LLM-Powered Multi-Agent Proactive Communication System for Embodied Intelligence

Anonymous ACL submission

Abstract

We presents a novel multi-robot collaboration framework leveraging large language models (LLMs) for improved communication, planning, and execution. By integrating a centralized message pool and LLM-assisted decision-making, our system addresses limitations of existing multi-agent frameworks. Experiments in the MuJoCo simulation environment demonstrate significant improvements in task completion rates, communication effectiveness, and decision-making accuracy. Our proactive communication system reduces redundancy and enhances fault tolerance, enabling efficient handling of unexpected situations. Future work will focus on improving information synchronization and multi-system collaboration, further enhancing efficiency and scalability in complex environments.

1 Introduction

The convergence of robotics and large language models (LLMs) is unlocking new potentials in embodied intelligence, demonstrating significant promise in guiding and understanding complex robotic tasks(Zeng et al., 2023; Wang et al., 2024). Initial advances have successfully integrated LLMs for controlling individual robots, resulting in sophisticated decision-making capabilities and efficient task execution. As the control of single robots via LLMs becomes increasingly refined, the focus is now shifting towards the collaborative efforts of multiple robots.

Multi-robot collaboration promises enhanced efficiency and productivity compared to single-robot operations. However, the coordination and control of multiple robots introduce significant challenges that underscore the critical role of LLMs. Effective multi-robot systems require not just the aggregation of individual robotic capabilities but also seamless communication and coordination to optimize decision-making processes.

Despite significant progress in multi-agent frameworks, their application in robotics remains underexplored and insufficiently sophisticated for real-world deployment. Existing frameworks often fail to address the complexities of robot collaboration, particularly in dynamic and unpredictable environments(Naveed et al., 2024). Key challenges include the insufficient integration of sensor data, inadequate utilization of memory resources, limited communication capabilities, and suboptimal planning and execution strategies. Moreover, many current solutions rely on centralized architectures, which, although effective in some scenarios, do not scale well with an increasing number of agents. These centralized systems are prone to single points of failure and cannot manage the complexity of distributed decision-making required for large-scale robot collaboration(Zhang et al., 2023; Wang et al., 2024).

To address these challenges, we propose a comprehensive multi-agent framework specifically designed for robotic collaboration. The normal framework is structured around five essential components:

1. **Sensor:** Robots gather key data about themselves and their environment, creating the basis for smart decisions.

2. **Memory:** A centralized message pool stores historical decisions, trajectories, and example instructions, which agents can access to enhance task execution efficiency.

3. **Communication:** Agents engage in dialogues to resolve conflicts and finalize decisions, with a leader agent ensuring the accuracy and completeness of the message pool.

4. **Plan/Task Assignment:** The planning process incorporates both centralized supervision and decentralized execution, enabling agents

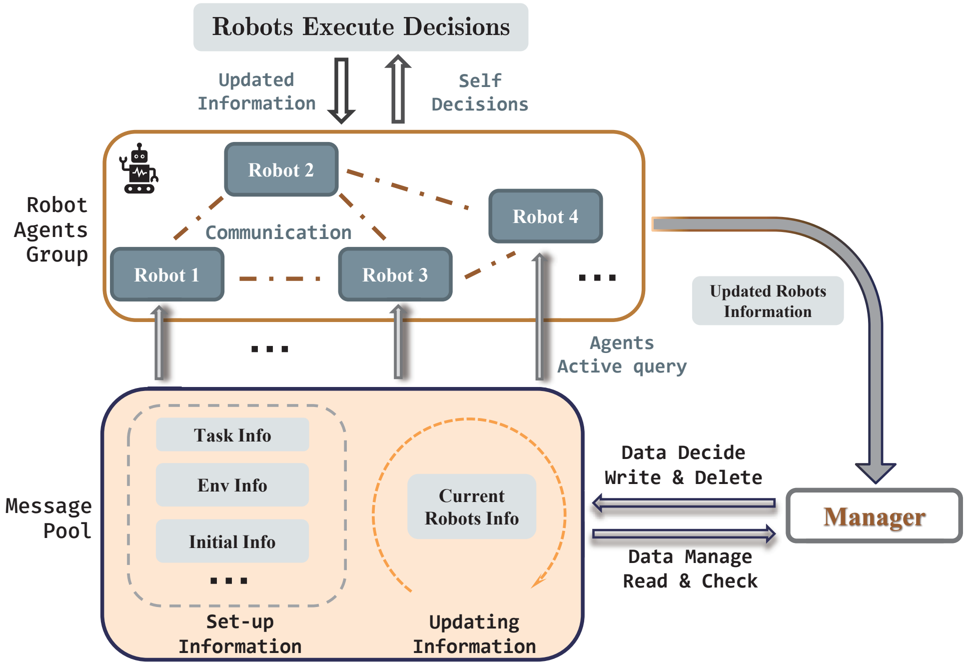

Figure 1: Overview of the multi-robot collaboration system. Robots execute decisions based on collected and self-generated information, communicated through the message pool. The message pool stores initial setup information, environmental data, and task-specific data, which robots actively query and update. The manager oversees data management, including writing, deleting, reading, and checking data, ensuring data integrity and accessibility for effective decision-making and task execution.

to autonomously determine their actions while a leader coordinates the overall strategy.

5. **Execution:** Using a multi-RRT method, robots plan exact paths within their activity ranges, ensuring smooth and conflict-free operations.

By integrating advanced components such as structured memory pools and leveraging LLMs for dynamic code generation, our framework offers a more sophisticated and effective approach to robot collaboration. This hybrid model, combining centralized oversight with decentralized execution, addresses the limitations of current frameworks and lays the groundwork for scalable and resilient multi-robot systems.

2 Related Work

2.1 LLMs for Robotics

Recent advancements in large language models (LLMs) have significantly impacted the field of robotics, enabling the development of more sophisticated and adaptable robotic systems. Initial works, such as SayCan(Ahn et al., 2022b) and Inner Monologue(Huang et al., 2022), utilized LLMs for selecting skill primitives and executing robotic tasks with environment feedback to improve planning. Further research leveraged the code-generation capabilities of LLMs to create robot policies in code format, exemplified by CaP, ProgGPT(Singh et al., 2022), and Demo2Code(Wang et al., 2023a), as well as generating longer programs for robot execution in works like TidyBot(Wu et al., 2023) and Instruct2Act(Huang et al., 2023).

In the realm of motion planning, studies such as Text2Motion(Lin et al., 2023), AutoTAMP(Chen et al., 2024), and LLM-GROP have combined LLMs with traditional task and motion planning (TAMP). Other research has explored the use of LLMs to facilitate human-robot collaboration, design rewards for reinforcement learning (RL), and control real-time motion planning in robotic tasks. However, most prior work has focused on single-robot setups and single-thread LLM planning. In contrast, our work addresses multi-robot settings, using dialog prompting for task reasoning and coordination(Mandi et al., 2023). This approach not only enhances the efficiency and accuracy of task execution but also allows for more dynamic and adaptive responses to changing environments. By leveraging the collaborative capabilities of multiple robots, we aim to achieve more complex and large-scale robotic operations.

2.2 Multi-Modal Prompting for Robotics

LLMs' lack of perception abilities presents a significant bottleneck in their integration with robotic applications. One approach to overcoming this limitation is multi-modal pre-training with both vision, language, and large-scale robot data. The multi-modal pre-trained model PALM-E(Driess et al., 2023) achieves both perception and task planning with a single model, while works like Interactive Language(Ahn et al., 2022a) and DIAL build large datasets of language-annotated robot trajectories(Guhur et al., 2023) for training generalizable imitation policies.

Another solution involves incorporating pre-trained vision-language models (VLMs) such as CLIP. In studies like Socratic Models(Zeng et al., 2022b), Matcha(Jang et al., 2023), and the work by Kwon et al(Kwon et al., 2023)., LLMs are used to query and synthesize information from other models to enhance environmental reasoning. Some works, such as CogLoop(Bai et al., 2023), also explore fine-tuning adaptation layers to better integrate different frozen models. Our research leverages simulation to extract perceptual information, and real-world experiments follow prior work using pre-trained object detection models to generate scene descriptions.

2.3 Dialogue, Debate, and Role-Play LLMs

Beyond robotics, LLMs have demonstrated capabilities in representing agentic intentions and behaviors, facilitating multi-agent interactions in simulated environments such as text-based games and social sandbox scenarios(Li et al., 2023). Recent studies indicate that dialog or debate-style prompting can enhance LLMs' performance on human alignment tasks and a variety of goal-oriented tasks(Wang et al., 2023b; **?**). While prior work has primarily focused on understanding LLM behaviors or solving single questions, our approach requires planning separate actions for each agent, adding complexity to discussions and the difficulty of achieving consensus.

2.4 Multi-Robot Collaboration and Motion Planning

Research on multi-robot manipulation has a long history, with initial efforts focusing on the low-level problem of finding collision-free motion trajectories. Sampling-based methods have been popular(Zeng et al., 2022a), with various algorithmic improvements proposed over time. More recent work has explored learning-based methods as alternatives(Hu et al., 2023). While our tasks are set in relatively static scenes(de Castro and Chaimowicz, 2023), significant research has also addressed more challenging scenarios involving dynamic objects or closed-chain kinematics.

High-level planning to allocate and coordinate sub-tasks is another critical area of multi-robot collaboration research(Guo et al., 2023), which our work is closely related to. Most prior work has tailored their systems to a small set of tasks, such as furniture assembly(Mandi et al., 2023). However, our approach aims to provide a more generalizable and adaptable framework for multi-robot collaboration.

In summary, our work builds upon extensive research in LLMs for robotics, multi-modal pre-training, dialogue and debate LLMs, and multi-robot collaboration. By integrating these advanced components and leveraging large language models for dynamic task planning and execution, our framework offers a novel approach to multi-robot collaboration that addresses the limitations of current systems and provides a scalable solution for complex, real-world applications.

3 Method

3.1 Robot Message Pool

In many current multi-robot collaboration frameworks and environments, such as ROCO(Mandi et al., 2023), the role of large language models is primarily focused on task classification and high-level decision-making. Although these frameworks exhibit certain multi-agent characteristics, robots as part of these agents lack sufficient intelligence. In most cases, existing robots function more like sensors, perceiving environmental information and transmitting it to the LLM for analysis and high-level decision-making. This setup does not foster optimal collaboration. True effective collaboration should involve each robot possessing a certain level of autonomy, actively analyzing information, and sharing and coordinating this information with others, allowing each agent to make independent, non-conflicting decisions. This approach ensures both efficiency and robustness.

We drew inspiration from MetaGPT(Hong et al., 2023). We discovered that information sharing is crucial, especially for complex decision-making tasks involving multiple agents, such as three-dimensional decision-making in robots, which entails significantly more information than simpler two-dimensional scenarios. Therefore, we decided to employ a message pool for aggregating and organizing information, granting each agent proactive access to it. By ensuring the integrity and reliability of the message pool, we can address the critical issue of agents making insufficient or erroneous decisions. The message pool is passively maintained and lacks any proactive capabilities; its writing and reading operations are conducted by the LLM through APIs and code execution. Consequently, the existence of the message pool enables agents to actively think and solve problems independently, which is essential for imparting autonomous intelligence to the agents.

Through testing, we have summarized the core essential information for the message pool, as illustrated in Figure2. Specifically, the message pool needs to include the following information:

- **Self-Information:** The message pool needs to include each robot's individual information, including basic attributes, functionalities, operational range, current position, and current status. This information helps other robots understand the status of each robot for future decision-making. It is also crucial for the allocation of task functions.

- **Task-Specific Information:** Additionally, the message pool needs to include the initial set of task objectives, specifying the tasks that require coordination among multiple robots. These initial settings are permanently stored in the message pool, allowing agents to consult them if there is any uncertainty or if they forget the tasks. Specifically, task objectives include tasks to be completed, such as packaging or collaborative unwrapping of packages.

  When constructing the message pool initially, the manager assigns a set of specific sub-tasks to each robot based on their unique capabilities. These sub-tasks are initial assignments that agents can modify through coordination and feedback to the manager for adjustments. However, such modifications are generally prohibited due to their inherent uncertainty.

  Moreover, we provide an optional historical task completion record for reference. This record can be consulted to review past task completions and derive potential solutions.

- **Environment Information:** Finally, the message pool also needs to include certain manually input environmental information. This includes details about potential obstacles in the scene, their coordinates and basic properties, the state of the environment, any prohibited zones, and the basic 3D information of the environment. This information is stored to provide agents with reference points for decision-making.

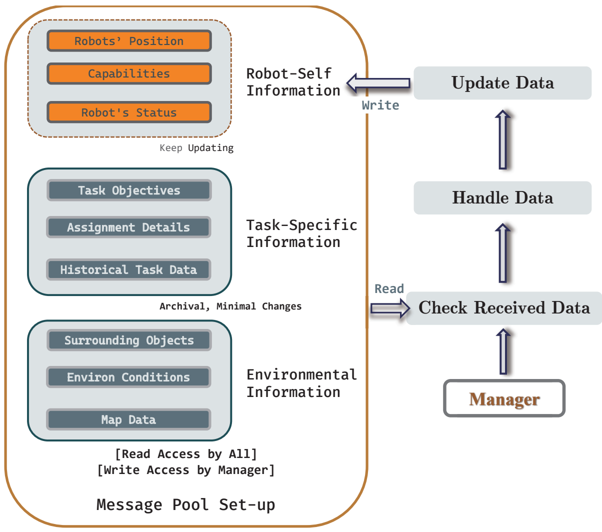

Figure 2: Message pool setup categorized into three main sections: Self-Information, Task-Specific Information, and Environmental Information. Self-Information provides data about the robot itself. Task-Specific Information offers details related to task objectives and assignments. Environmental Information supplies context about the surrounding environment. This setup facilitates comprehensive data storage and effective multi-robot collaboration.

The message pool is stored as a separate file and does not possess any proactive capabilities. However, we provide a set of Python API interfaces for reading and writing to this storage file. These APIs are communicated to the agents at the beginning. Robots have read-only access and are strictly prohibited from writing to the message pool. If a robot agent wants to retrieve specific information, it can write the relevant code, which is then checked and executed by the system to return the information in a specific format. The agent reads the returned information to obtain the desired content, thereby achieving proactive retrieval and reading operations.

By leveraging the message pool throughout the collaboration process, robots can make more informed decisions, adjust their actions dynamically, and continuously learn and improve from past experiences. This comprehensive approach ensures that the multi-robot system operates efficiently and effectively, even in complex and dynamic environments. However, given the large volume of information in the message pool, ensuring its accuracy is crucial. Therefore, a dedicated agent is designated as the project manager, responsible for overseeing the entire project. This manager has exclusive write access, while all other agents are limited to read-only access and cannot write to the message pool independently.

3.2 Data Management

Effective data management is crucial for the message pool, ensuring accuracy, consistency, and accessibility. After completing a round of tasks, robots send critical information to the manager, including their precise individual data, observable new environmental details, and information about nearby robots, all in a standardized format.

The manager first integrates the received data, merging any duplicate information. If discrepancies are found, the manager rejects the data and requests the agents to resend their information. After a preliminary merge, the manager reads related content from the message pool, such as the previous positions of the robots, and compares this data to ensure consistency. Once verified, the manager uses Python code and API calls to write the new information into the message pool, ensuring no conflicts arise. Additionally, a copy of the information is retained for optional comparison to prevent issues during the writing process.

The Manager, acting as an agent, oversees this process by performing several key functions:

- **Data Processing:** The Manager checks the current information in the message pool and identifies any duplicate or outdated data. It deletes redundant entries and formats the new information according to predefined standards to ensure compatibility and ease of access.
- **Data Updating:** The Manager compares the new information with existing data and updates the message pool with the latest data from the robots and the environment. This process ensures that the information remains current and relevant.
- **Conflict Resolution:** In case of conflicting data entries, the Manager resolves conflicts based on predefined rules and the current task context. This helps maintain data integrity and consistency.
- **Security and Fault Tolerance:** The Manager implements strict access control measures to prevent unauthorized modifications to the message pool. Regular backups are maintained to prevent data loss and ensure quick recovery in case of system failures.

By performing these tasks, the Manager ensures the message pool is accurate, consistent, and accessible, enabling efficient multi-robot collaboration and decision-making. The specific details can be found in Figure3.

3.3 Multi-Robot Communication

Effective communication is essential for the coordination and collaboration of multiple robots in our framework. The communication protocols ensure that all robots have access to the necessary information for decision-making and task execution. This section details the communication mechanisms, including the use of the message pool, feedback loops, and the protocols for inter-agent communication.

Our system allows communication between agents. Our tests indicate that such inter-agent communication significantly enhances their cooperative performance and feedback mechanisms. Additionally, agents can preliminarily eliminate redundant or duplicate observation information before sending data to the message pool, facilitating subsequent processing and reducing the manager's burden.

The primary communication between agents occurs during the execution steps and the agents' self-decision processes. When a robot agent decides on the information it needs, it communicates with surrounding robots to clarify its intended actions and understand the actions of other robots. If behavior conflicts arise, agents perform simple calculations to determine the required waiting time to avoid collisions during movement. Subsequent task analysis and decision-making adhere as closely as possible to the initial task classification, with each robot's function fixed at the outset. If issues arise, robot agents communicate and then provide feedback to the manager. Upon identifying the keywords in the response information, the manager updates the

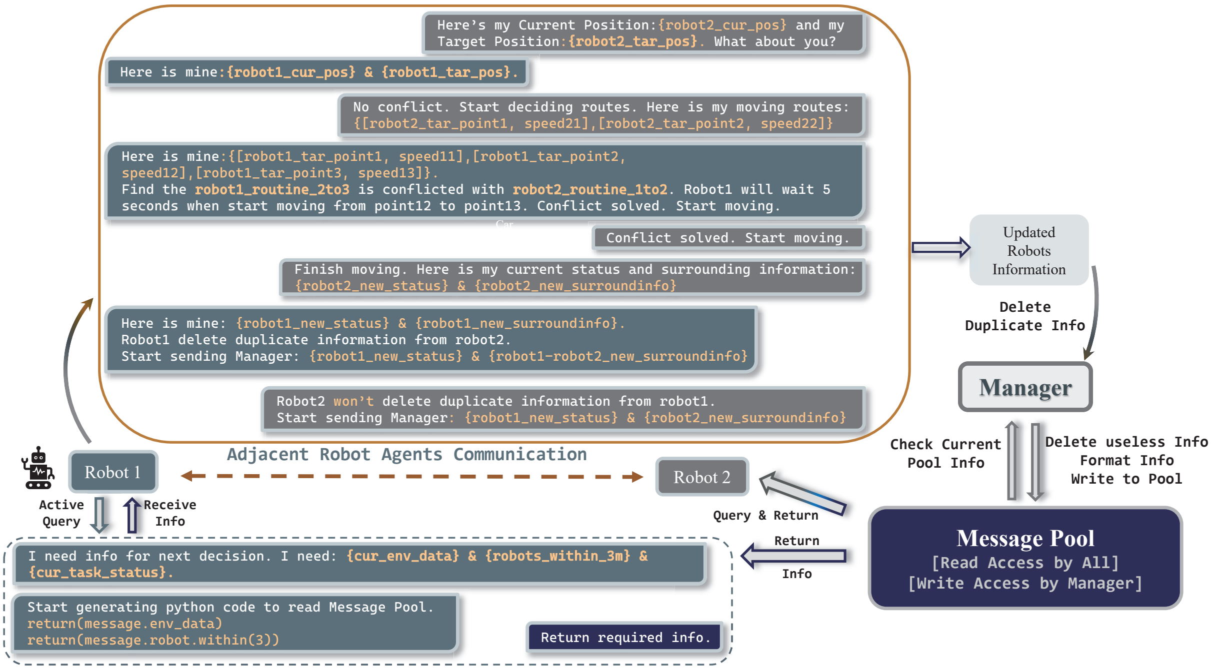

Figure 3: Communication and data management flow in the multi-robot collaboration system. Robots share status and environmental data through direct communication and the message pool. The manager oversees data management, ensuring data integrity and accessibility for effective decision-making.

task section of the sealed message pool for future reference and modifications.

This communication method reduces redundancy among agents and enhances fault tolerance, enabling efficient handling of unexpected situations. Feedback requirements are stringent, with agents needing to negotiate and synchronize submissions for manager approval.

Furthermore, we have implemented broadcast communication. If a robot discovers an unrecorded obstacle, such as a new obstacle, it proactively informs nearby robots and has the limited reporting authority to notify the manager first. This ensures the system can promptly address unexpected situations while effectively avoiding redundant reporting of environmental information, thereby reducing the manager's workload.

Meanwhile, both robots and the manager can communicate with the message pool, as previously mentioned. Robot agents have read-only access. By actively writing and running Python code and using the provided message pool API, they can retrieve the information they need. To prevent unauthorized writes by robot agents, we strictly prohibit them from executing related operations despite providing the API. Additionally, the manager periodically checks for discrepancies between the current message pool and its previous backup versions to ensure the security and reliability of the message pool.

Although the manager has write access, executing write operations is not straightforward. In most cases, besides routine writing tasks, when it comes to writing or modifying sealed information such as task data, the manager must ensure there are sufficient reasons and adequate logical information to proceed with the task. This requirement further enhances the stability of message pool communication. The specific operations can be referred to in Figure3.

3.4 Decision-Making and Task Execution

In task planning and decision-making, we adopted RoCo's multi-RRT approach. Leveraging the reliability of the message pool, this decision-making method can be executed more rapidly. The manager performs initial task allocation and aims to avoid changes in subsequent tasks unless special circumstances arise, as mentioned previously. Using the basic information of robots, environmental details, obstacles, and task data obtained during the initialization of the message pool, the manager formulates an initial plan outlining the tasks for each robot. This plan is then distributed to the robots.

During the initial rounds of actions, robots provide real-time feedback to validate the plan's feasibility. In most cases, due to the comprehensive information and the ability of robots to communicate and avoid mutual interference, there are rarely any rejections or errors. Once the tasks are clarified, robots determine their actions for each round and use the RRT algorithm and inverse kinematics (IK) for trajectory planning. Subsequently, they execute the relevant actions, completing the task planning and execution process.

Thanks to the completeness and reliable information provided by the message pool, the stability of task decision-making and execution is significantly enhanced. This framework not only demonstrates superior performance in overall robot coordination but also exhibits considerable robustness and adaptability to special circumstances.

4 Experiment

To validate the effectiveness and robustness of our proposed multi-robot collaboration framework, we designed a series of experiments. These experiments aim to evaluate the performance of the framework in various scenarios, including task completion efficiency, communication effectiveness, decision-making accuracy, and overall system scalability. The experiments were conducted in the MuJoCo (Multi-Joint dynamics with Contact) simulation environment, incorporating our complete communication system based on the ROCO framework.

4.1 Experiment Setup

We designed an experimental framework that examines multiple dimensions at both the system and individual levels. We conducted targeted experiments focusing on three key aspects: the stability of the message pool, the accuracy and reliability of robot communication, and the stability of robot execution.

Our experimental setup was based on the MuJoCo environment, utilizing GPT-3.5-turbo and GPT-4 for testing. We employed three task modes: low-load tasks (box unwrapping, requiring two robots), medium-load tasks (table cleaning, requiring two robots, and drink preparation, requiring three robots), and high-load tasks (sorting 30 cubes into four groups by color and stacking them sequentially, requiring five robots). These tests were conducted in the MuJoCo environment, leveraging its API to obtain environmental information. Summary of setup information and Table1.

Table 1: Experimental Task Setup

Task Name	Load Level	Robot Number
Unboxing	Low	2
Table Clean	Medium	2
Drink Prepare	Medium	3
Cube Sorting	High	5

To conduct the tests, we introduced our complete communication system into the environment as the test subject. We also incorporated the ROCO framework and a simple central Planprompt-based agent collaboration system, which allows agents to communicate with each other. The comparison of performance across three scenarios demonstrates the advantages and primary application areas of our communication system.

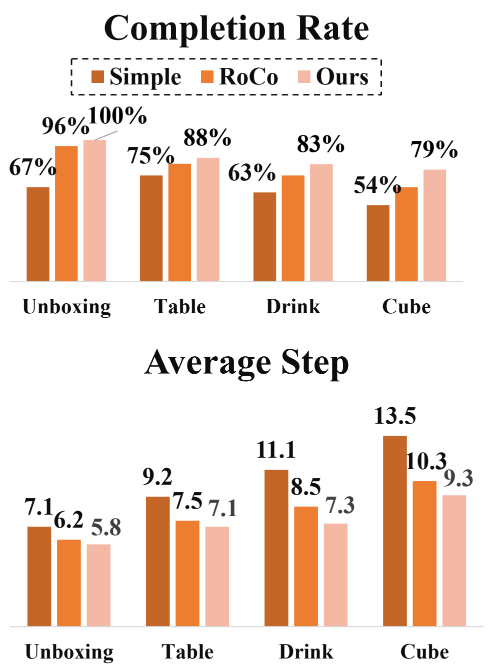

Figure 4: This figure shows the performance of our three communication systems across four environments during 24 rounds of load testing. The tests revealed that our proactive intelligent communication system using a message pool had a completion rate approximately 30% higher than the centralized plan system. Additionally, it had an advantage in average execution rounds, requiring 1-2 fewer rounds on average compared to the other two systems.

4.2 Intergroup Communication

We first tested the performance of three communication systems across four tasks. Each system

Table 2: Ablation Study Results

Configuration	Average Step	Effective information rate(%)	Error Rate (%)
Full System	7.1	94.7	12
Without Message Pool	9.7	78.4	26
Without Communication	9.5	83.2	24
Without Task Allocation	11.4	63.2	22

used GPT-3.5-turbo as the base parameter for each round. Each system was tested 24 times per task, and the completion rate and number of completed rounds were recorded to evaluate the effectiveness of intra-group communication.

As shown in Figure 4, our experiments indicate that our proactive communication system based on the message pool outperforms in both task completion rate and average steps to complete the task. In simpler scenarios, such as unboxing and table cleaning tasks, the message pool-based system demonstrates a time advantage over RoCo. In more complex scenarios, such as drink preparation and cube sorting tasks, our system significantly surpasses the central simple system in both completion rate and time savings and also shows improvements compared to RoCo. This indicates that for large-scale multi-robot cooperation, the message pool can fully utilize each agent's intelligence and leverage its stable information storage capacity for stronger performance.

4.3 Message Pool

Next, we explored the stability of the message pool. We simulated high-frequency communication between eight robot agents to test the response speed and accuracy of the message pool in the most complex cube task, with a particular focus on whether the manager could effectively perform task analysis and management. Due to the high load difficulty, we conducted parallel tests using both GPT-3.5-turbo and GPT-4.

Our separate tests showed that the message pool with GPT-4 had higher stability. Even when faced with multi-agent, multi-range loads, the manager could effectively manage relevant information without encountering issues such as information disorder. This indicates a certain level of stability in the scenario. However, when using GPT-3.5-turbo, some problems still occurred. In 30 rounds of testing, there were about 6 rounds where information overlap in the message pool was observed, with an overlap rate not exceeding 2.2%. This suggests that lower-intelligence managers might face information conflict issues, especially in highly concurrent scenarios. Therefore, enhancing the intelligence of the manager might be crucial.

We also believe that this is due to the manager's insufficient intelligence and the large number of related prompts (compared to ordinary agents, there are more content understanding tasks for writing). This is one of the reasons why managers with lower intelligence make mistakes.

4.4 Ablation Study

To further analyze the importance of each component, we conducted an ablation study. By systematically removing different modules—message pool, communication protocol optimizations, and task allocation algorithm—we assessed their impact on overall performance.

Using GPT-3.5-turbo in the table cleaning task, we conducted 30 rounds of testing for each configuration. This task, with its low difficulty level, offers significant general applicability. Specific data can be found in Table 2.

Analysis of the data reveals that each part of our system plays a crucial role. Communication between robots enhances the effective utilization of information, while the message pool reduces agent load and lowers the error rate. Task allocation significantly decreases the number of communications needed for task distribution, saving a substantial amount of time.

5 Conclusion

We proposed a novel multi-robot collaboration system that leverages LLMs for enhanced communication, planning, and execution. By integrating a centralized message pool and LLM-assisted decision-making, our approach outperforms existing multi-agent systems. Future work will improve information synchronization, robustness, and multi-system collaboration, enhancing efficiency in complex environments.

References

Michael Ahn, Anthony Brohan, Noah Brown, Kanishka Rao Burns, Yevgen Chebotar, Aakanksha Chowdhery, Hao-Tien Lewis Chu, Adam Coates, Andrew Dai, Chelsea Finn, et al. 2022a. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2204.01691*.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022b. Do as i can, not as i say: Grounding language in robotic affordances. *Preprint*, arXiv:2204.01691.

Haoxiang Bai, Qiu Gu, Chun-Yu Lin, Wei Liu, and Lei Song. 2023. Cogloop: A cognitive architecture for continual learning and task reasoning. *arXiv preprint arXiv:2302.05787*.

Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. *Preprint*, arXiv:2306.06531.

Gabriel GR de Castro and Luiz Chaimowicz. 2023. Coverage path planning for multi-robot systems in partially known dynamic environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5678–5685. IEEE.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Pierre-Louis Guhur, Aymeric Voisin, Thomas Wolf, and Julien Mairal. 2023. Instruction augmentation for vision-language navigation. *arXiv preprint arXiv:2301.08427*.

Xiaoli Guo, Lai Jiang, Xingyuan Liu, Jiale Hong, Hai Zhao, and Min Zhang. 2023. Task allocation and coordinated motion planning for multi-robot systems. *arXiv preprint arXiv:2305.12456*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Yang Hu, Yu Li, Haoyu Wang, Zhen Liu, and Jun Sun. 2023. Collision-free multi-robot collaborative manipulation using llm-based motion planning. *arXiv preprint arXiv:2307.04838*.

Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. 2023. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *Preprint*, arXiv:2305.11176.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. *Preprint*, arXiv:2207.05608.

Junghwan Jang, Hyung Jin Jeon, Jaewook Choi, and Dongheui Kim. 2023. Matcha: A middleware for adaptive task coordination and handoff in human-robot collaboration. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12345–12352. IEEE.

Hanna Kwon, Minje Kang, Sanghyun Park, Yonghwa Suh, and Bohyung Kim. 2023. Robotic imitation learning with vision-language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4567–4576. IEEE.

Junghwan Li, Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role-play with large language models. *arXiv preprint arXiv:2305.10142*.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. Roco: Dialectic multi-robot collaboration with large language models. *Preprint*, arXiv:2307.04738.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models. *Preprint*, arXiv:2307.06435.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. Progprompt: Generating situated robot task plans using large language models. *Preprint*, arXiv:2209.11302.

Huaxiaoyue Wang, Gonzalo Gonzalez-Pumariega, Yash Sharma, and Sanjiban Choudhury. 2023a. Demo2code: From summarizing demonstrations to synthesizing code via extended chain-of-thought. *Preprint*, arXiv:2305.16744.

Jiaqi Wang, Zihao Wu, Yiwei Li, Hanqi Jiang, Peng Shu, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Huaqin Zhao, Zhengliang Liu, Haixing Dai, Lin Zhao, Bao Ge, Xiang Li, Tianming Liu, and Shu Zhang. 2024. Large language models for robotics: Opportunities, challenges, and perspectives. *Preprint*, arXiv:2401.04334.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, et al. 2023b. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2301.08427*.

Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. Tidybot: personalized robot assistance with large language models. *Autonomous Robots*, 47(8):1087–1102.

Andy Zeng, Michael Laskin, Kevin Lee, Yilun Lu, Jackson Lee, Ted Xiao, Allen Guo, Alexander Herzog, Karol Hausman, Julian Ibarz, et al. 2022a. Multi-robot task and motion planning: a survey. *arXiv preprint arXiv:2212.02429*.

Andy Zeng, Michael Laskin, Kevin Lee, Yilun Lu, Jackson Lee, Ted Xiao, Allen Guo, Alexander Herzog, Karol Hausman, Julian Ibarz, et al. 2022b. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. 2023. Large language models for robotics: A survey. *Preprint*, arXiv:2311.07226.

Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. 2023. Large language models for human–robot interaction: A review. *Biomimetic Intelligence and Robotics*, 3(4):100131.