

From Language to Action: Can LLM-Based Agents Be Used for Embodied Robot Cognition?

Shinas Shaji^{*,†,§}, Fabian Huppertz^{*,†}, Alex Mitrevski[‡], and Sebastian Houben^{†,§}

Abstract—In order to flexibly act in an everyday environment, a robotic agent needs a variety of cognitive capabilities that enable it to reason about plans and perform execution recovery. Large language models (LLMs) have been shown to demonstrate emergent cognitive aspects, such as reasoning and language understanding; however, the ability to control embodied robotic agents requires reliably bridging high-level language to low-level functionalities for perception and control. In this paper, we investigate the extent to which an LLM can serve as a core component for planning and execution reasoning in a cognitive robot architecture. For this purpose, we propose a cognitive architecture in which an agentic LLM serves as the core component for planning and reasoning, while components for working and episodic memories support learning from experience and adaptation. An instance of the architecture is then used to control a mobile manipulator in a simulated household environment, where environment interaction is done through a set of high-level tools for perception, reasoning, navigation, grasping, and placement, all of which are made available to the LLM-based agent. We evaluate our proposed system on two household tasks (object placement and object swapping), which evaluate the agent’s reasoning, planning, and memory utilisation. The results demonstrate that the LLM-driven agent can complete structured tasks and exhibits emergent adaptation and memory-guided planning, but also reveal significant limitations, such as hallucinations about the task success and poor instruction following by refusing to acknowledge and complete sequential tasks. These findings highlight both the potential and challenges of employing LLMs as embodied cognitive controllers for autonomous robots.

I. INTRODUCTION

Robotic systems for everyday environments need various high-level abilities, such as complex task planning, execution monitoring, and failure recovery, so that they can flexibly perform a wide variety of tasks. Significant work has been done on systems with such broad capabilities in the context of cognitive agent architectures [1], [2], [3], [4], which equip systems with an ability to plan actions, perceive their environment, act accordingly, reason about the world, learn, and adapt [5]. Traditional cognitive architectures, however, usually require a large number of dedicated components that handle concrete aspects of cognition, or they depend on large

amounts of engineered prior knowledge to work effectively. This has generally limited the widespread adoption of such architectures on most robotics platforms.

In the last few years, large language models (LLMs) and vision-language models (VLMs) have been shown to exhibit notable abilities in reasoning, language understanding, and task execution [6], [7], [8], [9], [10], [11], [12]. 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. However, key challenges arise in bridging high-level natural language reasoning with low-level embodied actions. The agent must interpret abstract task descriptions and translate these into sequences of precise robotic actions such as navigation, grasping, and placement. This translation process is complicated by the inherent ambiguity in natural language instructions, leading to misinterpretations where agents may attempt to manipulate inappropriate objects.

This paper investigates whether LLMs (used in a zero-shot manner) can reasonably function as the core of a cognitive architecture within a simulated 3D robotic setting, where the agent physically manipulates objects and executes tasks in the real world. By assigning the agent physical tasks in a simple household environment, we probe how LLMs reason about spatial layouts, interpret user instructions, and adapt plans based on feedback; through this, we aim to investigate whether such systems can perceive, plan, act, and adapt in grounded, interactive settings. Additionally, we explore how episodic memory can enhance task performance through experience accumulation. Our system is illustrated in Fig. 1.

We concretely propose a cognitive agent architecture that embeds an LLM as its central reasoning engine and situates it in a 3D simulation of a simplified household environment. Here, the robot is embodied as a mobile manipulator with a 7-DOF arm mounted on an omnidirectional base, enabling navigation and object manipulation. The environment features a kitchen and living room layout, in which the robot must carry out tasks such as pick-and-place or sorting objects between designated areas. The agent receives observations from the environment primarily as natural language descriptions, detailing the positions and types of objects and entities. Memory is organised into two main tiers: *working memory* in the form of a prompt for immediate task information, and semantically searchable *episodic logs* for synthesised

This work was supported by the b-it foundation

[†]Institute of AI and Autonomous Systems (A²S), Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany, <shinas.shaji, fabian.huppertz>@smail.inf.h-brs.de, sebastian.houben@h-brs.de

[‡]Division for Systems and Control, Chalmers University of Technology, 41258 Gothenburg, Sweden, alemitr@chalmers.se

[§]Fraunhofer Institute for Intelligent Analysis and Information Systems, Sankt Augustin, Germany, <shinas.shaji, sebastian.houben>@iais.fraunhofer.de

*Corresponding author

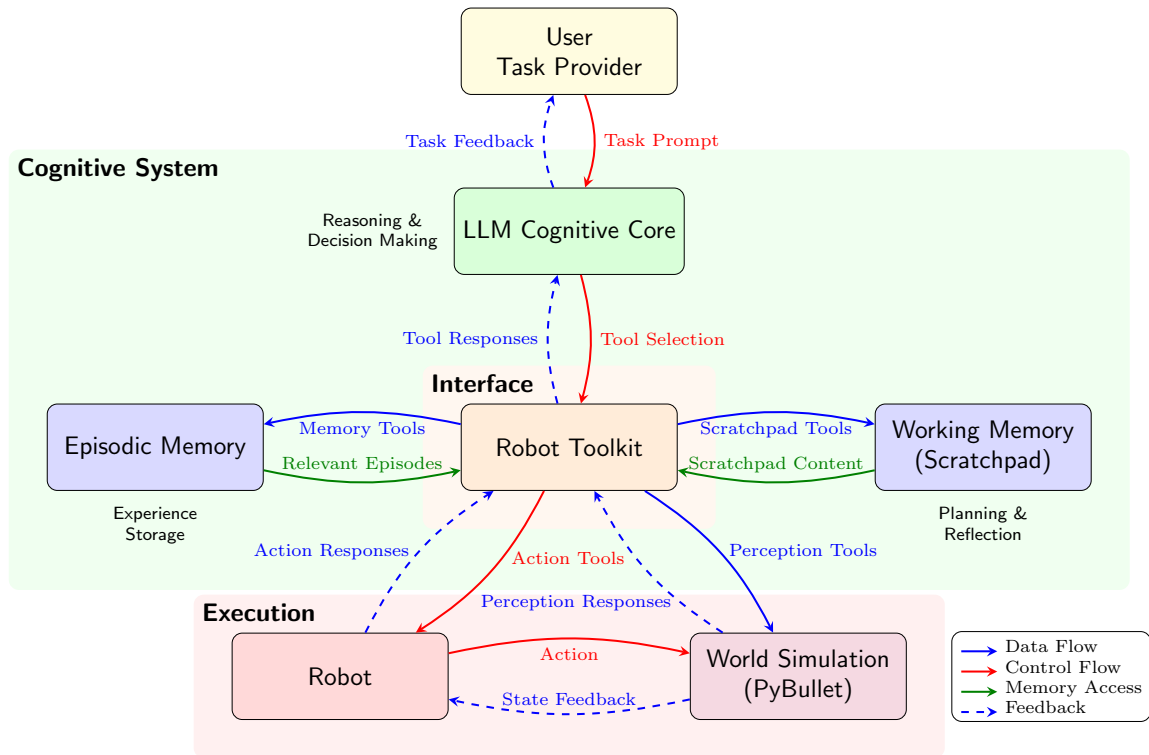


Fig. 1: Overview of our proposed system

summaries of past actions. The LLM is prompted with chain-of-thought prompting [13] to reason and form plans accounting for objects, entities, and anticipated outcomes. A set of high-level tool functions were implemented to expose perception, navigation, grasping, and placement as callable actions for a function-calling LLM [14]. Object placement and shelf reordering were chosen to test the agent’s ability to execute complex high level tasks, utilize its memory system, and adapt to constraints such as occupied spaces.

The contributions of this work include: (i) the *integration of an agentic LLM into a cognitive architecture* with memory and tool interfaces, allowing a robot to interact with its environment through perception, movement, and object manipulation, and (ii) *empirical evaluation of our approach in a simulated 3D environment on two household tasks*, which provides insights into the strengths and limitations of current LLMs when applied to embodied decision-making. The implementation of our method is publicly available.¹

II. RELATED WORK

a) Cognitive architectures: Cognitive architectures provide comprehensive frameworks for developing intelligent systems with human-like cognitive capabilities [1], [5]. Notable architectures include the SOAR architecture [2], which integrates symbolic reasoning with perceptual and motor systems for cognitive robotics; the architecture features mental imagery, episodic and semantic memory, reinforcement learning, and continuous model learning capabilities that

enable robust robotic control in complex environments. Similarly, ACT-R [15] offers a hybrid symbolic-subsymbolic framework for modelling human cognition, with detailed memory models, perceptual-motor integration, and strong empirical grounding in cognitive psychology. A notable robotics-specific architecture is KnowRob 2.0 [4], which combines rich knowledge (in the form of ontologies) with interfaces for logical querying and reasoning, narrative-enabled episodic memories, components for metacognition, and a simulation for physics-based reasoning. Such traditional architectures typically require extensive engineered knowledge and a large number of dedicated components, which makes it challenging to adapt the approaches to diverse robotic platforms. Unlike traditional symbolically grounded approaches, our system leverages LLMs’ natural language capabilities for high-level reasoning and flexible task adaptation, while maintaining the embodied, interactive aspects crucial for robotic cognition.

b) LLM-based embodied AI: Recent advances in LLMs have introduced new possibilities for developing embodied AI systems [6], [7], [8]; these correspond to the emergent cognition paradigm [5]. Concretely, LLM-based agents can, in principle, interpret natural language instructions, reason about spatial relationships, and generate complex plans without extensive domain-specific engineering, leveraging their pre-trained knowledge and reasoning capabilities [16]. However, these approaches often face challenges in grounding abstract reasoning to concrete robotic actions, managing long-term task execution, and maintaining consistency in multi-step operations [17].

¹<https://github.com/ShinasShaji/llm-robot-cognition>

Complementary work in embodied AI has explored reinforcement learning agents within simulation frameworks such as MineDojo [18], demonstrating robust performance in constrained environments. Generative models have shown promise for creating social agents that simulate believable human behaviour through architectures that maintain memory streams and enable reflection [19], while open-ended embodied agents such as Voyager [10] demonstrate lifelong learning capabilities by incrementally building skill libraries in complex environments. However, these approaches often face challenges in grounding abstract task commands to concrete robotic actions and exhibiting agency.

In [20], a comparison of various LLMs is performed for modular planning abilities such as subgoal decomposition and action sequencing, relying on a traditional logical representation of the world and goal. On the other hand, in [21], an LLM is used in conjunction with a pre-defined knowledge base of code snippets, where the LLM is used to generate executable code to perform sub-actions, similar to [6]. Unlike [20], we rely on natural language formulations, using the emergent capabilities of the LLM in understanding tasks and sequencing actions through agentic tool calling. In contrast to [21], we evaluate multiple models, and rely on tool calling (instead of generating executable code) to guarantee consistent behaviors composed from well-defined actions. Our agent also accumulates memory over repeated executions, which is not the case in [20], [21].

c) Robot foundation models: Generalist vision-language and vision-language-action policies, such as Octo [22], π_0 [23], RT-X [12], and OpenVLA [24] map multimodal observations to diverse robot actions and tasks. These are reactive execution models, which do not have the capability to reason and deliberate, and hence are on the level of lower-level skill models. However, they could eventually be integrated into the execution layer of our architecture; this is an interesting direction that could be explored in future work.

d) Agentic LLMs: Recent research on LLM agency distinguishes between *AI agents* (task-specific automation systems) and *agentic AI* (marked by dynamic task decomposition, persistent memory and even multi-agent collaboration) [25], [26]. The concept of *agency*, or the capacity to proactively shape events through one’s actions, has emerged as a critical aspect of LLM behaviour, and represents a progression from isolated reactive agents to systems capable of adaptive, system-level intelligence [27].

Our work bridges aspects of traditional cognitive architectures such as episodic and working memories, with agentic planning and reasoning. We particularly investigate whether agentic LLMs can serve as the core of a cognitive architecture for embodied robots, similar to how traditional cognitive architectures provide integrated memory structures and reasoning capabilities.

III. METHODOLOGY

In this paper, our goal is to investigate whether an LLM-based system is able to meaningfully perform high-level

cognitive processes that a robot needs to execute a variety of tasks in everyday environments. To achieve this, we investigate an agent architecture that enables a robot to (i) *generate task plans* as sequences of actions to satisfy natural language task instructions, (ii) *preserve episodic memories* to simplify the planning process in later executions, (iii) perform *online execution recovery* and replanning in case of detected failures. The processing of information in our system is completely done by an LLM, which receives (or requests) all details about the environment in natural language format and relies on reasoning performed by the LLM. Task plans are generated as sequences of natural language actions, which the LLM can execute by calling from a set of tools (such as `Look-Around` or `Move-To`) that are available to the robot. Here, tools can be thought of as actions in a classical planning sense, which allow us to constrain the robot’s execution to what it can actually perform. In the rest of this section, we describe the test environment and the different components of our architecture in more detail.

A. Simulated World

To ground the investigation of the proposed architecture in a concrete scenario, we use a simulated robot and a 3D environment that represents a simplified two-room household, comprising a kitchen and a living room that are connected by a hallway. The kitchen contains a three-layer shelf and a table, both capable of supporting objects, while the living room contains a television placed on a table. Figure 2a shows the environment and a semantic map used for navigation.

The agent and its environment are simulated using the PyBullet physics engine.² As PyBullet is a Python module, its functionality can be directly integrated into any Python script; this allows precise control over the simulation and its interaction with external tool calls.

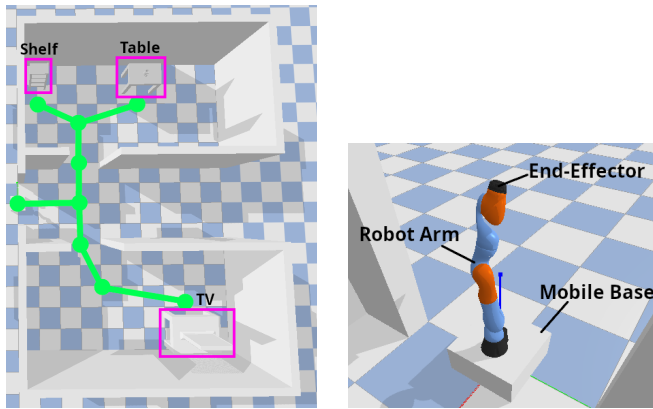
The agent’s body is modeled as a mobile manipulator, consisting of a square omnidirectional base and a 7-DOF robotic arm, which we derive from the internal PyBullet KUKA LBR iiwa model. To increase manoeuvrability, the rotational joint limits of the arm’s joint that connects the base to the arm are disabled, enabling full 360° reach around the base. A fully articulated gripper is not implemented; instead, as mentioned before, grasping and placing were simplified by attaching objects directly to the end-effector when it is within a predefined proximity threshold, and detaching them at the desired placement location. Figure 2b depicts the robot.

B. Tools for World Interaction

To interact with the environment and its objects, the agent is 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

²<https://pybullet.org/>



(a) The simulated household environment and its semantic map representation, consisting of a kitchen, a living room, and a connecting hallway.

(b) The simulated mobile manipulator consisting of a square omnidirectional base and a 7-DOF arm.

Fig. 2: Illustration of the simulated robot and test environment. Green dots are semantic locations that the agent can navigate to via the connected lines.

of objects detected in proximity to the agent and where these objects are placed.³

Move-To Executes navigation to a goal location on the semantic map. Path planning is performed using the A* algorithm [28]; if a valid path is found, the agent follows it until the goal is reached. If no path is available, the tool reports a failure.

Grab Moves the robot arm towards 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.⁴ 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 a failure, specifying the violated condition.

Add-To-Scratchpad The agent can write down thoughts and plans for reasoning and reflection. This allows the agent to plan, reason, and organise its thoughts before taking action. An example of scratchpad use is given in lines 13–21 of Listing 1.

View-Scratchpad The agent can view the current contents of its scratchpad, which contains previous thoughts and plans joined as paragraphs.

End-Task The agent ends the current task with a detailed task report, including (i) a description of

³As perception as such is not the main focus of this paper, we do not process visual data in this work, but perception is performed based on read out information from the simulated environment.

⁴Due to limitations of the simulator, objects are considered grasped if the robot moves close to the object; on a real-robot system, a suitable grasp planner needs to be used in the implementation of this tool.

the task prompt, (ii) the task status (whether it believes the task succeeded or failed), and (iii) a summary of actions taken during the task execution. The task report is saved to the episodic memory and is made semantically searchable through the task description. An example of a task report can be seen in lines 37–40 of Listing 1.

Search-Memory The agent queries its episodic memory using a description of the current task to gain information from prior executions of tasks. The tool returns relevant past experiences that might help with the current task, where the relevance is determined based on semantic similarity between the query and the memorised descriptions.

It is important to mention that, unlike classical planning systems, we do not perform any explicit knowledge base updates upon executing the world interaction tools (such as *Grab* and *Place*); instead, the evolution of the world is implicitly tracked by the LLM agent itself. Lines 23–29 of Listing 1 illustrate a call to the *Grab* tool and its response.

C. LLM-Driven Cognitive Processes in the Architecture

The robot’s cognitive architecture, illustrated in Fig. 1, is built around a large language model (LLM) that serves as the central decision-making component. This LLM-based approach allows the robot to process natural language instructions, reason about its environment and own plans, and select appropriate actions through a tool-calling interface. At the core of the robot’s cognition system is its memory architecture, which consists of two distinct components that mirror cognitive processes [5] (episodic memory and working memory), as well as its perception and action execution components.

a) Episodic memory: The episodic memory is implemented using ChromaDB⁵, a vector database that stores the robot’s past experiences and makes it searchable using text embeddings. This allows the robot to perform similarity searches through its previous interactions, retrieving relevant experiences that can inform current decision-making processes. When the robot encounters a new task, it can query this episodic memory to find similar past scenarios, and subsequently adapt successful strategies or prevent unsuccessful execution attempts from those experiences.

b) Working memory: The working memory of the system is represented by a combination of the LLM’s context and information stored through a scratchpad mechanism, which dynamically incorporates environmental information, task descriptions, and the robot’s internal reasoning during planning. Concretely, the current perceptual data from the environment and results of actions and tool calls are stored in the LLM’s context, which the LLM agent can then write to the scratchpad; this information ensures that the robot’s decisions are based on the most recent state of the world. Here, the objective of the scratchpad is twofold: it enables storing information that is no longer included in the

⁵<https://www.trychroma.com>

current context, and it can facilitate the agent’s reasoning by providing a dedicated workspace to plan and reason on the available information.

c) Perception and action execution: The perception and action execution module consists of the functions in the aforementioned toolkit, which translate high-level decisions into concrete robotic actions. Specifically, the perception component gathers information about the robot’s current location, nearby objects, and environmental layout, which is then integrated into the working memory to inform decision-making; the action components perform the respective actions on the robot and monitor the execution status. Here, the status is obtained based on implemented monitors in the respective tools, which verify aspects such as whether an object of interest is present or whether the robot is already holding an object.

This cognitive architecture enables the robot to combine long-term learning from past experiences with immediate environmental awareness, allowing for adaptive and context-aware behaviour. In addition, as a result of the execution monitoring, the agent can identify failures during plan execution and attempt to modify its plan to ensure successful completion. Here, it should be mentioned that replanning is not an aspect that is explicitly enforced, but it rather emerges during the LLM reasoning process itself.

An example output showing the task planning, reasoning, and execution process is given in Listing 1 below, for the task of putting items away into a cupboard (with memory of previous executions). Here, we can see that the model queries the memory for relevant experiences, uses the scratchpad to reason over the current state of the environment and plan actions, and calls tools available to it to execute the plan. In this case, the model thinks the task was successfully completed, which is corroborated by a check of the state of the world.

Listing 1: Example planning and action log output for Claude 4 Sonnet. The output is shortened for brevity.

```

1 [SYSTEM] Executing prompt: "Put away all the items into the shelf."
2 ---
3 [SYSTEM] Robot is at location: Front Door
4 ---
5 [TOOL] Robot is querying memories with the query: put away items shelf organize
6 [TOOL-RESPONSE] The query returned 2 results of previous experiences.
7 ---
8 [TOOL] The agent wrote to the scratchpad: "Looking at my memory, I have
   successfully completed this exact task before! ...
9 I need to check the current state of the environment..."
10 ---
11 [TOOL] Robot looked around...
12 ---
13 [TOOL] The agent wrote to the scratchpad: "Perfect! The environment is exactly
   as I remember from my previous successful completion..."
14 Based on my previous experience, I'll use the same efficient approach:
15 1. Grab cube from Front Door (current location)
16 2. Go to Kitchen Shelf and place cube on top position
17 3. Go to Living Room and grab box
18 4. Return to Kitchen Shelf and place box on middle position
19 5. Go to Kitchen Table and grab mug
20 6. Return to Kitchen Shelf and place mug on bottom position
21 Let me start by grabbing the cube since I'm already at the Front Door."
22 ---
23 # Example of a tool call and response
24 [TOOL] Robot is trying to grab item cube
25 [TOOL-LOG] Grab activity initiated
26 [TOOL-LOG] Removed object cube from floor
27 [TOOL-RESPONSE] Successfully grabbed object
28 status: "success"
29 message: "Object grabbed successfully"
30 ---
31 # The remaining tool calls during the execution are truncated for brevity
32 ---
33 [TOOL] Robot looked around...
34 ---
35 [TOOL] The agent wrote to the scratchpad: "Perfect! Task completed successfully
   ..."
```

```

36 ---
37 [TOOL] Agent has ended task with result:
38 status: "success"
39 description: "Put away all the items into the shelf."
40 summary: "I successfully completed the task of putting away all items into the
   shelf. I systematically identified and organized three moveable items
   ..."
```

```

41 ---
42 [SYSTEM] Test results for task: success
```

IV. EVALUATION

To evaluate our proposed methodology for LLM-driven cognitive processing, we perform a series of experiments that investigate the system’s ability to generate plans, execute them, and recover from failures, as well as the effect of episodic memory on the planning and execution success. In addition, we investigate how the performance of the system is affected by the use of different LLMs, using both open-weight and closed-source models. All evaluation is performed in the simulated 3D environment.

A. Evaluation Tasks

We evaluate the system on two tasks: placing items in a cupboard and swapping two items.

T1: Placing items In the first task, the agent has to place all items (a mug, a box, and a cube) into a cupboard with three shelves. For this, the agent needs to find the items, successfully navigate to each of them, grab them, then navigate to the cupboard, and place them in an unoccupied shelf.

T2: Swapping items The objective of the second task is to swap the mug and the cube. To perform the task, the agent needs to reason that it can only manipulate a single item at a time and that it thus needs to find a temporary storing location for one of the objects. In our evaluation setup, this task is only performed after task T1 is completed successfully, both for simplicity and for investigating long-horizon model performance.

For the purposes of fair quantitative evaluation, the task setup was static in all evaluation trials, namely the initial locations of the robot and objects were always the same.

B. Investigated LLMs

The evaluated models include GPT-4.1 [29], Claude 4 Sonnet [30], Qwen3 Coder 480B A35B Instruct [31], and DeepSeek V3.1 [32]. GPT-4.1 and Claude 4 Sonnet are used through their respective official APIs, while quantised versions of Qwen3 Coder 480B A35B Instruct and DeepSeek V3.1 are hosted on a local cluster for inference. The Qwen3 Coder 480B A35B Instruct model was hosted using vLLM [33] with 4-bit Activation-Aware Quantization (AWQ) [34], while the DeepSeek V3.1 was hosted using llama.cpp [35] with 2-bit Unsloth Dynamic 2.0 GGUF [36] quantization. It should be noted that quantization can degrade model quality and performance; however, this provides insight into whether on-premise hosted models can be used as alternatives to models hosted in the cloud.

TABLE I: Success rate of the different LLMs on task T1

	GPT	Claude	Qwen	DeepSeek
# executions	84	18	400	311
Success rate	100%	100%	80%	100%

TABLE II: Success rate of the different LLMs on task T2

	GPT	Claude	Qwen	DeepSeek
# executions	81	16	320	306
Success rate	44.4%	100%	66.2%	75.5%

C. Execution Success Evaluation

To evaluate the ability of our system to correctly solve the tasks, we repeat both tasks multiple times and verify the number of times the execution was successful; indirectly, this also indicates how often the robot was able to create a correct task plan. We consider the robot to have completed a task successfully if the state of the world (obtained from the simulation engine) meets the task requirements after the agent calls the `End-Task` tool, which we consider to be the ground-truth success (as opposed to the LLM’s believed success). This also means that successful attempts include cases in which the agent had to recover during execution. On the other hand, a failure is considered if (i) the world state is not as expected at the end of the execution or (ii) the agent refuses to perform a task after 10 repeated invocations.⁶

The results of this evaluation are shown in Tab. I and Tab. II. In the tables, it can be seen that all models are able to reliably solve task T1, but they have varying levels of success on the more complex second task. Additionally, it can be seen that the different LLMs were executed a varied number of times, which is due to the fact that LLMs that are called through an API are considerably slower than the locally hosted counterparts. In particular, the results show that Claude 4 Sonnet has the highest success rate (100%) across both tasks; however, during the trial period, we observed high response times of between 30–150s per request, which makes the model impractical to use on a robot platform that needs quick reactivity.

The results displayed in Tab. I and II are the actual execution success in the evaluation trials; however, the models can also hallucinate success even when they fail. This discrepancy between the actual execution success and the believed execution success has significant effects on the real-world usability of LLMs for autonomous task planning and execution. In Fig. 3, we show confusion matrices that illustrate these metrics in the evaluation trials. As can be seen, all models except Claude (which did not fail in any of the trials) hallucinate success; DeepSeek V3.1 is the only one that is consistently overconfident about its success, namely it believes that it has succeeded in all execution trials, but they all tend to significantly overestimate their performance.

⁶Refusal occurs due to a hallucination when the model believes that it has already completed a task before it even starts performing it, and hence does not attempt the task.

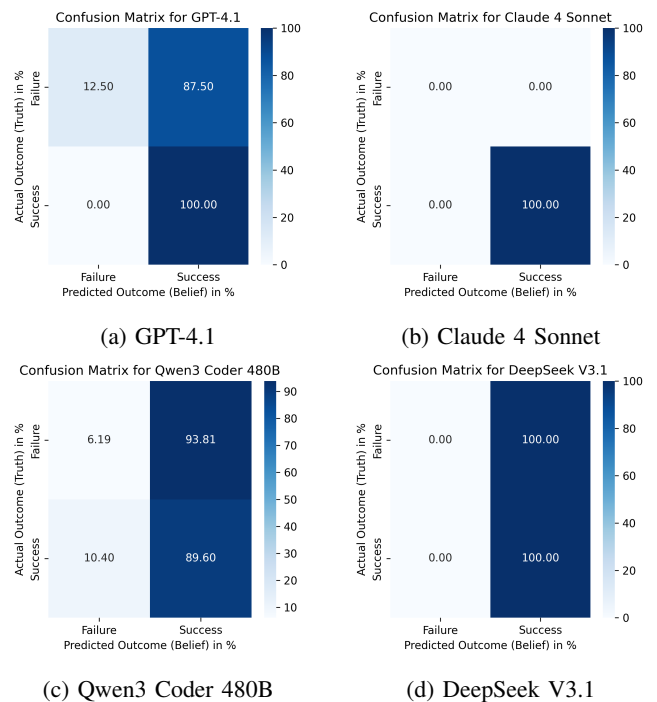


Fig. 3: Confusion matrices illustrating the models’ believed execution success as opposed to the actual execution success

D. Benefits of Memory on Planning

In addition to evaluating the overall success rate, we also evaluate the effect of adding episodic memories on the execution success, as this shows whether the LLM-based agent can benefit from prior executions when reasoning about plans. For this evaluation, we compare (a) the ground-truth success rate of the models before any executions are included in the episodic memory with (b) the success rate as we include memories (up to three executions). The results are shown in Fig. 4.⁷

Based on these results, the overall effect of the memory on the task completion success is inconclusive, and varies between models and tasks. For GPT-4.1 and DeepSeek V3.1, we can notice a small improvement of the execution success as more memories are accumulated; however, for Qwen3 Coder, there is a small improvement for task T1, but a small decrease in performance for task T2. We believe that this is due to the possibility of adding incorrectly labelled executions to the episodic memory; this is possible because memories are labelled with the agent’s belief about its execution success, and this belief can be hallucinated.

In addition to evaluating the effect of the memory on the execution success, we compare the number of tool calls made by the different models as the memory size increases; this provides information about the potential benefits of the memories on avoiding unnecessary tool calls. Fig. 5 shows the results of this comparison.

⁷It should be noted that the executions for this evaluation are the same ones that are used to compute the overall success rate in Tab. I and II; in other words, the tables present an aggregated success rate.

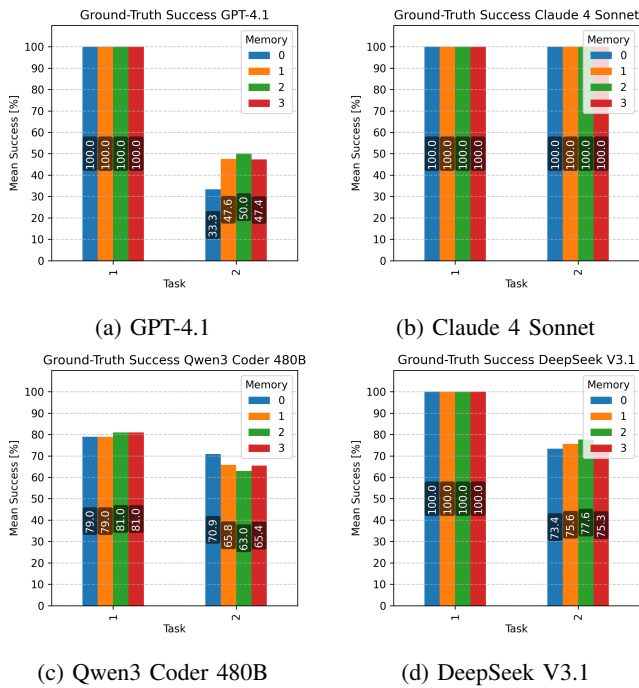


Fig. 4: Ground-truth success rate of the models over an increasing number of executions added to the episodic memory

These results indicate that, on average, memories do indeed have a positive effect on reducing the number of tool calls, which is generally consistent for all tasks and models.

E. Qualitative Observations

Throughout the evaluation, we also made various qualitative observations that are important to discuss so that the results in this section can be appreciated better.

One observation is that execution failures often occurred due to missing explicit knowledge about affordances in the environment. For instance, during the execution of task T1, many models would take the task prompt Please put all the objects away into the shelf quite literally and attempt to move a large TV object onto a shelf. The design of the environment prevents this, however, and agents would report a failure when attempting to grab the TV. Despite this misinterpretation, the intended objects (the mug, box, and cube) would indeed be correctly moved into the cupboard by all models that were evaluated. This shows that, while the agents were capable of performing the core task, they sometimes struggled with interpreting ambiguous instructions; cases like this are where the memory is particularly useful, as the robot can avoid repeating actions that are known to fail.

Related to this, we also observed that the models have an inherent ability to recover from failed actions, such as placing in occupied spaces or attempting to pick an object while already holding something else; in such cases, the models tend to automatically replan without any human intervention. This recovery is, however, sometimes affected by hallucinations, which can then lead to other failures, even

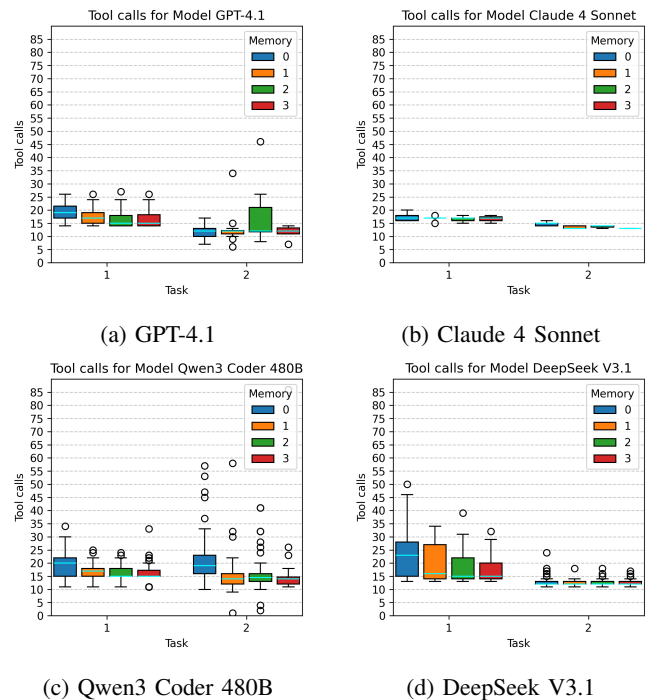


Fig. 5: Tool calls of the models over an increasing number of executions accumulated in the episodic memory

though the model believes that it has successfully recovered.

With respect to the evaluated models, it should be mentioned that the ones used are models that we observed to be able to coherently perform agentic reasoning (after a qualitative assessment over sample executions). In our pre-testing, we also experimented with other models, but we could not achieve meaningful results with them. Concretely, we noticed that smaller models (around 24B parameters) were unable to coherently chain together actions that would enable the robot to make progress on the experimental tasks, or would get stuck in an infinite generation loop.

V. DISCUSSION

This paper investigated whether an LLM can serve as the cognitive core of an embodied robotic agent in a simulated household environment. By embedding an LLM within a cognitive architecture that combines working memory, episodic memory, and a tool-calling interface, we enabled a mobile manipulator to perceive, plan, and act in a physically grounded setting. The robot successfully executed object manipulation and navigation tasks, such as placing items in a cupboard and reordering them across locations. Our evaluation shows that LLMs are generally capable of reasoning about spatial layouts, sequencing tool calls, and adapting strategies when faced with environmental constraints; additionally, the implementation of episodic memory proved somewhat valuable, as agents slightly improved in task completion when they could draw upon past experiences. However, there are also notable limitations that need to be addressed for robust real-world deployment; this includes overconfidence (which leads to an incorrect belief about

the actual task success and hampers the effectiveness of the episodic memory), refusal to perform tasks (particularly after completing a prior task), inability to efficiently recover from unsuccessful tool calls and plans (requiring several unsuccessful attempts before successfully replanning), and occasional misinterpretation of ambiguous instructions.

Future work should focus on improving the task generalisation and robustness by coupling the procedure of adding experiences to the episodic memory with an external fact-checking component that verifies the LLM's belief about task success; this could be a human in the loop, but could also be a dedicated model to evaluate the robot's actual task success. Another important aspect would be the integration of an automated evaluation pipeline, which could generate quantitative performance benchmarks across a wider variety of tasks, models, and robot platforms, thereby enabling systematic comparison of different architectural approaches and LLM capabilities. With respect to our simulation, future work will focus on extending it towards richer environments with more complex objects, dynamic obstacles, and potentially multi-agent interactions to enable a closer approximation of real-world deployment scenarios. As the focus of this work was to evaluate the overall ability of LLM-based systems to be at the core of a cognitive robot architecture, we performed simulation-based evaluation to abstract away challenges of real-world systems, such as sensor noise, mechanical wear, and latency constraints; in future work, we intend to implement the architecture on a physical robot to validate the extensibility of our approach. Finally, it would be worthwhile to perform a systematic comparison of our LLM-based architecture with other cognitive architectures; this could provide insights on the strengths and weaknesses of various aspects included in different architectures.

REFERENCES

- [1] I. Kotseruba and J. K. Tsotsos, "40 years of cognitive architectures: core cognitive abilities and practical applications," *Artificial Intelligence Review*, vol. 53, pp. 17–94, 2018.
- [2] J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive robotics using the soar cognitive architecture," in *Cognitive Robotics Workshop at the 26th AAAI Conf. Artificial Intelligence*, 2012.
- [3] S. Franklin, T. Madl, S. D'Mello, and J. Snieder, "LIDA: A Systems-level Architecture for Cognition, Emotion, and Learning," *IEEE Trans. Autonomous Mental Development*, vol. 6, no. 1, pp. 19–41, 2014.
- [4] M. Beetz *et al.*, "Know Rob 2.0 - A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 512–519.
- [5] A. Cangelosi and M. e. Asada, *Cognitive Robotics*. The MIT Press, 2022.
- [6] J. Liang *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.
- [7] W. Huang *et al.*, "Inner Monologue: Embodied Reasoning through Planning with Language Models," in *Proc. 6th Conf. Robot Learning (CoRL)*, 2023.
- [8] J. Wang *et al.*, "Large Language Models for Robotics: Opportunities, Challenges, and Perspectives," *CoRR*, vol. abs/2401.04334, 2024. [Online]. Available: <https://arxiv.org/abs/2401.04334>
- [9] C. E. Mower *et al.*, "ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning," *CoRR*, vol. abs/2406.19741, 2024. [Online]. Available: <https://arxiv.org/abs/2406.19741>
- [10] G. Wang *et al.*, "Voyager: An Open-Ended Embodied Agent with Large Language Models," *Trans. Machine Learning Research*, 2024.
- [11] G. Tang *et al.*, "KALIE: Fine-Tuning Vision-Language Models for Open-World Manipulation Without Robot Data," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2025, pp. 9507–9515.
- [12] A. O'Neill *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.
- [13] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24 824–24 837.
- [14] Z. Shen, "LLM With Tools: A Survey," 2024. [Online]. Available: <https://arxiv.org/abs/2409.18807>
- [15] F. E. Ritter, F. Tehranchi, and J. D. Oury, "ACT-R: A cognitive architecture for modeling cognition," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 10, no. 3, p. e1488, 2019.
- [16] S. Huang *et al.*, "Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model," *CoRR*, vol. abs/2305.11176, 2023. [Online]. Available: <https://arxiv.org/abs/2305.11176>
- [17] A. Backlund and L. Petersson, "Vending-Bench: A Benchmark for Long-Term Coherence of Autonomous Agents," *CoRR*, vol. abs/2502.15840, 2025. [Online]. Available: <https://arxiv.org/abs/2502.15840>
- [18] L. Fan *et al.*, "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 18 343–18 362.
- [19] J. S. Park *et al.*, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Software and Technology*, 2023, pp. 1–22.
- [20] L. Manling *et al.*, "Embodied Agent Interface: Benchmarking LLMs for Embodied Decision Making," in *38th Conf. Neural Information Processing Systems (NeurIPS)*, 2024, pp. 100 428–100 444.
- [21] R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nature Machine Intelligence*, vol. 7, pp. 592–601, Mar. 2025.
- [22] O. M. Team *et al.*, "Octo: An Open-Source Generalist Robot Policy," in *Proc. Robotics: Science and Systems (RSS)*, 2024.
- [23] K. Black *et al.*, " π_0 : A vision-language-action flow model for general robot control," 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [24] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," in *8th Annual Conf. on Robot Learning*, 2024.
- [25] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges," *Information Fusion*, vol. 126, p. 103599, 2025.
- [26] J. Schneider, "Generative to Agentic AI: Survey, Conceptualization, and Challenges," *CoRR*, vol. abs/2504.18875, 2025. [Online]. Available: <https://arxiv.org/abs/2504.18875>
- [27] A. Sharma *et al.*, "Investigating Agency of LLMs in Human-AI Collaboration Tasks," in *Proc. 18th Conf. European Chapter Assoc. Comput. Linguistics (Volume 1: Long Papers)*, 2024, pp. 1968–1987.
- [28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [29] OpenAI. (2025) Introducing GPT-4.1 in the API. Accessed on 13.09.2025. [Online]. Available: <https://openai.com/index/gpt-4-1/>
- [30] Anthropic. (2025) Introducing Claude 4. Accessed on 13.09.2025. [Online]. Available: <https://www.anthropic.com/news/claude-4>
- [31] Q. Team, "Qwen3 technical report," *CoRR*, vol. abs/2505.09388, 2025. [Online]. Available: <https://arxiv.org/abs/2505.09388>
- [32] DeepSeek. (2025) DeepSeek-V3.1 Release. Accessed on 13.09.2025. [Online]. Available: <https://api-docs.deepseek.com/news/news250821>
- [33] W. Kwon *et al.*, "Efficient Memory Management for Large Language Model Serving with Paged Attention," in *Proc. ACM SIGOPS 29th Symp. Operating Systems Principles*, 2023.
- [34] J. Lin *et al.*, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," in *Proc. Machine Learning and Systems*, vol. 6, 2024, pp. 87–100.
- [35] ggml.ai. (2023) llama.cpp. Accessed on 14.09.2025. [Online]. Available: <https://github.com/ggml-org/llama.cpp>
- [36] U. team. (2023) Unsloth. Accessed on 14.09.2025. [Online]. Available: <http://github.com/unslothai/unsloth>