2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

RoCo: Dialectic Multi-Robot Collaboration with Large Language Models

Zhao Mandi[1,2], Shreeya Jain[1], Shuran Song[1,2]

***Abstract*— We propose a novel approach to multi-robot collaboration that harnesses the power of pre-trained large language models (LLMs) for both high-level communication and low-level path planning. Robots are equipped with LLMs to discuss and collectively reason task strategies. They generate sub-task plans and task space waypoint paths, which are used by a multi-arm motion planner to accelerate trajectory planning. We also provide feedback from the environment, such as collision checking, and prompt the LLM agents to improve their plan and waypoints in-context. For evaluation, we introduce RoCoBench, a 6-task benchmark covering a wide range of multi-robot collaboration scenarios, accompanied by a text-only dataset that evaluates LLMs' agent representation and reasoning capability. We experimentally demonstrate the effectiveness of our approach — it achieves high success rates across all tasks in RoCoBench and adapts to variations in task semantics. Our dialog setup offers high interpretability and flexibility — in real world experiments, we show RoCo easily incorporates human-in-the-loop, where a user can communicate and collaborate with a robot agent to complete tasks together.**

I. INTRODUCTION

Multi-robot systems are intriguing for their promise of enhancing task productivity, but are faced with various challenges. For robots to effectively split and allocate the work, it requires high-level understanding of a task, and consideration of each robot's capabilities such as reach range or payload. Another challenge lies in low-level motion planning: as the configuration space grows with the number of robots, finding collision-free motion plans becomes exponentially difficult. Finally, traditional multi-robot systems typically require task-specific engineering, hence compromise generalization: with much of the task structures pre-defined, these systems are incapable of adapting to new scenarios or variations in a task. In this work, we propose **RoCo**, a zero-shot multi-robot collaboration method to address the above challenges. Our approach includes three key components:

- **Dialogue-style task-coordination**: To facilitate information exchange and task reasoning, we let robots 'talk' among themselves by delegating each robot to an LLM agent in a dialog, which allows robots to discuss the task in natural language, with high interpretability for supervision.
- **Feedback-improved Sub-task Plan Generated by LLMs**: The multi-agent dialog ends with a sub-task plan for each agent (e.g. pick up object). We provide a set of environment validations and feedback (e.g. IK failures or collision) to the LLM agents until a valid plan is proposed.
- **LLM-informed Motion-Planning in Joint Space**: From the validated sub-task plan, we extract goal configurations in the robots' joint space, and use a centralized RRT-sampler to plan motion trajectories. We explore a less-studied capability of LLM: 3D spatial reasoning. Given the start, goal, and obstacle locations in task space, we show LLMs can generate waypoint paths that incorporate high-level task semantics and environmental constraints, and significantly reduce the motion planner's sample complexity.

We next introduce RoCoBench, a benchmark with 6 multi-robot manipulation tasks, which we use to experimentally demonstrate the effectiveness of RoCo: by leveraging the common-sense knowledge captured by large language models (LLMs), RoCo is flexible in handling a variety of collaboration scenarios without any task-specific training.

In summary, we propose a novel approach to multi-robot collaboration, supported by two technical contributions: 1) An LLM-based multi-robot framework (**RoCo**) that is flexible in handling a wide variety of tasks with improved task-level coordination and action-level motion planning; 2) A new benchmark (**RoCoBench**) for multi-robot manipulation to systematically evaluate these capabilities. It includes a suite of tasks that are designed to examine the flexibility and generality of the algorithm in handling different task semantics (e.g., sequential or concurrent), different levels of workspace overlaps, and varying agent capabilities (e.g., reach range and end-effector types) and embodiment (e.g., 6DoF UR5, 7DoF Franka, 20DoF Humanoid).

II. PRELIMINARIES

Task Assumption. We consider a cooperative multi-agent task environment with N robot agents, a finite time horizon T, full observation space O. Each agent with index n has observation space $\Omega^n \subset O$. Agents may have asymmetric observation spaces and capabilities, which stresses the need for communication. We manually define description functions f that translate task semantics and observations at a time-step t into natural language prompts: $l_t^n = f^n(g^n, o_t), o_t \in \Omega^n$. We also define parsing functions that map LLM outputs (e.g. text string "PICK object") to the corresponding sub-task, which can be described by one or more gripper goal configurations.

Multi-arm Path Planning. Let $\mathcal{X} \in \mathbb{R}^d$ denote the joint configuration space of all N robot arms and $\mathcal{X}_{ob}$ be the obstacle region in the configuration space, then collision-free space $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{ob}$. Given an initial condition $x_{init} \in \mathcal{X}_{free}$, a goal region $x_{goal} \in \mathcal{X}_{free}$, the problem of multi-arm motion planning deals with finding an optimal

[1]Columbia University in the City of New York
[2]Stanford University
Project Webpage: https://project-roco.github.io
Correspondence to: mandi@stanford.edu

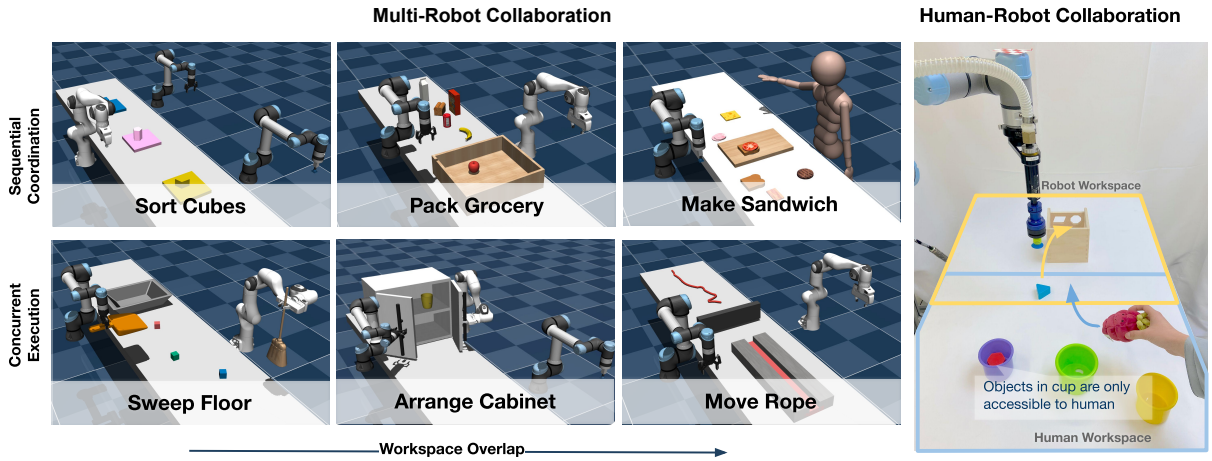

Fig. 1: We propose RoCo, a unified approach for multi-robot collaboration that leverages LLMs for both high-level task coordination and low-level motion planning. Left: we demonstrate its utility on RoCoBench, a benchmark we introduce that includes a diverse set of challenges in collaboration task scenarios. Right: in our real-world experiments, we show RoCo can be adapted for human-robot collaboration, where a human and a robot each has limited access to the workspace and must coordinate to succeed the task

$\sigma^* : [0, 1] \to \mathcal{X}$ that satisfies: $\sigma^*(0) = x_{init}$, $\sigma^*(1) \in x_{goal}$. We assume the resulting σ^* is used by the robot arms' position-based joint controllers to execute in open-loop.

III. Multi-Robot Collaboration with LLMs

We introduce RoCo, a novel method for multi-robot collaboration that leverages LLMs for robot communication and motion planning. The three key components in our method are illustrated in Fig. 2 and described below:

A. Multi-Agent Dialog via LLMs

We assume multi-agent task environments with asymmetric observation space and skill capabilities, which necessitates communication in order to coordinate meaningfully. We leverage pre-trained LLMs to facilitate this communication: specifically, before each environment interaction, we set up one round of dialog where each robot is delegated to an LLM agent, which receives agent-specific information and responds strictly according to its role. For each agent's LLM prompt, we use a shared overall structure with content that varies with each robot's individual status. The prompt is composed of 6 key components, as described below. See Appendix XI for further details and a concrete prompt example.

B. LLM-Generated Sub-task Plan

Once a round of dialog ends, the last speaking agent summarizes a 'sub-task plan', where each agent gets one sub-task (e.g. pick up an object) and optionally a path of 3D waypoints in the task space. This sub-task plan is first passed through a set of validations before going into execution. If any of the checks fail, the feedback is appended to each agent's prompt and another round of dialog begins. The validations are conducted in the order described below, and terminates at the first failed check (e.g. a plan must pass the parsing check before checking for task and agent constraints):

1) **Text Parsing** ensures the plan follows desired format and contains all required keywords
2) **Task Constraints** checks whether each action complies with the task and agent constraints
3) **IK** checks whether each robot arm's target pose is feasible via inverse kinematics
4) **Collision Checking** checks if the IK-solved arm join configurations cause collision
5) **Valid Waypoints** optionally, if a task requires path planning, each intermediate waypoint must be IK-solvable and collision-free, and all steps should be evenly spaced

The agents are allowed to re-plan until reaching a maximum number of attempts, after which the current round ends without any execution and the next round begins. The episode is considered failed if the task is not completed within a finite number of rounds.

C. LLM-informed Motion Planning in Joint Space

Once a sub-task plan passes all validations, we use Inverse Kinematics (IK) to produce the goal configuration jointly over all robot arms, and optionally, each step of the task space waypoint paths produces an intermediate goal configuration. The goal configuration(s) are passed to an RRT-based multi-arm motion planner that jointly plans over all robot arms, and outputs motion trajectories for each robot, which gets executed in the environment. Note that, in contrast to our assumption on decentralized high-level task planning, we use centralized collision-checking and planning for the multi-arm motion planner, because geometry-level information of an environment can be more easily gathered and planned over.

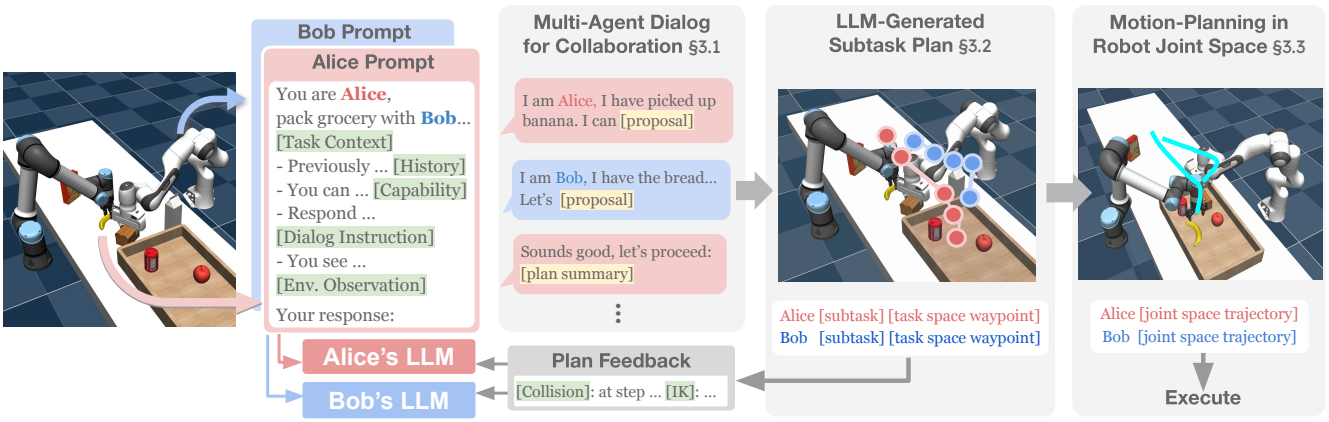

Fig. 2: RoCo consists of three main components: 1) **Multi-agent dialog via LLMs**: each robot is equipped with an LLM that 'talks' on its behalf, enabling a discussion of task strategy. 2) **LLM-Generated Sub-task Plan**: the dialog ends with a proposal of sub-task plan, including optionally a path of task space waypoints, and environment feedback on invalid plans are provided for the agents to improve. 3) **Multi-arm motion planning**: A validated sub-task plan then produces goal configurations for the robot arms, which are used by a centralized multi-arm motion planner that outputs trajectories for each robot.

	Sweep Floor	Pack Grocery	Move Rope	Arrange Cabinet	Make Sandwich	Sort Cubes
Task	Parallel	Parallel	Parallel	Seq	Seq	Seq
Observation	Asym.	Shared	Shared	Asym.	Asym.	Shared
Workspace	Med	Med	High	High	Low	Low

TABLE I: Overview of the key properties designed in RoCoBench tasks. Task Decomposition (1st row): whether sub-tasks can be completed in parallel or sequentially; Observation space (2nd row): whether all agents receive the same information of task status; Workspace Overlap (3rd row): proximity between robots during execution.

IV. RoCoBench : A Benchmark for Multi-Robot Collaboration

RoCoBench is a suite of 6 multi-robot collaboration tasks in a tabletop manipulation setting. The tasks involve common household objects that LLMs easily understand, and span a repertoire of collaboration scenarios that require different robot communication and coordination behaviors. See Appendix X for detailed documentation on the benchmark. Below we describe three key properties that define each task (also see Table I for a summary):

1) Task decomposition: whether a task can be decomposed into sub-parts that can be completed in parallel or in certain order. Three tasks in RoCoBench have a sequential nature (e.g. Make Sandwich requires food items to be stacked in correct order), while the other three tasks can be executed in parallel (e.g. objects in Pack Grocery can be put into bin in any order).
2) Observation space: how much of the task and environment information each robot agent receives. Three tasks provide shared observation of the task workspace, while the other three have a more asymmetric setup and robots must inquire each other to exchange knowledge.
3) Workspace overlap: proximity between operating robots; we rank each task from low, medium or high, where higher overlap calls for more careful low-level coordination (e.g. Move Rope requires manipulating the same object together).

V. Experiments

Overview. We design a series of experiments using RoCoBench to validate our approach. In Section V-A, we evaluate the task performance of RoCo compared to an oracle LLM-planner that receives oracle task information, and ablate on different components of the dialog prompting in RoCo. Section V-B shows empirically the benefits of LLM-proposed 3D waypoints in multi-arm motion planning. Section V-C contains qualitative results that demonstrate the flexibility and adaptability of RoCo. Additional experiment results, such as failure analysis, are provided in Appendix XII.

A. Main Results on RoCoBench

Experiment Setup. GPT-4 [32] was used for all our results. We empirically compare our method (denoted as 'Dialog') to an oracle LLM-planner, i.e. 'Central Plan', which receives full environment observation and knowledge about all robots' capabilities and plans for all robots at once. We deem this planner 'oracle' because it does not need to address the information asymmetry like the individual dialog agents in our method.

We also evaluate two ablations on the prompt components of RoCo: the first removes dialog and action history from past rounds, i.e. 'Dialog w/o History'. Second removes environment feedback, i.e. 'Dialog w/o Feedback', where a failed action plan is discarded and agents are prompted to continue discussion without detailed failure reasons. To offset the lack of re-plan rounds, each episode is given twice the budget of episode length.

		Pack Grocery	Arrange Cabinet	Sweep Floor	Make Sandwich	Sort Cubes	Move Rope
Central Plan (oracle)	Success step, replan	0.82 ± 0.06 11.1, 3.9	0.90 ± 0.07 4.0, 2.7	1.00 ± 0.00 8.4, 2.0	0.96 ± 0.04 8.8, 1.2	0.70 ± 0.10 8.6, 2.6	0.50 ± 0.11 2.3, 3.9
Dialog w/o History	Success step, replan	0.48 ± 0.11 9.2, 3.1	1.00 ± 0.00 4.2, 1.4	0.00 ± 0.00 N/A, 1.0	0.33 ± 0.12 9.6, 1.8	0.73 ± 0.11 5.8, 1.4	0.65 ± 0.11 3.7, 3.1
Dialog w/o Feedback	Success step, replan	0.35 ± 0.10 18.0, 1.0	0.70 ± 0.10 5.9, 1.0	0.95 ± 0.05 7.6, 1.0	0.35 ± 0.11 12.6, 1.0	0.53 ± 0.13 4.9, 1.0	0.45 ± 0.11 3.4, 1.0
Dialog (ours)	Success step, replan	0.44 ± 0.06 9.9, 3.5	0.75 ± 0.10 4.7, 2.0	0.95 ± 0.05 7.1, 1.0	0.80 ± 0.08 10.2, 1.7	0.93 ± 0.06 4.9, 1.3	0.65 ± 0.11 2.5, 3.1

TABLE II: Evaluation results on RoCoBench. We report averaged success rates ↑ over 20 runs per task, the average number of steps in *successful* runs ↓ , and average number of re-plan attempts used across all runs ↓ .

Evaluation Metric. Provided with a finite number of rounds per episode and a maximum number of re-plan attempts per round, the task performance is evaluated with three metrics: 1) task success rate within the finite rounds; 2) number of environment steps the agents took to succeed an episode, which measures the efficiency of the robots' strategy; 3) average number of re-plan attempts at each round before an environment action is executed – this reflects the agents' ability to understand and use environment feedback to improve their plans. Overall, a method is considered better if the task success rate is higher, it takes fewer environment steps, and requires fewer number of re-plans.

Result Analysis. The evaluation results are reported in Table II. We make the following remarks:

1) Comparison to centralized planner: With the oracle global information, the centralized planner out-performs dialog agents on 4 out of 6 tasks; however, results on the remaining 2 tasks suggest the shorter, agent-specific dialog prompts are better at representing the agents' individual interests and result in a more effective task strategy than when the centralized planner trying to satisfy all agents' constraints at once.

2) Takeaway from ablation results: When history information or plan feedback rounds are removed, the overall performance drops (with the exception of some individual tasks). In contrary to the assumption that LLMs might fail to efficiently process prompts with long context length, the LLM-powered dialog agents are capable of reasoning over the entire input prompt and utilize it to plan actions.

3) Centralized planner is better at utilizing waypoint feedback: We observe that, on Pack Grocery, the oracle planner shows better capability at collision-free waypoint planning. This suggests, on numerical 3D path outputs, centralized processing is better for allowing LLMs to incorporate feedback and improve on individual coordinate steps. A valuable line of future work lies in improving the spatial reasoning ability of LLM-powered agents under the decentralized dialog setting.

B. Effect of LLM-proposed 3D Waypoints

We demonstrate the utility of LLM-proposed task space waypoints using two tasks that were designed to have high workspace overlap, i.e. Pack Grocery and Move Rope, which require both picking and placing to complete the task. For comparison, we define a hard-coded waypoint path that performs top-down pick or place, i.e. always moving the gripper over a certain height atop an object before picking it up, and lifting an object up to a height above the table before moving and placing. We single-out one-step pick or place snapshots, and run multi-arm motion planning using the compared waypoints under a maximum of 300 second planning time budget. As shown in Fig. 4, LLM-proposed waypoints show no clear benefits for picking sub-tasks, but significantly accelerate planning for placing, where collisions are more likely to happen between the arms and the desktop objects.

		Object Init.	Task Order
Human Correction	Success	9/10	8/10
	Step	5.3	5.5
Imperfect Human	Success	7/10	6/10
	Step	5.6	5.2

TABLE III: Real world experiment results. We report number of successes and average number of steps in sucessful runs.

C. Zero-shot Adaptation to Task Variations

Leveraging the zero- and few-shot ability of LLMs, RoCo demonstrates strong adaptation ability to varying task semantics, which traditionally would require modification or re-programming of a system, e.g. fine-tuning a learning-based policy. We use Make Sandwich task in RoCoBench and showcase 3 main variation categories (see Figure 3 for an overview):

1. Object Initialization: Initial locations of the food items are randomized, and we show the dialog agents' reasoning is robust to this variation.

2. Task Goal: Agents must stack food items in the correct order given by the sandwich recipe, and coordinate sub-task strategies accordingly.

3. Robot Capability: Agents are able to exchange information on items that are within their respective reach in order to coordinate their action plans.

D. Real-world Experiments: Human-Robot Collaboration

Task Setup: A human user and a UR5E arm stand on opposite sides of the table. The overall task goal is to move

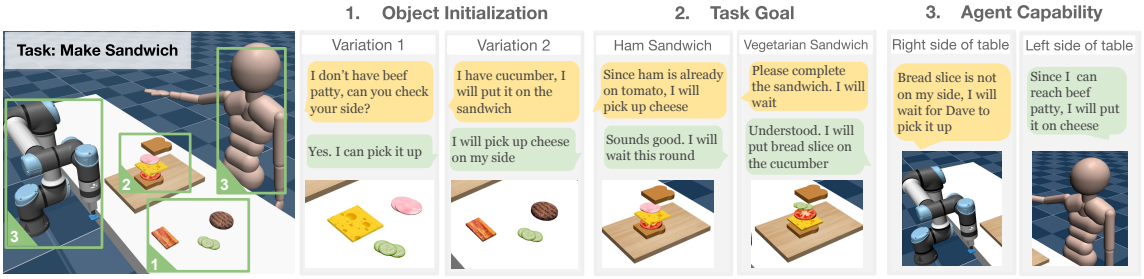

Fig. 3: RoCo demonstrates strong adaptation ability to variations in task semantics. We use Make Sandwich task in RoCoBench to showcase three variation categories: 1) object initialization, i.e. randomized food items' locations on the table; 2) task goals, i.e. robots must change behavior according to different sandwich recipes; 3) agent capability, e.g. agents can only pick food items that are within their reach range.

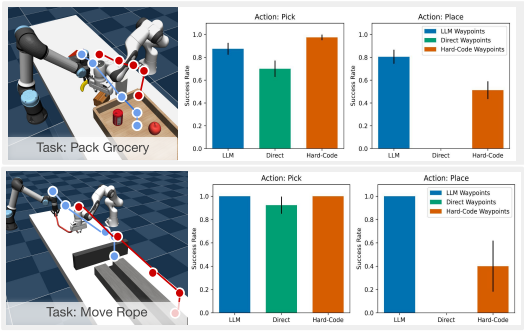

Fig. 4: We demonstrate the utility of LLM-proposed waypoints via comparison with two alternatives: a linear waypoint path that interpolates between start and goal; 'Hard-code', a pre-defined waypoint path that always performs top-down pick or place.

cubes into the wooden bin. The human is only capable of picking up cubes from inside the plastic cups and placing them on the table, while the robot can pick up cubes from the table and place them into the wooden bin.

Experiment Setup: We run RoCo with the modification that only the robot agent is controlled by GPT-4, and engages in the dialog with the human user. For perception, we use a pre-trained object detection model, OWL-ViT [30], to generate scene description from top-down RGB-D camera images. We evaluate 2 main task variation categories: 1) object initialization, i.e. initial block locations are randomized for each run (Fig. 5.1); 2) task order specification, where the agents are asked to follow a fixed order to move the blocks (Fig. 5.2). We also evaluate two types of human behaviors: 1) an oracle human that corrects mistakes in the OWL-ViT-guided scene descriptions and the robot's responses; 2) an imperfect human that provides no feedback to those errors.

Evaluation Results We evaluate a total of 40 runs, divided into 10 runs for each variation. In Table III, we report task success rate within the finite rounds, and number of steps the agents took to succeed an episode. We remark that task performance is primarily bottle-necked by incorrect object detection from OWL-ViT, which leads to either an incorrect object being picked up and resulting in failure or no object being picked up and resulting in higher steps. See Appendix XII-C for further details on the real

VI. Related Work

LLMs for Robotics.

An initial line of prior work uses LLMs to select skill primitives and complete robotic tasks, such as SayCan [1], Inner Monologue [15], which, similarly to ours, uses environment feedback to in-context improve planning. Later work leverages the code-generation abilities of LLMs to generate robot policies in code format, such as CaP [24], ProgGPT [37] and Demo2Code [40]; or longer programs for robot execution such as TidyBot [41] and Instruct2Act [14]. Related to our use of motion-planner, prior work such as Text2Motion [26], AutoTAMP [4] and LLM-GROP [8], [47] studies combining LLMs with traditional Task and Motion Planning (TAMP). Other work explores using LLMs to facilitate human-robot collaboration [6], to design rewards for reinforcement learning (RL) [21], and for real-time motion-planning control in robotic tasks [45]. While prior work uses single-robot setups and single-thread LLM planning, we consider multi-robot settings that can achieve more complex tasks, and use dialog prompting for task reasoning and coordination.

Multi-modal Pre-training for Robotics. LLMs' lack of perception ability bottlenecks its combination with robotic applications. One solution is to pre-train new models with both vision, language and large-scale robot data: the multi-modal pre-trained PALM-E [11] achieves both perception and task planning with a single model; Interactive Language [29] and DIAL [43] builds a large dataset of language-annotated robot trajectories for training generalizable imitation policies. Another solution is to introduce other pre-trained models, mainly vision-language models (VLMs) such as CLIP [34]). In works such as Socratic Models [46], Matcha [48], and [20], LLMs are used to repeatedly query and synthesize information from other models to improve reasoning about the environment. While most use zero-shot LLMs and VLMs, works such as CogLoop [17] also explores fine-tuning adaptation layers to better bridge different frozen models. Our work takes advantage of simulation to extract perceptual information, and our real world experiments fol-

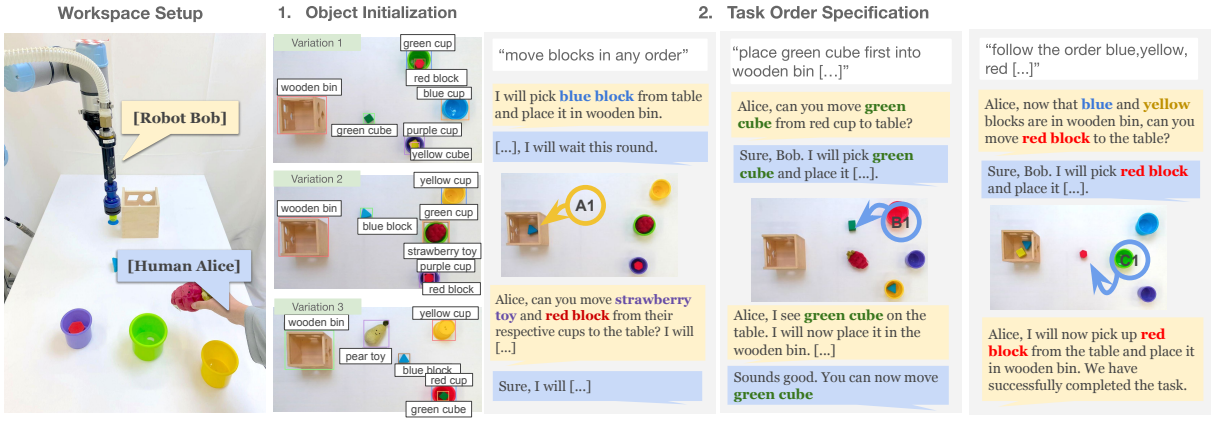

Fig. 5: Real world experiments: collaborative block sorting between a robot and a human, with varying task semantics. We test two variation categories: 1) object initialization, i.e. the object locations are randomized for each episode 2) task order specification, i.e. agents must follow the specified order to move blocks.

low prior work [23], [37], [41] in using pre-trained object detection models [30] to generate scene description.

Dialogue, Debate, and Role-play LLMs. Outside of robotics, LLMs have been shown to possess the capability of representing agentic intentions [2] and behaviors, which enables multi-agent interactions in simulated environments such as text-based games [36], [3] and social sandbox [33], [22], [27]. Recent work also shows a dialog or debate style prompting can improve LLMs' performance on human alignment [16] and a broad range of goal-oriented tasks [31], [25], [12]. While prior work focuses more on understanding LLM behaviors or improve solution to a single question, our setup requires planning separate actions for each agent, thus adding to the complexity of discussion and the difficulty in achieving consensus.

Multi-Robot Collaboration and Motion Planning. Research on multi-robot manipulation has a long line of history [19]. A first cluster of work studies the low-level problem of finding collision-free motion trajectories. Sampling-based methods are a popular approach [18], where various algorithmic improvements have been proposed [9]. Recent work also explored learning-based methods [13] as alternative. While our tasks are mainly set in static scenes, much work has also studied more challenging scenarios that require more complex low-level control, such as involving dynamic objects [35] or closed-chain kinematics [44], [42]. A second cluster of work focuses more on high-level planning to allocate and coordinate sub-tasks, which our work is more relevant to. While most prior work tailor their systems to a small set of tasks, such as furniture assembly [10], we highlight the generality of our approach to the variety of tasks it enables in few-shot fashion.

VII. RoCoBench-Text: Multi-Agent Representation and Reasoning Dataset

In addition to our main experimental results, we curate a text-based dataset, RoCoBench-Text, to evaluate an LLM's agent representation and task reasoning ability. This dataset aligns LLM with desirable capabilities in multi-agent collaboration, without requiring robotic environment interaction. It builds on data from our evaluation on RoCoBench, and contains a series of additional questions that are more open-ended and go beyond simply finding the next best action plan. We provide detailed descriptions on the construction of this text-based dataset, and report experimental results for evaluating popular large language models on our dataset. Due to space limitation, we yield the majority of this section to Appendix XIII

VIII. Conclusion

We present RoCo, a new framework for multi-robot collaboration that leverages large language models (LLMs) for robot coordination and planning. We introduce RoCoBench, a 6-task benchmark for multi-robot manipulation to be open-sourced to the broader research community. We empirically demonstrate the generality of our approach and many desirable properties such as few-shot adaptation to varying task semantics, while identifying limitations and room for improvement. Our work falls in line with recent literature that explores harnessing the power of LLMs for robotic applications, and points to many exciting opportunities for future research in this direction.

IX. Appendix

Please refer to our project website for the full appendix: https://project-roco.github.io/roco-icra-appendix.pdf

Acknowledgment

This work was supported in part by NSF Award #2143601, #2037101, and #2132519. We would like to thank Google for the UR5 robot hardware. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

REFERENCES

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can, not as i say: Grounding language in robotic affordances," 2022.

[2] J. Andreas, "Language models as agent models," *arXiv preprint arXiv:2212.01681*, 2022.

[3] K. Chalamalasetti, J. Götze, S. Hakimov, B. Madureira, P. Sadler, and D. Schlangen, "clembench: Using game play to evaluate chat-optimized language models as conversational agents," 2023.

[4] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," *arXiv preprint arXiv:2306.06531*, 2023.

[5] M. M. Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/deepmind/mujoco_menagerie

[6] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh, "" no, to the right"–online language corrections for robotic manipulation via shared autonomy," *arXiv preprint arXiv:2301.02555*, 2023.

[7] S. Dasari, A. Gupta, and V. Kumar, "Learning dexterous manipulation from exemplar object trajectories and pre-grasps," in *IEEE International Conference on Robotics and Automation 2023*, 2023.

[8] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," *arXiv preprint arXiv:2303.06247*, 2023.

[9] A. Dobson and K. E. Bekris, "Planning representations and algorithms for prehensile multi-arm manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6381–6386.

[10] M. Dogar, A. Spielberg, S. Baker, and D. Rus, "Multi-robot grasp planning for sequential assembly operations," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 193–200.

[11] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[12] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," 2023.

[13] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," in *Conference on Robotic Learning (CoRL)*, 2020.

[14] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li, "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," *arXiv preprint arXiv:2305.11176*, 2023.

[15] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. R. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*, 2022.

[16] G. Irving, P. Christiano, and D. Amodei, "Ai safety via debate," 2018.

[17] C. Jin, W. Tan, J. Yang, B. Liu, R. Song, L. Wang, and J. Fu, "Alphablock: Embodied finetuning for vision-language reasoning in robot manipulation," 2023.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," 2011.

[19] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 945–952 vol.2, 1994.

[20] M. Kwon, H. Hu, V. Myers, S. Karamcheti, A. Dragan, and D. Sadigh, "Toward grounded social reasoning," *arXiv preprint arXiv:2306.08651*, 2023.

[21] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," *arXiv preprint arXiv:2303.00001*, 2023.

[22] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for "mind" exploration of large scale language model society," *ArXiv*, vol. abs/2303.17760, 2023.

[23] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022.

[24] ——, "Code as policis: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[25] T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, Z. Tu, and S. Shi, "Encouraging divergent thinking in large language models through multi-agent debate," *ArXiv*, vol. abs/2305.19118, 2023.

[26] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[27] R. Liu, R. Yang, C. Jia, G. Zhang, D. Zhou, A. M. Dai, D. Yang, and S. Vosoughi, "Training socially aligned language models in simulated human society," 2023.

[28] A. LLC. (2023) Introducing claude. [Online]. Available: https://www.anthropic.com/index/introducing-claude

[29] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *arXiv preprint arXiv:2210.06407*, 2022.

[30] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby, "Simple open-vocabulary object detection with vision transformers," 2022.

[31] V. Nair, E. Schumacher, G. Tso, and A. Kannan, "Dera: Enhancing large language model completions with dialog-enabled resolving agents," *arXiv preprint arXiv:2303.17071*, 2023.

[32] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023.

[33] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," 2023.

[34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[35] S. S. M. Salehian, N. Figueroa, and A. Billard, "Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty." in *Robotics: Science and Systems*, 2016.

[36] D. Schlangen, "Dialogue games for benchmarking language understanding: Motivation, taxonomy, strategy," 2023.

[37] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," *arXiv preprint arXiv:2209.11302*, 2022.

[38] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[39] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020.

[40] H. Wang, G. Gonzalez-Pumariega, Y. Sharma, and S. Choudhury, "Demo2code: From summarizing demonstrations to synthesizing code via extended chain-of-thought," *arXiv preprint arXiv:2305.16744*, 2023.

[41] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023.

[42] Z. Xian, P. Lertkultanon, and Q.-C. Pham, "Closed-chain manipulation of large objects by multi-arm robotic systems," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1832–1839, 2017.

[43] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson, "Robotic skill acquisition via instruction augmentation with vision-language models," *arXiv preprint arXiv:2211.11736*, 2022.

[44] J. Yakey, S. LaValle, and L. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 951 – 958, 01 2002.

[45] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, *et al.*, "Language to rewards for robotic skill synthesis," *arXiv preprint arXiv:2306.08647*, 2023.

[46] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence, "Socratic models: Composing zero-shot multimodal reasoning with language," 2022.

[47] X. Zhang, Y. Zhu, Y. Ding, Y. Zhu, P. Stone, and S. Zhang, "Visually grounded task and motion planning for mobile manipulation," in *2022*

International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 1925–1931.

[48] X. Zhao, M. Li, C. Weber, M. B. Hafez, and S. Wermter, "Chat with the environment: Interactive multimodal perception using large language models," *arXiv preprint arXiv:2303.08268*, 2023.

APPENDIX

X. RoCoBench

A. Overview

RoCoBench is built with MuJoCo [38] physics engine. The authors would like to thank the various related open-source efforts that greatly assisted the development of RoCoBench tasks: DMControl [39], Menagerie [5], and MuJoCo object assets from [7]. The sections below provide a detailed documentation for each of the 6 simulated collaboration tasks.

B. Task: Sweep Floor

Task Description. 2 Robots bring a dustpan and a broom to opposite sides of each cube to sweep it up, then the robot holding dustpan dumps cubes into a trash bin.

Agent Capability. Two robots stand on opposite sides of the table:

1) UR5E with robotiq gripper ('Alice'): holds a dustpan
2) Franka Panda ('Bob'): holds a broom

Observation Space. 1) cube locations: a. on table; b. inside dustpan; c. inside trash bin; 2) robot status: 3D gripper locations

Available Robot Skills. 1) MOVE [target]: target can only be a cube; 2) SWEEP [target]: moves the groom so it pushes the target into dustpan; 3) WAIT; 4) DUMP: dump dustpan over the top of trash bin.

C. Task: Make Sandwich

Task Description. 2 Robots make a sandwich together, each having access to a different set of ingredients. They must select the required items and take turns to stack them in the correct order.

Agent Capability. Two robots stand on opposite sides of the table:

Observation Space 1) the robot's own gripper state (either empty or holding an object); 2) food items on the robot's own side of the table and on the cutting board.

Available Robot Skills. 1) PICK [object]; 2) PUT [object] on [target]; WAIT

D. Task: Sort Cubes

Task Description. 3 Robots sort 3 cubes onto their corresponding panels. The robots must stay within their respective reach range, and help each other to move a cube closer.

Agent Capability. Three robots each responsible for one area on the table

1) UR5E with robotiq gripper ('Alice'): must put blue square on panel2, can only reach: panel1, panel2, panel3.
2) Franka Panda ('Bob'): must put pink polygon on panel4, can only reach: panel3, panel4, panel5.
3) UR5E with suction gripper ('Chad'): must put yellow trapezoid on panel6, can only reach: panel5, panel6, panel7.

Observation Space 1) the robot's own goal, 2) locations of each cube.

Available Robot Skills. 1) PICK [object] PLACE [panelX]; 2) WAIT

E. Task: Pack Grocery

Task Description. 2 Robots pack a set of grocery items from the table into a bin. The objects are in close proximity and robots must coordinate their paths to avoid collision.

Agent Capability. Two robots on opposite sides of table

1) UR5E with robotiq gripper ('Alice'): can pick and place any object on the table
2) Franka Panda ('Bob'): can pick and place any object on the table

Observation Space 1) robots' gripper locations, 2) locations of each object, 3) locations of all slots in the bin.

Available Robot Skills. (must include task-space waypoints) 1) PICK [object] PATH [path]; 2) PLACE [object] [target] PATH [path]

F. Task: Move Rope

Task Description. 2 robots lift a rope together over a wall and place it into a groove. They must coordinate their grippers to avoid collision.

Agent Capability. Two robots on opposite sides of table

1) UR5E with robotiq gripper ('Alice'):: can pick and place any end of the rope within its reach
2) Franka Panda ('Bob'): can pick and place any end of the rope within its reach

Observation Space 1) robots' gripper locations, 2) locations of rope's front and end back ends; 3) locations of corners of the obstacle wall; 4) locations of left and right ends of the groove slot.

Available Robot Skills. (must include task-space waypoints) 1) PICK [object] PATH [path]; 2) PLACE [object] [target] PATH [path]

G. Task: Arrange Cabinet

Task Description. 3 robots, two of them each hold one side of the cabinet door open, while the third robot takes the cups out and place them onto the correct coasters.

Agent Capability. Three robots, one on left side of the table, two on right side of table

1) UR5E with robotiq gripper ('Alice'): stands on left side, can only reach left cabinet door
2) Franka Panda ('Bob'): : stands on right side, can only reach right cabinet door
3) UR5E with suction gripper ('Chad'):: stands on right side, can reach right cabinet door and cups and mugs inside the cabinet.

Observation Space 1) locations cabinet door handles; 2) each robot's reachable objects, unaware of other robot's reach range.

Available Robot Skills. 1) PICK [object]; 2) OPEN [one side of door handle]; 3) WAIT; 3) PLACE [object] [target]

XI. Details on LLM Prompting

We describe our proposed method of multi-agent dialog in Algorithm 1: during each call to **PromptDialogs**, each agent speaks at least once before reaching an acton plan; and after each call to **GiveFeedback**, the proposed plan is passed

through a set of validation check and optionally results in a text feedback that's used in the next round of dialog; the finalized plan is used by **MotionPlanner** to produce robot motion trajectories for execution in the environment.

To produce each agent response in a dialog, we use a separate query call to the LLM with an agent-specific prompt. The prompt provides information regarding the agent's capability, the overall task objective, past history, plan feedback, and environment observation. As a concrete example, we provide in the text box below the LLM prompt for one agent at the second time-step during one evaluation run of the Sort Cube task (some texts are omitted for readability)

Algorithm 1 Multi-agent dialog for collaboration

Require: agent $u^1, ..., u^N$, task horizon T;
Require: max number of re-plans K, max number of dialog per round M,
Require: history buffer H; feedback buffer F
  $t \leftarrow 0$
  $o\_t \leftarrow$ env.reset()
  H.empty()
  while $t < T$ **do**
    F.empty()
    while len(F)$< K$ **do**
      dialog, plan $\leftarrow$ PromptDialogs(H, F, o_t, u^n)
      plan-valid, feedback $\leftarrow$ GiveFeedback(plan)
      if plan-valid **then**
        final-plan $\leftarrow$ parsed-plan
        break
      end if
      F.append(feedback)
    end while
    if plan-valid **then**
      $\sigma\_t \leftarrow$ MotionPlanner(o_t, final-plan)
      $o\_t + 1, r\_t + 1 \leftarrow$ env.step(σ_t)
      if $r\_t + 1 > 0$ **then**
        break
      end if
    end if
    H.append(dialog)
    $t \leftarrow t + 1$
  end while

1. Agent Capability
[Action Options]
1) PICK [object name] PLACE [location] 2) WAIT
Only PICK an object if your gripper is empty. Target [location] for PLACE should be panel or a bin.
[Action Output Instruction]
You must first output 'EXECUTE ', then give **exactly** one action per robot [omitted: rest of format instruction]
2. Round History
[History]
== Round#0 ==
[Chat History]
[Alice]: [...]
[Bob]: Hello Alice and Chad, I am Bob. [omitted: rest of dialog history]
[Executed Action]
Alice: PICK pink_polygon PLACE panel3
Bob: PICK yellow_trapezoid PLACE panel5
Chad: PICK blue_square PLACE panel7
== Current Round ==
3. Task Context
7 panels on the table, ordered left to right: panel1,...,panel7. They form a straight assembly line, panel1 is closed to panel2 and farthest from panel7. You are robot Alice in front of panel2. You are collaborating with Bob, Chad to sort cubes into their target panels. The task is NOT done until all three cubes are sorted.
4. Current Observation
At current round:
blue_square is on panel7
pink_polygon is on panel3
yellow_trapezoid is on panel5
Your goal is to place blue_square on panel2, but you can only reach panel1, panel2, panel3 [omitted: rest of observation description]
5. Communication Instruction
When you respond, tell others about your goal and all constraints.[omitted: rest of communication instruction]
[Previous Chat]
[Alice]: [omitted: rest of previous dialog in current round]
6. Plan Feedback
[Environment Feedback]: - Previous Plan:
NAME Alice ACTION WAIT
NAME Bob ACTION PICK pink_polygon PLACE panel4
NAME Chad ACTION PICK blue_square PLACE panel3
- Goal Step: Reachability failed: Out of reach: Chad

You are Alice, your response is:
[...] seems that Chad cannot reach the blue_square on panel7.[...]

XII. Additional Experiment Results

A. Failure Analysis on RoCoBench

We provide example failure scenarios observed in the agent dialog and environment interaction.

Factual Errors in Dialog. We observe that when one LLM-generated agent makes a mistake, it could sometimes cause subsequent dialog to build on the wrong reasoning. See an example below: one agent (Alice) wrongfully decided the task is complete, and other agents repeat this wrong claim and choose to wait for multiple rounds while

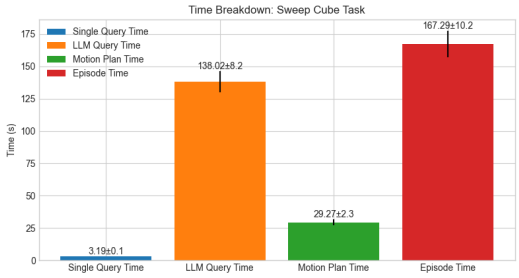

Fig. 6: Timing breakdown during evaluation runs on RoCoBench. We run 10 episodes of Sweep Cubes task and up to 10 environment steps. 8 out of 10 episodes ended successfully within the given time budget

the task is, in fact, not finished.

[Alice]: [...] we have successfully completed our task. [...], let's wait for further instructions.
[Bob]: I agree that we have successfully completed our task. Great teamwork, everyone! Let's wait [...]
[Chad]: I concur that we have accomplished our goal. Excellent teamwork! Let's wait [...].

Errors in Motion Trajectory Execution Due to the open-loop nature of the execution, small errors in a motion trajectory could lead to unexpected errors, e.g. knocking of an object by accident.

B. Timing Breakdown

In this section, we aim to provide a better sense of time cost requirement for running RoCo on an example task (i.e. Sweep Cubes). We ran 10 additional episodes with maximum 5 re-plans and 10 environment steps per episode, which results in 8 out of 10 successful episodes. During each evaluation run, we record the timing per each GPT-4 querying, and the motion planning time for each sub-task plan (including inverse kinematics calculation, collision-checking time, and joint multi-arm RRT sampling time). The average time across all 10 episodes is reported in Figure 6. We remark the LLM-querying bottlenecks the timing much more so than motion planning time, which is mainly a result of requiring multiple agent exchanges before an action plan is proposed. More specifically, the evaluations average 7.1 steps to either succeed or reach time-out of an episode, whereas on average each episode queries GPT-4 43.8 times in total. However, this time cost can potentially be significantly reduced via better accessibility to the OpenAI API services.

C. Real World Experiment Setup

The robot agent is a 6DoF UR5E arm with suction gripper, and dialog is enabled by querying a GPT-4 model to respond as agent 'Bob', who is discussing with a human collaborator 'Alice'. The human user provides text input to engage in the dialog, and arranges cubes on the same tabletop. For perception, we use top-down RGB-D image from an Azure Kinect sensor. See the text below for an example of the robot's prompt:

==== System Prompt ====
[Action Options]
1) PICK <obj> PLACE <target>: robot Bob must decide which block to PICK and where to PLACE. To complete the task, Bob must PLACE all blocks in the wooden bin.
2) WAIT: robot Bob can choose to do nothing, and wait for human Alice to move blocks from inside cups to the table.

[Action Output Instruction]
First output 'EXECUTE
', then give exactly one ACTION for the robot.
Example#1: 'EXECUTE
NAME Bob ACTION PICK green_cube PLACE wooden_bin
' Example#2: 'EXECUTE
NAME Bob ACTION WAIT
'

You are a robot called Bob, and you are collaborating with human Alice to move blocks from inside cups to a wooden bin.
You cannot pick blocks when they are inside cups, but can pick blocks when they are on the table. Alice must help you by moving blocks from inside cups to the table.
You must WAIT for Alice to move blocks from inside cups to the table, then you can PICK blocks from the table and PLACE them in the wooden bin.
[mention task order specification]
Talk with Alice to coordinate and decide what to do.

At the current round:
[object descriptions from observation]
Think step-by-step about the task and Alice's response.
Improve your plans if given [Environment Feedback].
Propose exactly one action for yourself at the current round, select from [Action Options].
End your response by either: 1) output PROCEED, if the plans require further discussion; 2) If everyone has made proposals and got approved, output the final plan, must strictly follow [Action Output Instruction]!

==== User Prompt ====
You are Bob, your response is:
response from GPT-4:
EXECUTE
NAME Bob ACTION ...

XIII. Multi-Agent Representation and Reasoning Dataset

A. Dataset Overview

This dataset contains yes/no, multiple-choice or short question-answering questions, spanning a range of different reasoning abilities:
Self-knowledge evaluates how well the agent establishes its identity under a given task context, divided into two categories: 1) understanding an agent's own capability (e.g. which objects/area are not reachable); 2) memory retrieval,

i.e. inferring information from past dialog and actions.

Communication Skills evaluates an agent's ability to effectively exchange information and drive a discussion into an agreeable plan. The questions ask an LLM to 1) choose appropriate response to other agents' questions; 2) choose appropriate inquiries to other agents.

Adaptation evaluates adaptation to unexpected situations that were not specified in context. We use a subset of RoCoBench tasks to design unexpected occurrences, either regarding task state (e.g. a missing object) or a response from another agent, and ask an LLM agent to choose the best response. See below for an example question: two agents make a sandwich together; one agent is informed of a broken gripper and must infer that the sandwich can actually be completed without any item from its side of the table.

B. Example Questions

1) Self-Knowledge Question-Answering:

- **Agent Capability.** This category contains 57 questions, based on Sort Cubes task from RoCoBench. By asking an LLM to explain an agent's own capability under the given task constraints, these questions evaluate how well the LLM represents and establishes the identity of individual agents.

  - Context (system prompt):
  7 panels on the table, ordered left to right: panel1,...,panel7. They form a straight assembly line, panel1 is closed to panel2 and farthest from panel7.
  You are robot Alice in front of panel2. You are collaborating with Bob, Chad to sort cubes into their target panels. The task is NOT done until all three cubes are sorted.
  At current round:
  blue_square is on panel5
  pink_polygon is on panel1
  yellow_trapezoid is on panel3
  Your goal is to place blue_square on panel2, but you can only reach panel1, panel2, panel3: this means you can only pick cubes from these panels, and can only place cubes on these panels.
  Never forget you are Alice! Never forget you can only reach panel1, panel2, panel3!
  - Question (user prompt):
  You are Alice. List all panels that are out of your reach. Think step-by-step. Answer with a list of panel numbers, e.g. [1, 2] means you can't reach panel 1 and 2.
  - Solution:
  panels [4,5,6,7]

- **Memory Retrieval.** This category contains 44 total questions, based on Make Sandwich and Sweep Floor tasks from RoCoBench. By providing a history of agent dialog and environment actions and asking an LLM to reason about an agent's past, the questions evaluates how well the LLM performs memory retrieval and reasoning for individual agents.

  - Context (system prompt):
  [History]
  Round#0:
  [Chat History] [Chad]: ... [Dave]:... [Chad]: ... ... [Executed Action]...
  Round#1:
  .....
  - Current Round
  You are a robot Chad, collaborating with Dave to make a vegetarian_sandwich [.....] You can see these food items are on your reachable side: ...
  - Question (user prompt) You are Chad. Based on your [Chat History] with Dave and [Executed Action] from previous rounds in [History], what food items were initially on Dave's side of the table? Only list items that Dave explicitly told you about and Dave actually picked up. Don't list items that you are unsure about. Output the item names as a list. Think step-by-step.
  - Solution:
  bread_slice1

2) Effective Communication:

- **Inquiry.** This category contains 41 multiple-choice questions, based on Arrange Cabinet task from RoCoBench. The questions ask an LLM to speak as an agent and choose the most appropriate inquiry to seek information that helps their task reasoning.

  - Context (system prompt):
  You are Bob, collaborating with Alice, Chad to pick a mug and a cup out of cabinet, and place them on correct coasters. Both left and right cabinet doors should be OPENed and held open, while anything inside can be PICKed. You must coordinate to complete the task.
  At current round: left door is closed, right door is closed, mug is inside cabinet; cup is inside cabinet;
  Alice's gripper is holding nothing,
  Your gripper is holding nothing,
  Chad's gripper is holding nothing,
  Never forget you are Bob! Never forget you can only reach right door handle!
  - Question (user prompt):
  You are thinking about picking right door handle. Who and what should you ask to confirm this action? Think step-by-step, then choose exactly one option from below.
  [A] tell others about this plan because you are free and right door handle is within your reach.
  [B] ask if Alice and Chad can reach right door handle because it's not within your reach.
  [C] ask if Alice and Chad can help, because you can reach right door handle, but you are busy and they are free.
  [D] all three of you are busy, so it's better to wait until later.
  - Solution: [A]

- **Responsiveness.** This category contains 96 yes/no questions, based on Sort Cubes task from RoCoBench. The questions ask an LLM to speak for one agent and choose the most appropriate response to other agents under a given task context.

7 panels on the table, ordered left to right: panel1,...,panel7. They form a straight assembly line, panel1 is closed to panel2 and farthest from panel7.
You are robot Alice in front of panel2. You are collaborating with Bob, Chad to sort cubes into their target panels. The task is NOT done until all three cubes are sorted. At current round:
blue_square is on panel5 pink_polygon is on panel3 yellow_trapezoid is on panel3
Your goal is to place blue_square on panel2, but you can only reach panel1, panel2, panel3: this means you can only pick cubes from these panels, and can only place cubes on these panels. You can help others by moving other cubes that are within your reachable panels. Never forget you are Alice! Never forget you can only reach panel1, panel2, panel3!

You are Alice. Chad needs to put yellow_trapezoid on panel6, and asked you for help. How should you respond? Think step-by-step. If you can help, your answer must start with YES and propose a plan to help. If you can't help, must answer NO.

3) Adaptation to Unexpected Scenarios:

- This category contains 31 questions in the format of multiple choice (A, B, C), based on 3 RoCoBench tasks: Arrange Cabinet, Make Sandwich, Sweep Floor. The questions hypothesize unexpected situations that were not specified in the task context, and evaluate an LLM-represented agent's ability to choose the most appropriate response.

You are a robot Chad, collaborating with Dave to make a [vegetarian_sandwich]. Food items must be stacked following this order: bread_slice1, tomato, cheese, cucumber, bread_slice2, where bread_slice1 must be PUT on cutting_board. You must stay on right side of the table! This means you can only PICK food from right side, and Dave can only PICK from the other side. Only one robot can PUT at a time, so you must coordinate with Dave. At the current round: You can see these food items are on your reachable side: bread_slice1: on cutting_board cheese: atop tomato tomato: atop bread_slice1 cucumber: atop cheese ham: on your side beef_patty: on your side Your gripper is empty
You are Chad. Your gripper is not working right now. What should you say to Dave? Select exactly one option from below. You must first output a single option number (e.g. A), then give a very short, one-line reason for why you choose it.
Options:
A: Sorry Dave, we can't complete the task anymore, my gripper is broke.
B: Let's stop. The recipe needs ham but Dave can't reach my side and my gripper is not functioning.
C: Dave, go ahead and finish the sandwich without me, there isn't anything we need on my side anyway.

C. Evaluation Results

Setup. All questions are designed to have only one correct answer, hence we measure the average accuracy in each category. We evaluate GPT-4 (OpenAI), GPT-3.5-turbo (OpenAI), and Claude-v1 (Anthropic[28]). For GPT-4, we use two models marked with different time-stamps, i.e. 03/14/2023 and 06/13/2023. Results are summarized in Table IV: we observe that, with small performance variations between the two versions, GPT-4 leads the performance across all categories. We remark that there is still a considerable gap from fully accurate, and hope this dataset will be useful for improving and evaluating language models in future work.

Qualitative Results. We observe GPT-4 is better at following the instruction to formulate output, whereas GPT-3.5-turbo is more prone to confident and elongated wrong answers. See below for an example response from an agent capability question (the prompt is redacted for readability).

You are robot Chad .. [cube-on-panel locations...]. You can reach: [panels] Which cube(s) can you reach? [...] Answer with a list of cube names, answer None if you can't reach any.
Solution: None
GPT-4: None
Claude-v1: yellow_trapezoid
GPT-3.5-turbo:
At the current round, I can reach the yellow_trapezoid cube on panel3.

	Self-knowledge		Communication		Adaptation
	Capability	Memory	Inquiry	Respond	
GPT-4-0314	0.67 ± 0.06	0.84 ± 0.06	**0.79 ± 0.06**	0.83 ± 0.04	0.68 ± 0.08
GPT-4-0613	**0.68 ± 0.06**	**0.91 ± 0.04**	0.57 ± 0.08	**0.86 ± 0.03**	**0.71 ± 0.08**
GPT-3.5-turbo	**0.68 ± 0.06**	0.59 ± 0.07	0.48 ± 0.08	0.30 ± 0.05	0.58 ± 0.09
Claude-v1	0.37 ± 0.06	0.70 ± 0.07	0.55 ± 0.08	0.60 ± 0.05	0.65 ± 0.09

TABLE IV: Evaluation results on the multi-agent LLM reasoning dataset. We measure the question-answering accuracy on each test category and compare performance of four different models.