conditions prevent execution. For successful actions, the module also evaluates whether the sub-task has been fully accomplished.

Feedback Formation. The feedback is aggregated from two sources: (1) environmental observations updated after robot actions are executed in the simulator, and (2) outputs from the *agent executors*. This information is then integrated into the *context memory* for the next grouping–planning–feedback–execution cycle. The feedback both guides the *general proposal planner* during grouping (Sec. 3.1) and aids centralized discussions by the *local oracle planners* (Sec. 3.2). In this way, the accumulated observations and executor outputs provide essential context for refining both the global task proposal and the detailed local commands.

3.4 CONTEXT MEMORY AND ENVIRONMENT

Following the grouping–planning–feedback–execution cycle described above, our framework depends on two essential modules to make the workflow of the entire cycle operate smoothly: the *context memory* module and the *simulation environment*.

Context Memory. As shown in Fig. 2(d), the *context memory* collects (1) current feedback and planning dialogue together with the dialogue history from previous cycles, (2) robot observations from the *simulation environment*, and (3) the latest outputs from agents and planners. For the *general proposal planner*, it retains the previous five dialogue turns and the newest observations, allowing agent feedback to inform new proposals and groupings. For the *local oracle planners*, it stores each group’s latest observations and the last five dialogue turns, providing rich situational context to guide and refine subsequent planning decisions.

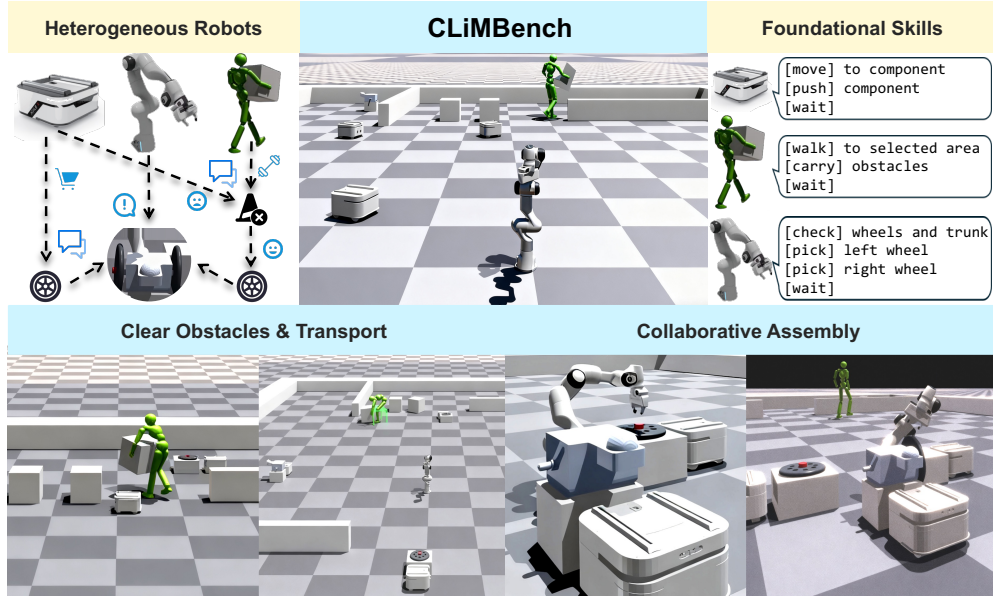


Figure 3: **CLiMBench Benchmark**. **CLiMBench** is a heterogeneous multi-robot collaboration benchmark designed to evaluate **CLiMRS**. It features multi-agent robots with diverse skills, enabling collaboration on tasks like transportation, conveyance, and assembly across varying difficulty levels.

Simulation Environment. Shown in Fig. 2(e), the simulation environment serves as the execution backbone of our framework. It receives the robot skill execution signals issued by the *agent executors* and immediately carries out the corresponding low-level actions in real time. During execution, it monitors the evolving state of the environment and produces both updated robot observations and environment-level feedback. These outputs are fed back to the *context memory*, allowing the overall method to track task progress, refine its understanding of the environment, and supply the information required for the next round of the grouping–planning–feedback–execution cycle.

4 BENCHMARK

In this section, we present **CLiMBench**, a benchmark for heterogeneous multi-robot collaboration. As shown in Fig. 3, we construct an assembly environment in IsaacGym Makoviychuk et al. (2021) that features diverse robotic agents and modular components. To enable effective integration with LLM-based planning, robot actions are executed by invoking predefined skills.

Unlike some other multi-agent collaboration benchmarks Liu et al. (2025), which decouple skill execution from the planning–execution loop and assume that all robot skills succeed by default, **CLiMBench** executes every robot skill within a realistic physics simulation, enabling genuine interaction between planning and execution. This distinction is critical because collaborative assembly tasks are inherently difficult, demanding not only high-precision manipulation but also effective coordination among multiple agents. The following subsections describe the scene construction and skill design mechanisms of **CLiMBench**, and additional details are provided in Appendix B.

4.1 SCENE CONSTRUCTION IN CLiMBENCH

CLiMBench features an industrial assembly scene that includes both assembly components and robotic agents. To increase task complexity and enhance realism, we introduce blocking obstacles into the environment settings. As is illustrated in Fig. 3, our robotic arm is implemented using a Franka Emika Panda arm, the AGV platform is based on the TRACER Mini robot, and the humanoid is implemented using the virtual humanoid agent.

Scene Initialization and Randomization. We initialize the environment and introduce controlled variations in task parameters and object configurations to enhance generalization. At the start of

Table 1: **Robot Skill List in CLiMBench.** We assign each robot type a distinct set of skills in **CLiMBench** based on its specific capabilities.

Robot type	Num	Skill list
Robotic Arm	1	[check] <franka>check <trunk> [check] <franka>check <left wheel> [check] <franka>check <right wheel> [pick] <franka>pick and place <left wheel>on <trunk> [pick] <franka>pick and place <right wheel>on <trunk> [wait] <franka>wait
AGV	3	[move] <mobile_car>move to component location using RRT path [push] <mobile_car>push selected component to franka area [wait] <mobile_car>wait
Humanoid	1	[walk] <humanoid>move to selected area [carry] <humanoid>carry <obstacles> [wait] <humanoid>wait

each episode, robots execute their skills under randomized task conditions, leading to diverse skill sequences and varying levels of inter-agent synchronization. This setup provides a robust testbed for evaluating the effectiveness of different LLM architectures in multi-agent collaborative tasks.

Environment feedback. We design the environment feedback along two dimensions: (1) updating the state of all agents and the coordinates of objects within their perceptible range, and (2) reporting conflicts that arise when multiple robots execute skills simultaneously.

4.2 ROBOT SKILL DESIGN IN CLiMBENCH

In **CLiMBench**, each robot receives both the global task objectives and observations pertinent to its specific skill set (e.g., a humanoid robot observes its joint states, torso status, and target positions). This requirement makes it essential to clearly specify how each agent’s designated skills are implemented in practice in **CLiMBench**. Summarized in Table 1, we design distinct skill sets for different types of robots, with some other details provided in Appendix B.

Robotic Arm Manipulation with Franka. We employ a two-stage control strategy to balance speed and precision. The Franka arm first executes a rapid coarse motion, then slows for fine adjustment to ensure accurate placement. An operational-space controller (OSC) uses the task-space inertia matrix and gravity compensation to compute joint torques, yielding a spring–damper response Narang et al. (2022). Smooth, continuous waypoints are generated by interpolation for reliable execution.

AGV Transportation with TRACER Mini Robot. The robot uses the Rapidly-exploring Random Tree (RRT) algorithm to locate disassembled components and transport them to the destination. The resulting path is executed via differential drive control, enabling smooth turns with the AGV robot. During delivery, the planned route is constrained to straight-line motion to enhance transportation reliability and ensure accurate placement at the assembly location.

Humanoid Carrying Skills. We formulate physics-based humanoid control as a goal-conditioned reinforcement learning problem and adopt the AMP-based single-object manipulation paradigm from previous research Peng et al. (2021); Gao et al. (2024). Style rewards encourage rapid postural dynamics such as quick recovery and linear locomotion, while target rewards guide precise object manipulation, enabling the humanoid to learn efficient carrying behaviors.

5 EXPERIMENTS

In this section, we present a comprehensive evaluation of **CLiMRS** to address the following questions:

- (1) Is **CLiMRS** effective for simple daily-life multi-robot collaboration?
- (2) Can **CLiMRS** perform well in challenging industrial scenarios with multi-robot assembly tasks?
- (3) Through ablation studies, how critical are the individual components of **CLiMRS**?

Table 2: **Comparison Across Task Types in the COHERENT Benchmark.** CLiMRS outperforms all the baselines, achieving the largest gain on the most challenging trio-type tasks.

Method	Mono-type Task		Dual-type Task		Trio-type Task		Average	
	SR	AS	SR	AS	SR	AS	SR	AS
DMRS-1D	0.700	10.6	0.467	18.0	0.667	20.7	0.600	17.2
DMRS-2D	0.500	11.5	0.267	19.9	0.400	24.5	0.375	19.6
CMRS	0.900	7.9	0.533	16.4	0.533	22.2	0.625	16.5
Primitive MCTS	0.000	14.0	0.000	21.5	0.000	26.9	0.000	21.7
LLM-MCTS	0.700	10.2	0.067	20.9	0.000	26.9	0.200	20.5
COHERENT	0.900	7.4	1.000	11.9	1.000	16.1	0.975	12.4
CLiMRS(Ours)	0.900	6.8	1.000	11.5	1.000	13.1	0.975	10.9
Ground Truth (GT)	–	6.5	–	10.3	–	12.9	–	10.3

Table 3: **Comparison Across Scenes in the COHERENT Benchmark.** CLiMRS outperforms all the baselines in every scene, demonstrating its superior performance.

Method	S1		S2		S3		S4		S5		Average	
	SR	AS	SR	AS	SR	AS	SR	AS	SR	AS	SR	AS
DMRS-1D	0.500	17.4	0.625	15.8	0.625	18.3	0.750	15.1	0.500	19.3	0.600	17.2
DMRS-2D	0.500	18.9	0.500	18.3	0.375	20.6	0.250	18.9	0.250	21.1	0.375	19.6
CMRS	0.875	13.1	0.625	16.6	0.625	18.5	0.375	18.1	0.625	15.9	0.625	16.5
Primitive MCTS	0.000	21.5	0.000	21.8	0.000	22.5	0.000	20.5	0.000	22.0	0.000	21.7
LLM-MCTS	0.250	20.0	0.250	20.4	0.250	21.3	0.125	19.9	0.125	20.9	0.200	20.5
COHERENT	1.000	13.1	1.000	11.4	1.000	11.9	1.000	11.4	0.875	14.0	0.975	12.4
CLiMRS(Ours)	1.000	10.8	1.000	10.4	1.000	11.8	1.000	10.4	0.875	11.4	0.975	10.9
Ground Truth (GT)	–	10.3	–	10.4	–	10.8	–	9.8	–	10.5	–	10.3

We evaluate **CLiMRS** in two distinct environments: **CLiMBench** and a simpler heterogeneous multi-robot collaboration benchmark from COHERENT Liu et al. (2025). For LLM api use, we use *gpt-4-0125-preview* to align with the setting in COHERENT. For quantitative analysis, we use task Success Rate (SR) and Average Step (AS) as evaluation metrics in this paper.

5.1 EVALUATING CLiMRS ON SIMPLE DAILY-LIFE MULTI-ROBOT COLLABORATION

To answer Question (1), we evaluate **CLiMRS** on the COHERENT benchmark, a simpler heterogeneous multi-robot benchmark that includes diverse tasks across five real-world scenes, but involves at most three heterogeneous robots and assumes perfect skill execution. We adopt its evaluation metrics and use the reported results as our baseline.

Results shown in Table 2 and 3 suggest that **CLiMRS** succeeds on nearly all COHERENT tasks and achieves higher efficiency with fewer steps. This trend holds across every scene, demonstrating our **CLiMRS**’ superior performance. Notably, in the most challenging trio-type tasks, which require all three robots to collaborate, **CLiMRS** delivers the largest gain, reducing the Average Step count by 18.6%, indicating that our approach offers stronger improvements on more complex tasks.

5.2 EVALUATING CLiMRS ON CLiMBENCH WITH ROBOT ASSEMBLY TASKS

To answer Question (2), we evaluate **CLiMRS** on **CLiMBench**. Our baselines include the following:

- DMRS-1D: a variant of CoELA Zhang et al. (2024b), this decentralized framework lets robots determine their next step through dialogue, with the final decision summarized by the last robot.
- CMRS: a primitive centralized system Huang et al. (2022) that uses a single decision-making LLM to output executable actions, where all information is stored in the prompt.
- COHERENT: an approximately centralized approach combining an oracle planner LLM and feedback LLM for robots, where dialogue is passed through memory, forming a Proposal–Execution–Feedback–Adjustment cycle.

Table 4: **Comparison Across Tasks in CLiMBench.** CLiMRS outperforms all our baselines and reduces the Average Step (AS) by over 40%.

Method	Task 1 (Easy)		Task 2 (Easy)		Task 3 (Hard)		Task 4 (Hard)		Average	
	SR	AS	SR	AS	SR	AS	SR	AS	SR	AS
DMRS-1D	0.000	15.0	0.000	15.0	0.000	19.0	0.000	19.0	0.000	17.0
CMRS	0.000	15.0	0.000	15.0	0.000	19.0	0.000	19.0	0.000	17.0
COHERENT	1.000	13.6	0.800	13.6	0.400	18.2	0.600	17.8	0.700	15.8
CLiMRS (Ours)	1.000	8.2	1.000	8.4	1.000	9.4	1.000	9.2	1.000	8.8
Ground Truth (GT)	–	7.0	–	7.0	–	9.0	–	9.0	–	8.0

Table 5: **Ablation Studies.** Removing dialogue history, feedback information, or the grouping stage significantly reduces both Success Rate (SR) and Average Step (AS).

Method	Task 1 (Easy)		Task 2 (Easy)		Task 3 (Hard)		Task 4 (Hard)		Average	
	SR	AS	SR	AS	SR	AS	SR	AS	SR	AS
CLiMRS w/o history	0.000	15.0	0.000	15.0	0.000	19.0	0.000	19.0	0.000	17.0
CLiMRS w/o feedback	0.200	14.8	0.200	14.8	0.200	18.8	0.200	18.8	0.200	16.8
CLiMRS w/o grouping	0.600	14.0	0.800	13.2	0.600	17.2	0.600	17.4	0.650	15.5
CLiMRS (Ours)	1.000	8.2	1.000	8.4	1.000	9.4	1.000	9.2	1.000	8.8
Ground Truth (GT)	–	7.0	–	7.0	–	9.0	–	9.0	–	8.0

For quantitative evaluation, we fixed the scene parameters and selected four representative scenarios, manually deriving minimal-step solutions as ground-truth references. A task is deemed successful only if completed within twice the ground-truth step count. Due to stochastic skill execution in **CLiMBench**, we run each task five times and report mean Success Rate (SR) and Average Step (AS).

Results in Table 4 show that **CLiMRS** achieves 100% success in **CLiMBench**, surpassing every baseline. It also reduces the Average Step (AS) by 44.30% compared with the best baseline, a substantial efficiency gain highlighting the strength of **CLiMRS** for long-horizon heterogeneous multi-robot collaboration. Moreover, comparing baseline performance in Tables 2 and 4 reveals that the assembly tasks in **CLiMBench** are more challenging than those in the COHERENT benchmark, demonstrating the value of **CLiMBench** as a tougher testbed for heterogeneous multi-robot systems.

5.3 ABLATION STUDIES ON CLiMRS

To answer Question (3), we assess the necessity of each component of **CLiMRS** through: (i) removing the dialogue history, (ii) removing the feedback information, and (iii) removing the grouping stage from the grouping–planning–feedback–execution cycle. We use the same evaluation tasks and metrics as in Section 5.2, and the results are reported in Table 5. The results show that removing any of these components lowers the task success rate and markedly increases the average steps, underscoring the crucial roles of dialogue history, feedback information, and the grouping stage in our method.

6 CONCLUSION

In this paper, we present **CLiMRS**, a human-team-inspired adaptive negotiation paradigm for heterogeneous multi-robot systems. To evaluate these capabilities, we introduce **CLiMBench**, a heterogeneous multi-robot benchmark of challenging assembly tasks. Extensive experiments suggest that **CLiMRS** surpasses all baselines, boosting success rates and improving efficiency by over 40% on more complex tasks. Our results demonstrate that leveraging human-inspired group formation and negotiation principles markedly enhances the efficiency of heterogeneous multi-robot collaboration.

Discussion and Limitation. In this paper, we primarily aim to enhance the efficiency of multi-robot collaboration, while leaving inference latency and computational cost of the LLMs outside the present scope. Managing API costs under inference-efficiency constraints and exploring asynchronous inference–execution are promising aspects that we plan to investigate in future work.

ETHICS STATEMENT

Our study investigates multi-robot collaboration using large language models (LLMs) for planning and negotiation. Below we address the main ethical considerations relevant to this work:

- **No Human or Sensitive Data.** Our research involves only simulated robotic environments and does not include human subjects, personally identifiable information, or sensitive real-world data.
- **Safety and Deployment.** Although our benchmark and methods are evaluated only in simulation, real-world deployment of autonomous multi-robot systems may present physical-safety risks. Any future use outside simulation should therefore incorporate rigorous testing, appropriate safety protocols, and adherence to all relevant regulations.
- **Potential Bias and Fairness.** The LLMs used are pretrained by third parties and may inherit societal biases. Our work does not amplify these biases in deployment scenarios; nevertheless, we acknowledge this limitation and recommend further bias auditing for any real-world applications.

The authors affirm compliance with the ICLR Code of Ethics and accept full responsibility for the integrity and societal implications of the research.

REPRODUCIBILITY STATEMENT

We have taken extensive measures to ensure the reproducibility of our results. A full description of the **CLiMRS** framework is provided in Sec. 3. Details on the **CLiMBench** are provided both in Sec. 4 and in the Appendix. Evaluation protocols are reported in Sec. 5. An anonymous link to the source code and the Appendix is included in the supplementary material, allowing reproduction.

REFERENCES

- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.
- Guillaume Bellegarda, Milad Shafiee, and Auke Ijspeert. Visual cpg-rl: Learning central pattern generators for visually-guided quadruped locomotion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1420–1427. IEEE, 2024.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.
- Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*, 2023a.
- Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? *arXiv preprint arXiv:2309.15943*, 2023b.
- Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

-
- Jiawei Gao, Ziqin Wang, Zeqi Xiao, Jingbo Wang, Tai Wang, Jinkun Cao, Xiaolin Hu, Si Liu, Jifeng Dai, and Jiangmiao Pang. Coohoi: Learning cooperative human-object interaction with manipulated object dynamics. In *Advances in Neural Information Processing Systems*, 2024.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VtmBAGCN7o>.
- Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models. *arXiv preprint arXiv:2309.10062*, 2023.
- Abhishek Kumar Kashyap and Dayal R Parhi. Particle swarm optimization aided pid gait controller design for a humanoid robot. *ISA transactions*, 114:306–330, 2021.
- Sotaro Katayama, Masaki Murooka, and Yuichi Tazaki. Model predictive control of legged and humanoid robots: models and algorithms. *Advanced Robotics*, 37(5):298–315, 2023.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.
- Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4501–4510, 2020.
- Matthew Lai, Keegan Go, Zhibin Li, Torsten Kröger, Stefan Schaal, Kelsey Allen, and Jonathan Scholz. Roboballet: Planning for multirobot reaching with graph neural networks and reinforcement learning. *Science Robotics*, 10(106):eads1204, 2025.
- Yiming Li, Dekun Ma, Ziyang An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 7(4):10914–10921, 2022.
- Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, 44(5):840–888, 2025.
- 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, 2023a. doi: 10.1109/ICRA48891.2023.10160591.
- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023b.
- Wenlong Liang, Rui Zhou, Yang Ma, Bing Zhang, Songlin Li, Yijia Liao, and Ping Kuang. Large model empowered embodied ai: A survey on decision-making and embodied learning. *arXiv preprint arXiv:2508.10399*, 2025.

-
- Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, Nov 2023. ISSN 1573-7527. doi: 10.1007/s10514-023-10131-7. URL <https://doi.org/10.1007/s10514-023-10131-7>.
- 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. IEEE, 2025.
- Xinzhu Liu, Peiyan Li, Wenju Yang, Di Guo, and Huaping Liu. Leveraging large language model for heterogeneous ad hoc teamwork collaboration. *arXiv preprint arXiv:2406.12224*, 2024a.
- Xinzhu Liu, Peiyan Li, Wenju Yang, Di Guo, and Huaping Liu. Leveraging large language model for heterogeneous ad hoc teamwork collaboration, 2024b. URL <https://arxiv.org/abs/2406.12224>.
- Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.
- Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners, 2022. URL <https://arxiv.org/abs/2210.07128>.
- Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5442–5451, 2019.
- Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
- Zhao Mandi, Shreya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023.
- Zhao Mandi, Shreya 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. IEEE, 2024.
- Christopher E Mower, Yuhui Wan, Hongzhan Yu, Antoine Grosnit, Jonas Gonzalez-Billandon, Matthieu Zimmer, Jinlong Wang, Xinyu Zhang, Yao Zhao, Anbang Zhai, et al. Ros-llm: A ros framework for embodied ai with task feedback and structured reasoning. *arXiv preprint arXiv:2406.19741*, 2024.
- Yashraj Narang, Kier Storey, Ireteyayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, Ankur Handa, and Dieter Fox. Factory: Fast contact for robotic assembly, 2022. URL <https://arxiv.org/abs/2205.03532>.
- Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4): 1–20, July 2021. ISSN 1557-7368. doi: 10.1145/3450626.3459670. URL <http://dx.doi.org/10.1145/3450626.3459670>.
- Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. Reasoning with large language models, a survey. *CoRR*, 2024.
- Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- Sahar Salimpour, Lei Fu, Farhad Keramat, Leonardo Militano, Giovanni Toffetti, Harry Edelman, and Jorge Peña Queralta. Towards embodied agentic ai: Review and classification of llm-and vlm-driven robot autonomy and interaction. *arXiv preprint arXiv:2508.05294*, 2025.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Vmp: Versatile motion priors for robustly tracking motion on physical characters. In *Computer graphics forum*, volume 43, pp. e15175. Wiley Online Library, 2024.
- Jiyuan Shi, Chenjia Bai, Haoran He, Lei Han, Dong Wang, Bin Zhao, Mingguo Zhao, Xiu Li, and Xuelong Li. Robust quadrupedal locomotion via risk-averse policy learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11459–11466. IEEE, 2024.
- Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.
- Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- Weizheng Wang, Le Mao, Ruiqi Wang, and Byung-Cheol Min. Multi-robot cooperative socially-aware navigation using multi-agent reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12353–12360. IEEE, 2024.
- Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 2023a.
- Yue Wu, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Yuanzhi Li, Tom Mitchell, and Shrimai Prabhumoye. Plan, eliminate, and track – language models are good teachers for embodied agents, 2023b. URL <https://arxiv.org/abs/2305.02412>.
- Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5571–5580. PMLR, 2018.
- Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Shengshan Hu, and Leo Yu Zhang. Badrobot: Jailbreaking llm-based embodied ai in the physical world. *arXiv preprint arXiv:2407.20242*, 3, 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, 2024b. URL <https://arxiv.org/abs/2307.02485>.
- Qiang Zhang, Peter Cui, David Yan, Jingkai Sun, Yiqun Duan, Gang Han, Wen Zhao, Weining Zhang, Yijie Guo, Arthur Zhang, et al. Whole-body humanoid robot locomotion with human reference. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11225–11231. IEEE, 2024c.
- Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657. IEEE, 2019.

A THE USE OF LARGE LANGUAGE MODELS

In preparing this manuscript, we employed large language models (LLMs) solely for language editing and polishing of text drafted by the authors. No LLM was used to generate original scientific content, to formulate claims, or to fabricate data. All conceptual ideas, experimental designs, analyses, and conclusions were conceived and verified by the authors.

During the experimental phase, we investigated the application of LLMs as part of our research itself, and therefore LLM-based components were involved in the experiments described in the paper. However, these uses were confined to the technical scope of the study and did not influence the integrity of the reported results.

The authors take full responsibility for the accuracy and authenticity of all content in this paper.

B ADDITIONAL DETAILS ON CLiMBENCH

B.1 AGENT SKILL AND CLiMBENCH PARAMETERS

Physical Simulation Space. As the multi-LLM planning–feedback–execution cycle drives task progress through agent collaboration, another critical component lies in executing the planned skills within the physical environment. Unlike some other multi-agent-collaboration benchmarks Liu et al. (2025), the tasks in **CLiMBench** must be carried out step by step in physics simulation. This requirement makes it essential to clearly specify how each agent’s designated skills are implemented in practice in **CLiMBench**. The assembly scene in **CLiMBench** supports three configuration types: robotic arm, automated guided vehicle (AGV), and humanoid, working together to assemble structural body elements and wheel modules.

The key parameters of the agents in **CLiMBench** are summarized in Table 6. The following sections detail the skill parameters and capabilities of Franka, TRACER Mini, and the humanoid.

Table 6: Agent Working Parameters in **CLiMBench**.

	Degrees of Freedom	Payload	Maximum Reach	Number
Franka	7	3kg	855mm	1
TRACER Mini	3	80kg	all	3
humanoid	28	–	all	1

Humanoid Parameters. The humanoid have $N_j = 15$ joints. For each joint $j \in \{1, \dots, N_j\}$, we define the joint-specific observation vector as

$$\mathbf{o}_j = [\mathbf{q}_j, \mathbf{p}_j, \mathbf{v}_j, \mathbf{a}_j],$$

where

- $\mathbf{q}_j \in \mathbb{R}^6$ encodes the six degrees of freedom of joint j , capturing its relative rotational and translational configuration,
- $\mathbf{p}_j \in \mathbb{R}^3$ represents the Cartesian position of the joint in the humanoid’s local frame,
- $\mathbf{v}_j \in \mathbb{R}^3$ is the linear and angular velocity of the joint in 3D space,
- $\mathbf{a}_j \in \mathbb{R}^3$ is the corresponding acceleration vector, including both linear and angular components.

The full observation vector of the humanoid is then constructed by concatenating the joint-level observations:

$$\mathbf{o} = [\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_{N_j}] \in \mathbb{R}^{223}.$$

Observations corresponding to the root body and the global reference frame are explicitly excluded, as they represent the global configuration rather than the local state of individual joints. This formulation ensures that \mathbf{o} fully characterizes the local kinematic and dynamic state of the humanoid while remaining independent of its absolute position and orientation in the environment.

Franka Parameters. We employ a numerical inverse kinematics solver to compute the positions of the first seven joints. The base link is initially positioned at $(0, -2.0, 0)$ with an orientation

represented by the quaternion $(0, 0, 0, 1)$. The solver is configured with a maximum of 200 iterations and a residual threshold of 10^{-4} . The Franka operational space controller applies the impedance control law.

$$\mathbf{F} = k_p \mathbf{e} + k_v \dot{\mathbf{e}},$$

where the proportional gain is set to $k_p = 5$, the derivative gain is chosen as $k_v = 2\sqrt{k_p}$ to achieve critical damping, and the gripper closing threshold is set to 0.05 m, taking into account that the object grasping radius of components in **CLiMBench** is 0.03 m. During the Franka assembly, wheels and trunk are aligned via their axles and attached using magnetic attraction within a range of 0.03 m. We utilized Operational-space (OSC) motion controller, which uses the task-space inertia matrix and gravity compensation to generate joint torques, achieving desired spring-damper behavior in task-space Narang et al. (2022). For each manipulation trajectory, an interpolation-based approach is employed to automatically plan operational waypoints, ensuring smooth and continuous motion execution.

Tracer-Mini Parameters. The mobile car paths are planned using the RRT algorithm, taking into account all current obstacles in the environment, including the Franka robot, as part of the collision space. The planner uses a collision-checking resolution of 0.1 mm along each edge, a rewire count of 32, and a path resolution coefficient of 0.1 mm.

Initialization Parameters. The environment spans a rectangular domain, and the initial positions of objects and agents are stochastically assigned as summarized in Table 7. The scene also includes four obstacles of size 1×5 m placed at the four corners, as well as three smaller obstacles of size $1 \times 1 \times 0.5$ m positioned along the main paths traversed by the mobile cars.

Table 7: Randomized initialization of the environment components and agents.

Entity	Initialization Range
Environment domain	$(x, y) \in [-6, 6] \times [-10, 10]$
Components (two wheels and a trunk)	$(x, y) \in \{(4, 8), (-4, 8), (-4, -8), (4, -8)\}$
Mobile cars	$x \in [-3, 3], y \in [-5, 5]$
Humanoid agent	Central region of the environment

B.2 LLM-DRIVEN ROBOT SKILL EXECUTION IN CLiMBENCH

Pattern Matching via Regular Expressions. All action commands follow a standardized format specifying the action type, target object name, object identifier, and a textual description of the action, and are parsed using three key regular expression functions to extract structured information, as illustrated in Fig. 4. Formally, each input command C is mapped to the group identifier G , the agent set $A = \{(a_i, \text{id}_i)\}$, and the target state with object ID O and location ID L as

$$C \xrightarrow{\text{parse}} \{G, A, O, L\}.$$

Four-Stage Verification for LLM Function Calls. Each LLM function call f is validated through a four-stage mapping

$$f \xrightarrow{V} \{v_1, v_2, v_3, v_4\},$$

where v_1 checks format correctness, v_2 enforces semantic consistency, v_3 ensures physical feasibility, and v_4 guarantees safety, thereby ensuring reliable execution of robotic skills.

The verification operator V is implemented as a four-stage pipeline:

1. **Capability assessment.** The system first queries the LLM to determine whether it is capable of performing the requested task; the LLM returns a binary response or confidence score, and requests below a preset capability threshold τ_c are rejected or deferred.
2. **Action selection.** If the capability assessment is positive, the LLM selects an action a^* from the available action set \mathcal{A} , optionally providing a ranked subset or per-action confidence.
3. **Action parsing.** The selected action a^* is processed by an intelligent parsing operator \mathcal{P} , which extracts a structured command $C^{\text{struct}} = \mathcal{P}(a^*)$ containing fields such as action type, target object name, object identifier, agent assignment, and target location.

Single Agent Prompt Templates

Think of yourself as one of the robots #ID(101, 606, 202, 203, or 204), and your available skills are:

Available Skills for Wheeled Robots:

- [#SKILL#] <#ROBOT#> (#ID) move to component location using RRT path

Your role: You are a #character skill description#

The list of available actions you can perform in your current environment is: #ACTIONLIST#

The instructions for the next step of the task is: #INSTRUCTION#

Please choose the best available action to achieve the goal as soon as possible.

Note: If there is an action in the available actionlist that can satisfy the instruction, the first sentence in the output needs to be "YES I CAN."

Note: if there is not an action in the available actionlist that can satisfy the instruction, then output the possible reasons, such as the agent does not have the ability to execute the current instruction and needs the assistance of other types of agents; or the current agent is not in the state of executing this instruction and an additional action needs to be performed as a prerequisite before the instruction can be executed; or the current state already meets the requirements of the task instructions and no other actions need to be performed. The first sentence in the output needs to be "SORRY I CANNOT."

Note: Think of yourself as #ROBOT#, generating content in a first-person conversation.

Let's think step by step, but you should strictly obey the rules above about the first sentence in the output, do not include "*" in the first sentence to affect output parsing.

Figure 4: **Single Agent Prompt Templates.** It illustrates the single-agent prompt templates, where each template encodes the agent’s name, unique identifier, and action specification to ensure structured and unambiguous command interpretation.

4. **Execution verification (judge).** A dedicated judge prompt \mathcal{J} evaluates C^{struct} for format, semantic consistency, physical feasibility, and safety, producing the validation vector $\{v_1, v_2, v_3, v_4\}$ or a pass/fail decision; only commands that satisfy the judge’s criteria are forwarded to the execution module.

Formally, V can be expressed as the composition of the pipeline operators,

$$V = \mathcal{J} \circ \mathcal{P} \circ \mathcal{S}_2 \circ \mathcal{S}_1,$$

where \mathcal{S}_1 and \mathcal{S}_2 denote the capability-assessment and action-selection stages respectively, \mathcal{P} is the parsing operator, and \mathcal{J} is the judge that yields $\{v_1, v_2, v_3, v_4\}$.

B.3 HUMANOID CARRY SKILL TRAINING

Humanoid robots generally possess greater degrees of freedom, enabling them to execute more intricate tasks compared to other types of agents. Compared to the other two robotic types, the humanoid boasts an expansive field of vision and can be trained for object manipulation, further enabling it to navigate obstacles and replicate industrial scenarios. To train control policies that enable humanoid robot to achieve high-level tasks in a natural and life-like manner, we adopt the AMP framework Peng et al. (2021). While this framework aims to optimize the expected cumulative task reward, it introduces a discriminator to encourage the character to produce behaviors similar to those in the dataset by providing a style reward $r^S(s_t, s_{t+1})$. The agent’s reward r_t at each time step t is defined by Peng et al. (2021):

$$r_t = w^G r^G(s_t, \mathbf{g}_t, s_{t+1}) + w^S r^S(s_t, s_{t+1}) \quad (1)$$

where w^G and r^G represent the task reward weights and associated reward functions, respectively, while \mathbf{g}_t denotes the current target. Similarly, w^S and r^S correspond to the style reward weights and respective functions. In developing our approach for humanoid box manipulation, inspired by COHOI Gao et al. (2024), initially developed for multi-agent collaborative object transportation, we have adopted its AMP-based single-object manipulation training paradigm from COHOI. By employing style rewards to refine rapid postural dynamics such as quick standing and linear locomotion, and leveraging target rewards to guide the execution of object manipulation tasks, the humanoid agent is proficiently trained to perform these actions. Additionally, we meticulously

refined the domain randomization parameters to enable the humanoid to adeptly navigate and execute global-scale manipulation tasks.

To ensure the solvability of generated assembly scenarios within multi-agent systems while simultaneously posing challenges for LLMs, the current randomized configurations feature fixed placements for both components and obstacles. Additionally, each agent operates solely on local observations; for example, the AGV detects only dynamic obstacles within its immediate vicinity, whereas the humanoid leverages AGV movement information to preemptively relocate potential obstructions, thereby maximizing parallel execution efficiency.

In accordance with previous studies on utilizing AMP training for single humanoid robots to transport boxes Gao et al. (2024), we adopted the same reward function framework and corresponding weightings. This includes task-specific rewards for walk r_{walk} , held r_{held} , and placement r_{target} , as well as the AMP-styled reward detailed in Peng et al. (2021):

$$r^S(\mathbf{s}_t, \mathbf{s}_{t+1}) = -\log(1 - D(\mathbf{s}_t, \mathbf{s}_{t+1})). \quad (2)$$

The discriminator employs an identical configuration, utilizing data from the ACCAD sub-dataset sourced from AMASS Mahmood et al. (2019), which encompasses actions such as locomotion, lifting, carrying, and placement. The generator’s policy updates are executed using PPO Schulman et al. (2017), primarily owing to its exceptional performance in contemporary control algorithms.

Table 8: Humanoid carrying skill training parameters.

Parameters	Value
num envs	16384
episode length	600
discount factor	0.99
e_clip	0.2
horizon_length	32
minibatch_size	16384
amp_minibatch_size	4096
task_reward_weight	0.5
disc_reward_weight	0.5

Moreover, unlike COHOI, we undertook extensive domain randomization—including object orientation, human alignment, and object proximity—to cultivate a scenario that supports global transport. We conducted training in IsaacGym headless mode on NVIDIA A100, followed by rendered deployment on a workstation with RTX 4060Ti. Our training environment’s hyperparameter configuration is detailed in Table 8.

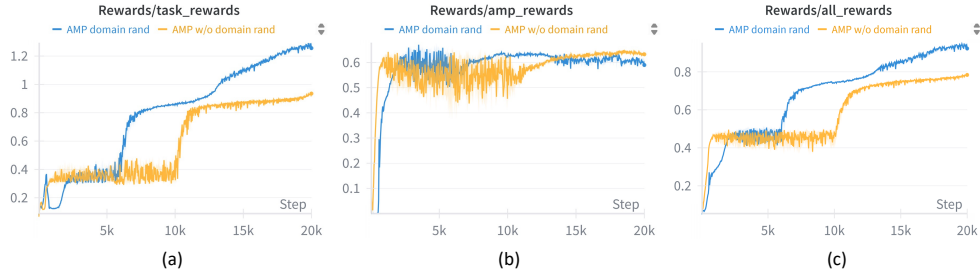


Figure 5: Ablation study comparing the AMP with and without domain randomization.

The training results are illustrated in Figure 5. We compared our approach against the AMP experiment without domain randomization; by approximately 20k steps, the non-randomized AMP had converged to a local optimum and failed to exhibit robust generalization. In contrast, the incorporation of domain randomization consistently enhanced the task rewards, thereby demonstrating superior practical applicability.

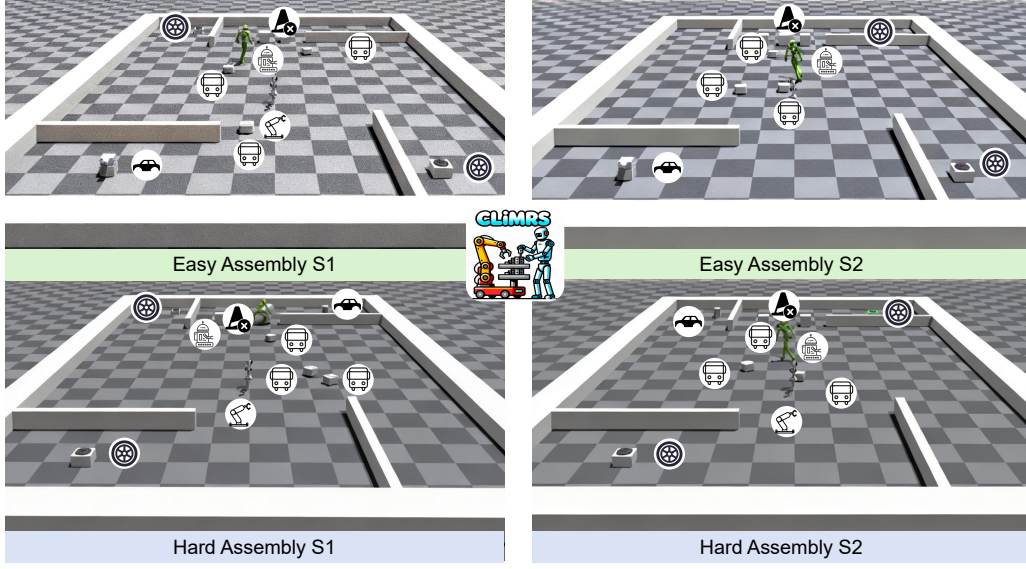


Figure 6: **Different Difficulty Scenarios.** To evaluate the effectiveness of our method in facilitating multi-agent cooperation, we design four heterogeneous agent transport scenarios under two difficulty tiers, namely *easy* and *hard*.

B.4 CLIMBENCH EXPERIMENTS

Different Difficulty Scenarios. In our benchmark, we define two levels of task difficulty: easy and hard, as shown in Fig. 6. The easy configuration assumes an obstacle-free AGV transportation process with initial placements designed to eliminate collision risks with the humanoid agent. Conversely, the hard configuration introduces environmental obstacles along the AGV trajectory and initializes agent positions in a manner that entails potential collision risks with the humanoid.

Dynamic Grouping. As illustrated in Fig. 7, the grouping log records the decision process at each step. At the beginning of the task, the environment is initialized and the LLM is invoked. The agents are then partitioned into groups, with each group pursuing its designated sub-goal independently. As the steps progress, the LLM dynamically reconfigures agent groupings based on each agent’s current observation space, designated role, and remaining skills. The detailed evolution of these groupings is illustrated in C. The LLM is invoked periodically every five simulation steps to produce high-level decisions. During execution, agents performing tasks are subject to real-time collision checking and continuous state feedback; task progress and completion events are recorded in the experiment log for subsequent analysis. Thus, at the next LLM planning call, the agents can be dynamically regrouped based on their current states and the feedback.

Experimental data description. As shown in B.2, we abstract the low-level physical simulation layer of agent execution, such that the commands issued by the LLM can be interpreted as a concrete skill list. This unified abstraction interface further enables seamless integration with different baselines. For each task, we manually design the optimal sequence of execution steps as the ground truth. To evaluate performance, we adopt the $2GT+1$ success metric. Let $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ denote the set of critical sub-goals and $\mathcal{C} \subseteq \mathcal{T}$ the set of sub-goals successfully completed. A task is considered successful if

$$\text{Success} = \begin{cases} 1, & \text{if } |\mathcal{C}| \geq 2|\mathcal{T}| + 1, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

and the overall success rate over N trials is computed as

$$\text{Success Rate} = \frac{1}{N} \sum_{i=1}^N \text{Success}_i. \quad (4)$$

Parallel Execution

```

Group 0: <humanoid>(101) - Sub-goal: Clear obstacles to ensure paths are clear for component transportation
Group 1: <mobile_car_1>(201) - Sub-goal: Transport the left wheel (405) to the assembly area
Group 2: <mobile_car_2>(202) - Sub-goal: Transport the right wheel (406) to the assembly area
Group 3: <mobile_car_3>(203) - Sub-goal: Transport the trunk (303) to the assembly area
Group 4: <robot arm>(606) - Sub-goal: Assemble the components by attaching the wheels to the trunk once all
components are delivered
Processing group 0 message: Hello <humanoid>(101): Please use [carry] skill to clear obstacles.
#####
the first sentence is YES I CAN
#####
No more things to do!
Processing group 1 message: Hello <mobile_car_1>(201): Please use [move] skill to navigate to <left
wheel>(405) location, then use [push] skill to transport it to the franka assembly area.
#####
the first sentence is YES I CAN
#####
[move] <mobile_car_1> (201) move to assigned component location using RRT path
Processing group 4 message: "Hello <franka>(606): Please use [check] skill to check <trunk>(303) for
assembly readiness."
#####
the first sentence is YES I CAN
#####
[check] <franka> (606) check <trunk> (303)

```

Figure 7: **Parallel Execution.** To enhance task execution efficiency, agents within each group are executed in parallel, as reflected in the execution log.

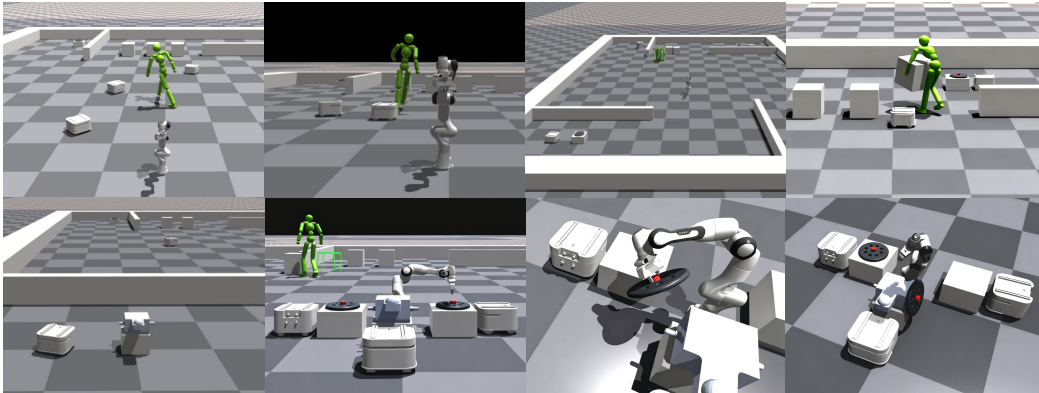


Figure 8: **Simulation.** It shows the assembly process in the dynamic simulation.

Parallel Execution. As illustrated in Fig. 8, the assembly process unfolds dynamically over time, highlighting both the evolution of the environment and the reconfiguration of agent groupings as they execute their assigned tasks. Meanwhile, during task execution, agents within a group collaborate to achieve their sub-goals. For instance, the Humanoid clears obstacles while the AGV performs its transportation tasks, and during the Franka assembly process, if a component is unreachable, the AGV dynamically repositions to facilitate assembly.

C EXAMPLE LOGS AND PROMPTS

We show some examples of logs and prompts in all the figures below.

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

Scene Dynamics Log

You are a structured output formatter for multi-agent task coordination. Your job is to convert a detailed grouping strategy into a precise, parseable format.

Step:0

Assemble robot components: attach left and right wheels to trunk

Group 0: <humanoid>(101) - Sub-goal: **Clear** obstacles to ensure paths are clear for component transportation

Group 1: <mobile_car_1>(201) - Sub-goal: Transport the left wheel (405) to the assembly area

Group 2: <mobile_car_2>(202) - Sub-goal: Transport the right wheel (406) to the assembly area

Group 3: <mobile_car_3>(203) - Sub-goal: Transport the trunk (303) to the assembly area

Group 4: <robot arm>(606) - Sub-goal: Assemble the components by attaching the wheels to the trunk once all components are delivered

Step:1

Assemble robot components: attach left and right wheels to trunk

Group 0: <humanoid>(101) - Sub-goal: **Clear** any obstacles that might prevent the **mobile cars** from reaching and transporting their components

Group 1: <mobile_car_1>(201) - Sub-goal: Transport the left wheel to the franka assembly area

Group 2: <mobile_car_2>(202) - Sub-goal: Transport the right wheel to the franka assembly area

Group 3: <mobile_car_3>(203) - Sub-goal: Transport the trunk to the franka assembly area

Group 4: <robot arm>(606) - Sub-goal: Assemble the components once delivered to the assembly area

Step:2

Assemble robot components: attach left and right wheels to trunk

Group 0: <humanoid>(101) - Sub-goal: Ensure all paths are clear for mobile cars to transport components

Group 1: <mobile_car_1>(201) - Sub-goal: Transport left wheel to the assembly area

Group 2: <mobile_car_2>(202) - Sub-goal: Transport right wheel to the assembly area

Group 3: <mobile_car_3>(203) - Sub-goal: Transport trunk to the assembly area

Group 4: <robot arm>(606) - Sub-goal: Assemble left and right wheels onto trunk when components are ready

Step:3

Assemble robot components: attach left and right wheels to trunk

Group 0: <mobile_car_1>(201), <mobile_car_2>(202), <mobile_car_3>(203) - Sub-goal: Reposition components if not in optimal position for assembly

Group 1: <robot arm>(606) - Sub-goal: **Check** readiness of left and right wheels (405 and 406) for assembly, then assemble onto trunk (303)

Group 2: <humanoid>(101) - Sub-goal: Clear any obstacles if necessary to ensure accessibility of assembly area

Step:4

Assemble robot components: attach left and right wheels to trunk

Group 0: <robot arm>(606) - Sub-goal: **Perform final check** on right wheel readiness and assemble components when ready

Group 1: <mobile_car_2>(202) - Sub-goal: **Push right wheel** to within franka's operational range for readiness check

Group 2: <mobile_car_1>(201), <mobile_car_3>(203) - Sub-goal: **Position** left wheel and trunk **within franka's** operational range for assembly

Group 3: <humanoid>(101) - Sub-goal: Remain on standby to clear any unforeseen obstacles

Step:5

Assemble robot components: attach left and right wheels to trunk

Group 0: <mobile_car_1>(201), <mobile_car_2>(202), <mobile_car_3>(203) - Sub-goal: Transport their designated components to the franka assembly area sequentially to avoid traffic and ensure all components are ready for assembly

Group 1: <robot arm>(606) - Sub-goal: Assemble the robot by attaching the left and right wheels to the trunk once all components are within its operational range

Group 2: <humanoid>(101) - Sub-goal: **Remain on standby** to clear any unforeseen obstacles that might arise during the transport of components

Step:6

Assemble robot components: attach left and right wheels to trunk

Group 0: <robot arm>(606) - Sub-goal: **Assemble** the robot by attaching the left and right wheels to the trunk

Group 1: <humanoid>(101) - Sub-goal: Standby for path clearing and obstacle management if unforeseen obstacles appear

Step:7

Assemble robot components: attach left and right wheels to trunk

Group 0: <humanoid>(101) - Sub-goal: **Ensure** clear paths for mobile cars

Group 1: <mobile_car_2>(202) - Sub-goal: Transport the right wheel to franka assembly area

Group 2: <robot arm>(606) - Sub-goal: **Attach** the right wheel to the trunk

Figure 9: **Scene Dynamics Log**. It illustrates the formation of sub-groups and the corresponding sub-task for each sub-group.

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Humanoid Prompt

Think of yourself as **humanoid (101)**, and your available skills are:

Available Skills for Humanoid (101):

- [walk] <humanoid> (101) move to selected area
- [carry] <humanoid> (101) carry <obstacles> (507)
- [wait] <humanoid> (101) wait

Your role: The humanoid robot has advanced bipedal locomotion and manipulation capabilities. You can walk around the assembly workspace, navigate between different areas, and handle obstacles. You specialize in mobility and can **carry obstacles (507)** to clear paths for other agents. You work collaboratively with wheeled robots and the franka arm to complete assembly tasks.

Note: To perform actions, you must **first [walk]** to the target area. For example, to carry obstacles, you must first walk to the area where the obstacles are located.

Note: Your primary function is obstacle management using the **[carry]** skill for <obstacles> (507). You work collaboratively with **wheeled robots (202, 203, 204)** for component transportation and with **franka (606)** for assembly operations.

Note: You can navigate complex terrain and access areas that wheeled robots might find difficult, making you valuable for clearing paths and removing obstructions.

Note: Use **[wait]** when you need to pause and coordinate with other agents or when your immediate assistance is not required.

Franka Prompt

Think of yourself as **franka (606)**, and your available skills are:

Available Skills for Franka (606):

- [check] <franka> (606) check <trunk> (303)
- [check] <franka> (606) check <left wheel> (405)
- [check] <franka> (606) check <right wheel> (406)
- [pick] <franka> (606) pick and place <left wheel> (405) on <trunk> (303)
- [pick] <franka> (606) pick and place <right wheel> (406) on <trunk> (303)
- [wait] <franka> (606) wait

Your role: You are a fixed manipulator robot arm specialized in precise assembly operations. You can check components and perform pick-and-place operations to assemble the final robot. You work with components that are brought to your workspace by **wheeled robots (202, 203, 204)**. Your primary task is to assemble wheels onto the trunk to complete the robot assembly.

AGV Prompt

Think of yourself as one of the **wheeled robots (202, 203, or 204)**, and your available skills are:

Available Skills for Wheeled Robots:

- [move] <wheeled robot1/2/3> (202/203/204) move to component location using RRT path
- [push] <wheeled robot1/2/3> (202/203/204) push selected component to franka area
- [wait] <wheeled robot1/2/3> (202/203/204) wait

Your role: You are a wheeled robot optimized for efficient component transportation around the assembly workspace. You can navigate to component locations using RRT path planning and push components like <trunk> (303), <left wheel> (405), and <right wheel> (406) to the franka assembly area. You work collaboratively with other wheeled robots, the humanoid, and the franka arm.

Note: Normally you must **first use [move]** skill to navigate to a component's location **before you can [push]** it. However, if your available actions only include [push] and [wait] (no [move] option), this means you have already completed the movement phase and are positioned at the component location. In this case, you can directly use the [push] skill to transport the component to the franka assembly area.

Note: You specialize in ground-level component transportation using RRT path planning. You cannot perform assembly operations - those require the franka (606) robot arm's assistance.

Note: When transporting components, coordinate with other wheeled robots to efficiently move all required components. **Prioritize moving the <trunk> (303) first**, then the wheels <left wheel> (405) and <right wheel> (406).

Note: Your **[push]** action transports components directly to the franka assembly area where the robot arm can access them for final assembly operations.

Note: Use **[wait]** when you need to pause for coordination with other agents or when your immediate assistance is not required.

Figure 10: **Single Agent Prompts.** It illustrates the single-agent prompt design for Franka, Humanoid, and AGV, specifying their roles and available skills.

Agent Grouping Vanilla Prompt

You are an intelligent task coordinator for multi-agent assembly systems. Your job is to analyze the current situation and develop a comprehensive strategy for grouping agents to accomplish the robot assembly task efficiently.

Given the following information:

These robot agents and their **capabilities and action spaces** are:

1. **humanoid (101)**: The humanoid can walk around the environment and carry obstacles to clear paths for other robots.

The humanoid has advanced mobility and can navigate complex terrain.

When other agents cannot reach component locations due to obstacles, humanoids can assist by clearing the path.

The humanoid can carry obstacles but cannot perform component assembly directly.

NOTE: The humanoid should focus on path clearing and obstacle removal tasks.

2. **mobile_car (201/202/203)**: The mobile cars are wheeled robots optimized for component transportation. Each mobile car is assigned to a specific component:

- **mobile_car_1 (201)**: Responsible for left wheel transportation

- **mobile_car_2 (202)**: Responsible for right wheel transportation

- **mobile_car_3 (203)**: Responsible for trunk transportation

The mobile cars can navigate to component locations using RRT path planning and transport components to the franka assembly area.

Mobile cars cannot perform assembly operations directly but are essential for component logistics.

3. **franka robot arm (606)**: The franka is a fixed manipulator specialized for precise assembly operations.

The franka can check components, pick and place wheels on the trunk to complete the final assembly.

The franka can check trunk (303), left wheel (405), and right wheel (406).

The franka performs the final assembly by placing wheels on the trunk.

The franka cannot reach ground-level objects directly and relies on mobile cars to bring components to its workspace.

Available Agents: **#AGENTS_INFO#**

Task Goal: **#TASK_GOAL#**

Current Environment Observations: **#OBSERVATIONS#**

Previous Dialogue History: **#DIALOGUE_HISTORY#**

Pay attention to the dialogue history, because it records your conversations with different agents and the progress of the task execution. Depending on the content of the dialogue, you can avoid repeating actions that have been completed and make more informed grouping decisions based on what has already been accomplished.

Please develop a comprehensive strategy for agent grouping and task coordination. Consider the following aspects:

1. ****Situation Analysis****: Analyze the current environment state, what has been accomplished (based on dialogue history), and what still needs to be done.

2. ****Spatial Analysis****: Consider the physical locations of agents and components, area layouts, and any spatial constraints that might affect agent coordination.

3. ****Task Decomposition****: Break down the overall assembly task into logical sub-tasks that can be assigned to different groups.

4. ****Grouping Strategy****: Explain your reasoning for how to group agents, considering:

- Which agents can work **simultaneously** without interfering with each other

- Which agents need to **collaborate closely** and should be in the same group

- Spatial separation to avoid conflicts

- Efficiency considerations

- Remember that agents going to interact with the same component and in the same area should be in the same group.

5. ****Sub-goal Assignment****: For each proposed group, describe:

- What specific sub-goal they should accomplish

- Why this sub-goal makes sense for this particular combination of agents

- How this sub-goal contributes to the overall assembly task completion

6. ****Coordination Strategy****: Explain how the different groups will coordinate with each other, if needed, and the sequence or timing of their operations.

7. ****Risk Assessment****: Identify potential issues or conflicts that might arise and how your grouping strategy addresses them.

Please provide a detailed explanation on Grouping Strategy and Sub-goal Assignment part of your strategy while keeping the other parts brief. Focus on the reasoning and rationale rather than strict formatting. Be comprehensive in your analysis and make sure to address all aspects of effective multi-agent coordination.

Let's think step by step and develop the best possible strategy for this assembly situation.

Figure 11: **Vanilla Agent Grouping Prompt**. It illustrates the prompt to extract agent sub-groups and their corresponding sub-tasks with the general proposal planner.

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

Agent Grouping Prompt

You are a structured output formatter for multi-agent task coordination. Your job is to convert a detailed grouping strategy into a precise, parseable format.

Based on the comprehensive grouping strategy provided below, please extract and format the agent groups and their sub-goals according to the exact format requirements.

Grouping Strategy: #VANILLA_GROUPING_OUTPUT#

Available Agents: #AGENTS_INFO#

IMPORTANT FORMAT RULES:

1. Each group must be on a single line starting with "Group X:" where X is a number
2. List agents using EXACTLY this format: <agent_class_name>(id)
3. Separate multiple agents with commas
4. After listing agents, use " - Sub-goal: " followed by a clear, specific sub-goal
5. Sub-goals must be actionable and directly related to the task
6. Do not include any extra text, comments or explanations

Example correct format:

Group 1: <humanoid>(101), <mobile_car_1>(201) - Sub-goal: Transport trunk from area A to franka assembly area

Group 2: <franka>(606) - Sub-goal: Assemble left wheel onto trunk when components are ready

For agents that should not participate in this step:

Non-assigned Agent: <mobile_car_2>(202) - Reason: No tasks available at current location

INVALID FORMATS (DO NOT USE):

- Group 1 - The humanoid and mobile car will...
- First group: humanoid(101), mobile_car_1(201)
- Group 1: <humanoid>(101) will walk while <mobile_car_1>(201) moves
- **Group 1:** <humanoid>(101)
- Group 1: <humanoid>(101) - The agent should move to...

Remember:

- Use exact agent class names and IDs from the available agents list
- Keep groups spatially separated to avoid conflicts
- Only include agents that are actually needed for the current step
- Agents from all groups are expected to operate concurrently, so group them together if they need to act sequentially instead of act in parallel
- Provide clear, actionable sub-goals for each group
- Never assign the same agent to multiple groups
- Never include extra comments or explanations
- If previous plan failed, reflect and improve your plan by assigning sub-tasks to proper agent groups
- If part of the plan is already done, you don't need to assign that part of task again.
- Group agents together if they share similar subgoals (e.g. interacting with the same object)

Let's think step by step.

Figure 12: **Agent Grouping Prompt.** It illustrates the prompt to form agent sub-groups with a general proposal with the general proposal planner.