CogRob Report

Your Name

Abstract—Large Language Models (LLMs) are increasingly used as decision-making components in embodied AI, yet their integration into robotic systems remains underexplored. This project investigates how an LLM can serve as the cognitive core of a simulated mobile manipulator operating in a household environment. The agent is embodied in PyBullet based simulation as a mobile manipulator with an omnidirectional base and a 7-DoF robotic arm. It can interact with its environment through a set of high-level tool functions for perception, reasoning, navigation, grasping, and placement. Two household tasks, object placement and shelf reordering, were designed to evaluate the agent's reasoning, planning, and memory utilization. Results demonstrate that the LLM-driven agent can complete structured tasks and adapt strategies, but also reveal limitations such as task refusal and difficulties in managing complex tasks including non obvious subtasks. These findings highlight both the potential and challenges of employing LLMs as embodied cognitive controllers for autonomous robots.

Index Terms—Cognitive Robotics, Embodied AI, Large Language Models

I. Introduction

Large Language Models (LLMs) have been shown to exhibit impressive abilities in reasoning, language understanding, and task execution. This project explores how an LLM can serve as the core of a cognitive architecture embodied in a simulated 3D robot. By assigning the agent physical tasks in a realistic environment, we aim to investigate how such systems can perceive, plan, act, and adapt in grounded, interactive settings.

A. Motivation

To fully explore the cognitive capabilities of large language models, it is essential to situate them in environments that demand embodied, goal-directed interaction. This project investigates how LLMs function as the core of a cognitive architecture within a simulated 3D robotic setting, where the agent must physically manipulate objects and execute tasks in the real world.

Unlike static text-based benchmarks, a robotic context enables examination of embodied reasoning, memory-guided action, and the capacity for grounded, anticipatory planning. By focusing on a single agent operating in a realistic environment – such as a kitchen – we can probe how LLMs reason about spatial layouts, interpret user instructions, and adapt plans based on feedback.

Importantly, the robot's (and by extension, the LLM agent's) planning and decision-making capabilities take on a more grounded role, reasoning in natural language about the physical consequences of movements and manipulative actions. This opens new directions for exploring embodied cognition and could inform future developments in cognitive robotics, assistive AI, and multi-modal agent systems.

B. Problem Statement

[ChatGPT draft] While LLMs have shown remarkable performance in language-based reasoning and planning, their ability to operate as the cognitive core of embodied agents in interactive environments is not well understood. The central problem addressed in this project is how to enable an LLM-driven agent to perceive, plan, and act within a simulated household setting, where it must manipulate objects, navigate between rooms, and complete multi-step tasks. This requires bridging the gap between high-level natural language reasoning and low-level embodied interaction, while coping with the inherent challenges of memory, grounding, and decision consistency.

C. Proposed Approach

[Modified ChatGPT draft] To address this problem, a cognitive robotic agent was designed that embeds an LLM as its central reasoning engine and situates it in a PyBullet simulation of a household environment. The agent is embodied as a mobile manipulator with a 7-DoF arm mounted on an omnidirectional base, enabling navigation and object manipulation. A set of high-level tool functions were implemented to expose perception, navigation, grasping, and placement as callable actions for the LLM. Object placement and shelf reordering were chosen to test the agent's ability to execute complex high level tasks, utilize episodic memory, and adapt to constraints such as occupied spaces. This work evaluates to wich extent LLMs can act as cognitive controllers for robots and identify their strengths and limitations in performing high level tasks.

II. Related Work

This project draws on multiple domains:

A. Cognitive Architectures

Cognitive architectures such as SOAR, which integrates symbolic reasoning with perceptual and motor systems for cognitive robotics [1], and ACT-R, which offers a hybrid symbolic-subsymbolic framework for modeling human cognition [2].

*Submitted to the Department of Computer Science at Hochschule Bonn-Rhein-Sieg in partial fulfilment of the requirements for the degree of Master of Science in Autonomous Systems

†Supervised by Supervisor 1 (Affiliation) and Supervisor 2 (Affiliation)

‡Submitted in Month 20XX

B. Embodied AI and Reinforcement Learning

Embodied-AI and reinforcement-learning agents have been used within simulation frameworks like OpenAI Gym, DeepMind Lab, and MineDojo. Generative models have been used to power social agents that produce believable simulacra of human behaviour, as exemplified by Park et al. [3]. Complementary work on open-ended embodied agents, such as Voyager, a GPT-4-powered agent that incrementally builds a skill library while autonomously exploring Minecraft [4], demonstrates lifelong learning in complex environments.

C. Vision-Language-Action Policies

Generalist vision-language-action policies like Octo map multimodal observations to diverse robot actions and tasks [5]. Our project extends these directions by introducing an explicit cognitive architecture that equips an embodied robot with hierarchical memory, anticipatory reasoning, and explainable decision-making.

D. LLM-Driven Robot Control

Recent research has also explored LLM-driven control of simulated or real robots capable of translating natural-language instructions into sensorimotor actions. Notable examples include:

- RT-2, a vision-language-action model that transfers web-scale knowledge to real-world robotic control [6]
- PaLM-E, an embodied multimodal language model that fuses a large-scale LLM with visual and state inputs [7]
- Code-as-Policies, which prompts code-generating LLMs to produce executable robot control policies [8]
- Instruct2Act, a framework that converts multimodal instructions into perception-planning-action code, outperforming specialist policies on tabletop manipulation tasks [9]

III. Background

This is an optional section in which you can introduce concepts, terms, or methods that are important for understanding your approach and that would not directly fit in Sec. IV. If you do not need this section, comment out the respective line in report.tex.

IV. Methodology

A. LLM Cognition

[TODO]

B. Simulation

The agent and its environment were simulated using the PyBullet physics engine. As PyBullet is a Python module, it's functionality can be directly integrated in any python script. This allows precise control over the simulation and its interaction with external tool calls. PyBullet also comes with methods to create custom objects as well as a fully implemented robotic arm, the KUKA iiwa model. These methods and the KUKA iiwa robotic arm where used to create the agents embodiment and the environment it can act in.

The agent's body is modeled as a mobile manipulator, consisting of a square omnidirectional base and a 7-DoF robotic arm derived from the KUKA iiwa model. To increase maneuverability, the rotational joint limits of the arms joint connecting the base to the arm were disabled, enabling full 360° reach around the base. A fully articulated gripper was not implemented. Instead, grasping and placing were simplified by attaching objects directly to the end-effector when within a predefined proximity threshold, and detaching them at the desired placement location. Figure 1 depicts the agent.

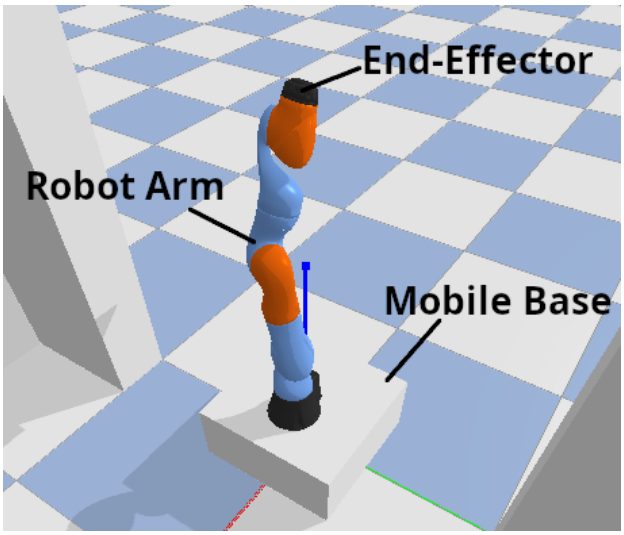

Fig. 1: The simulated mobile manipulator consisting of a square omnidirectional base and a 7-DoF arm.

The environment was designed as a simplified two-room household, comprising a kitchen and a living room. The kitchen contains a three-layer shelf and a table, both capable of supporting objects. The living room contains a television placed on a table. The two rooms are connected by a hallway. Figure 2 shows the environment and the semantic map used for navigation. The green dots are locations the agent can navigate to via the connected lines.

To interact with the environment and its objects, the agent was equipped with several tool functions, each exposed through a tool-calling interface with text-based status responses. The implemented tools are summarized below:

- Look around: Returns the agent's location on a semantic map of the environment, as well as a list of objects detected in proximity of the agent and where these objects are placed.
- Query Memory: The agent queries its episodic memory to gain information from prior executions of the task.

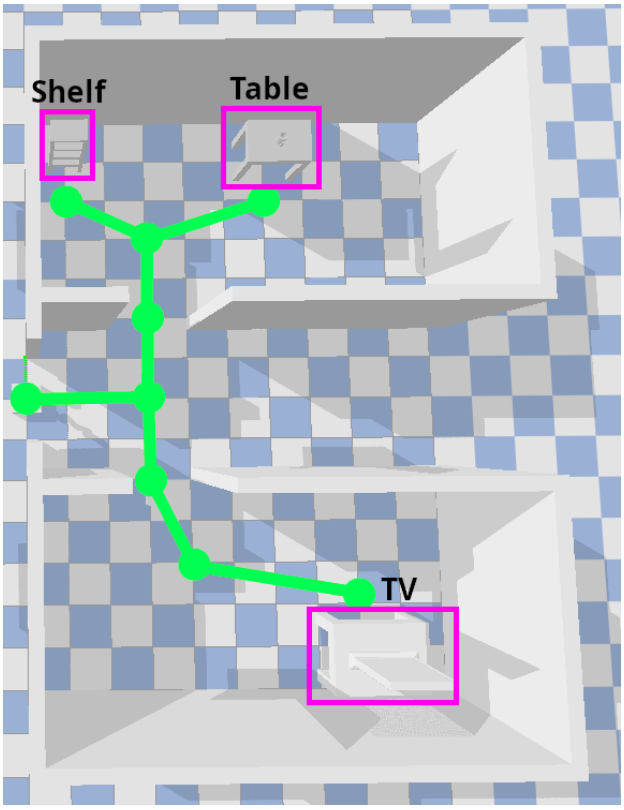

Fig. 2: The simulated household environment and its semantic map representation consisting of a kitchen, a living room, and a connecting hallway.

- Move To: Executes navigation to a goal location on the semantic map. Path planning is performed using the A* algorithm; if a valid path is found, the agent follows it until the goal is reached. If no path is available, the tool reports failure.
- Grab: Moves the robot arm toward a specified target object. If the end-effector reaches within a proximity threshold, the object is attached to the arm, and the tool reports success. Failure is reported if the target object does not exist, is out of reach, or if the agent is already holding an object.
- Place: Allows the agent to release the currently held object at a specified location. Placement succeeds if the end-effector reaches the designated location, the location is unoccupied, and the agent is holding an object. Otherwise, the tool reports failure, specifying the violated condition.
- Scratch Pad: The agent can write in lenght about anything it wants to reason about. This allows the agent to plan its actions beforehand.
- Summarize Task: The agent provides a summary of an atempted task, including if it succeded or failed and what actions it did or attempt to complete the task.

V. Evaluation

To evaluate the agents performance, multiple models have been tested on completing tasks within the environment. The evualted models are [INSERT MODELS HERE]. The tasks on which the agent was evaluated on are described in the following sections.

A. Task 1: Placing all items into the shelf

The first task the agent had to perform within the simulated environment consist of placing all items, here a mug, a box and a cube, into the three-layered shelf. The agent needs to sucessfully navigate to each of the items, grab them, then navigate to the shelf and place them in an unoccupied layer.
The agent perfomed this task successfully without running into larger failures. All models performed in a similar manner.

B. Task 2: Reordering the items in the shelf

After completing the first task, the second task for the agent is to change the order of the items in the now fully occupied shelf. All items need to be placed on a different layer to the one they are currently placed on. To successfully complete this task, the agent needs to figure out that it can not place rearrange the items without storing one item in a location outside of the shelf.
The success of the agent is not as certain as the prior task. Many times the agent gets stuck in trying to place items into occupied locations but it can find a plan involving a temporary storage location, for example the kitchen table, after a while. After successfully utilizing a temporary storage location the agent successfully completes the tasks. The biggest issue with the task does not lie in its execution but in getting the agent to attempt the task in the first place. This is described in more detail in the next section.

C. Intaction Problems with the agent

During prompting the agent with a new task, after it has successfully completed a task, it can happen that the model refuses to attempt the new task. It states that it already has completed the new task, as it assumes the new task is the same as the old task. Sometimes clarifiying that the new task is indeed a new task works, but the agent can remain in this deadlock of refusing any new tasks.

VI. Conclusions

A. Summary

B. Contributions

C. Future Work

Getting the agent to fullfil a new task and not hallucinating about its completion.
Automated test to get quantative results.

References

[1] J. E. Laird, "Cognitive robotics using the soar cognitive architecture," AAAI Workshop - Technical Report, p. 01, 2012.
[2] C. Ritter, J. F. Neves, M. Bagerman, and N. A. Taatgen, "Act-r: A cognitive architecture for modeling cognition," WIREs Cognitive Science, vol. 10, no. 3, 2018.
[3] J. S. Park, E. Kay, J. Zou, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '23. Association for Computing Machinery, 2023.
[4] G. Wang, Y. Xie, L. Dai, T. Zhao, Z. Wang, H. Zhou, H. Chen, R. Li, S. Ma, W. Zhang et al., "Voyager: An open-ended embodied agent with large language models," Transactions on Machine Learning Research, 2024.
[5] O. Mees, Y. Li, Y. He, C. Zhang, L. Tai, and W. Burgard, "Octo: An open-source generalist robot policy," in First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024, 2024. [Online]. Available: https://openreview.net/forum?id=jGrtIvJBpS
[6] A. Brohan, N. Brown, I. Chebotar, O. Cortes, B. Duet, C. Finn, K. Gopalakrishnan, A. Higuera, A. Herzog et al., "Rt-2: Vision-language-action models transfer web knowledge to robotic control," arXiv preprint arXiv:2307.15818, 2023.
[7] D. Driess, F. Xia, M. Wahid, C. Allen-Blanchette, P. Joshi, A. Bengio, B. Faenza, A. Singletary, Y. Wu, A. Gurkaynak et al., "Palm-e: An embodied multimodal language model," arXiv preprint arXiv:2303.03378, 2023.
[8] M. Liang, B. Ichter, T. Zhang, S. Xiao, A. Zhang, F. Xia, and S. Levine, "Code as policies: Language model programs for embodied control," arXiv preprint arXiv:2209.07753, 2022.
[9] Y. Huang, X. Liang, B. Chen, S. Liu, D. Huang, Z. Zhu, H. Ma, J. Zhu, Y. Zhang, H. Yu et al., "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," arXiv preprint arXiv:2305.11176, 2023.

ACKNOWLEDGMENT

Write your acknowledgments here.

STATEMENT OF ORIGINALITY

[If AI assistants have not been used, use this sentence] I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

[If an AI assistant has been used, use this sentence] I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work. The report was, in part, written with the help of the AI assistant [AI assistant name] as described in the appendix. I am aware that content generated by AI systems is no substitute for careful scientific work, which is why all AI-generated content has been critically reviewed by me, and I take full responsibility for it.

Date Signature

Appendix

Please limit the main part of the report to 20 pages (not including the references, the statement of originality, and the appendix).

In your appendix, you can add any additional details about your work, such as:

- extra results that do not necessarily belong in Sec. V
- more detailed justifications of certain algorithm design decisions
- algorithm proofs

Additionally, in the case of using AI assistants, describe in detail what content was generated using an AI assistant. In particular, name the AI assistant(s) that you used and how they were used (e.g. which prompts were used, and for which parts of the project).