8th Conference on Production Systems and Logistics

LLM-Based Multimodal Prompting For Adaptive Robot Control In Production Systems

Dominik Koch[1,†], Jakob Wolber[1,†], Zhuo Shi[3], Bo Cheng Ji[3] , Lucas Bretz[2], Alexander Geiser[1], Felix Baer[3], Martin Benfer[1], Florian Stamer[4], Gisela Lanza[1,2]

[1]*wbk Institute of Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76133 Karlsruhe, Germany*
[2]*Global Advanced Manufacturing Institute (GAMI, KIT China Branch), 10 Yueliangwan Road, Suzhou 21523, China*
[3]*Siemens Limited, China, 18 Dongfang park, Jinfang Road, Suzhou 215129, China*
[4]*Leuphana University, Universitätsallee 1, 21335 Lüneburg, Germany*
† Both authors contributed equally

Abstract

Large Language Models (LLMs) open new opportunities for adaptive automation in production systems by enabling robots to interpret human instructions and generate context-aware actions. In contrast to conventional robot programming, which requires expert knowledge and frequent reconfiguration, LLM-based control promises greater flexibility and easier interaction between humans and machines. However, generic LLMs still face major challenges when applied to manufacturing environments, as they lack grounding in real-world perception and may produce infeasible or unsafe actions. This paper presents a laboratory demonstrator that evaluates how different prompting strategies affect the performance of an LLM-controlled pick-and-place robot. The study systematically compares zero-shot and multimodal few-shot prompting, where visual examples such as annotated video frames and image captions are integrated into the LLM input. A dedicated evaluation model with metrics for plan success, action success, and plan optimality is used to quantify system behavior. The experimental results demonstrate that multimodal few-shot prompting significantly improves planning accuracy, robustness, and adaptability compared to a zero-shot baseline. These findings illustrate the potential of LLM-driven control for future intelligent production systems that combine semantic reasoning, multimodal perception, and human-interpretable automation.

Keywords

Large Language Models (LLMs); Adaptive Production Systems; Multimodal Few-Shot Prompting; Intelligent Robot Control; Human-Interpretable Automation

1. Introduction

Recent advancements in natural language processing (NLP) with LLMs like generative pre-trained transformers (GPT) have gained a lot of momentum in 2024. These models generate human-like texts, understand context, and enable meaningful conversations [1]. Alongside the growth of LLMs, prompt engineering has emerged as a technique to extend the capabilities of LLMs [2]. Integrating LLMs into robotics is reshaping human-robot interactions but is not trivial [3]. In conventional industrial robotics, tasks like pick-and-place are typically pre-programmed, which requires a reconfiguration when adjusting to new environments or tasks. Another disadvantage is that in traditional robotics, specialized experts are required, which increases the complexity and cost of deployment. LLM powered robots, on the contrary, can interpret natural language, adapt to real-time conditions, and generate actions based on contextual understanding, reducing the need for constant reprogramming. This shift enables robots to handle dynamic, ambiguous

DOI: https://doi.org/10.15488/20980
ISSN: 2701-6277

situations, making them more flexible and autonomous [4] Current research on LLMs in robotics focuses on task planning and action derivation, but lacks progress in feedback, multimodal input, and generalization [5]. This work focuses on a proof-of-concept environment simulating pick-and-place tasks with wooden blocks, providing a valuable testbed for evaluating how LLMs can control robotic systems in dynamic environments. In experiments, the LLM receives and interprets natural language commands and determines both the appropriate tools and their execution sequence. These tools include parametrized Python functions for gripper control or robot movement, chosen based on multimodal inputs such as annotated video frames or sensor data. By exploring these structured yet flexible tasks, the study investigates how LLMs can contribute to more adaptive robotic systems, particularly in scenarios requiring frequent task adjustments and natural human–robot interaction. Using prompt engineering and multimodal input techniques, the impact of few-shot and multimodal prompting on LLM performance in robotic control is assessed. This work introduces an evaluation model with generic tasks, manipulation families, and performance metrics to evaluate LLM effectiveness in robotics. Performed experiments quantify zero-shot performance and show improvement with multimodal few-shot prompting, particularly using annotated video frames as solution examples. Additionally, these experiments show that image captioning enhances object detection, thereby improving the LLM's task-solving capabilities.

2. Fundamentals and Related Work

2.1 Fundamentals

An LLM generates responses by calculating the most probable next tokens based on the statistical distribution of tokens in the training data [6] Although LLMs are generally capable, they must be adapted for specific tasks or domains. This is done through prompt engineering and/or finetuning. Fine-tuning updates a pre-trained neural network's weights with task-specific data, adjusting the model's parameters, creating specialized models. However, it requires labeled data, time, computational resources, risks overfitting, and is challenging for tasks with changing data or specifications [7]. Prompt engineering designs and refines prompts to get specific responses from LLMs without adjusting the parameters of the model [8]. Prompts are generally given within a template, which is a predefined structure that organizes the input into clearly defined sections [5]. When only the prompt is provided, without additional examples, it is referred to as a "zero-shot". However, when the prompt includes additional input, such as solution examples, it is called a "few-shot" prompt [9]. Despite their strengths, LLMs struggle with simple tasks like factual lookup. This limitation can be overcome by allowing LLMs to use external tools, which extend their capabilities beyond text generation and allow them to perform real-world actions [10]. An external tool can include an application programming interface, a database, hardware components such as a robotic gripper, or a module that integrates multiple tools to handle complex tasks. In this context, LLMs function as a reasoning hub, using their decision-making ability to interact with and access these tools [11].

The following illustrates the LLM agent concept with the example request: *„pick-up the green block"*. The LLM agent receives a user request and available tools, e.g., a *json*-file describing available Python functions to the agent. The agent then creates a plan, a sequence of tool calls. Since the LLM predicts tokens, tool selection is probabilistic. Tool execution can be deterministic or probabilistic, depending on the tool. The tool call returns an outcome that the agent processes before selecting the next tool. The agent loop continues until the agent’s final observation fulfills the initial user request. Vision-language models (VLMs) are a popular type of multi-modal networks in current research [12]. For example, computer vision, like object detection, is crucial in robotics [13], but traditional closed-set systems can only detect predefined categories and need new labeled data to add others [14]. Detecting arbitrary objects specified by human language is called open-set object detection [15]. VLMs merge computer vision and NLP and enable arbitrary objects detection specified by human language input. Among other things, VLMs can take images and text as input

and generate text, such as describing visual content or annotations. They are used for tasks like image captioning, object detection, and visual question-answering [13].

2.2 Related Work

This section summarizes recent approaches for structured prompting for code generation in task planning, grounding in the environment, granular robot actions, retrieval-augmented prompting, and using the Planning-Domain-Definition-Language (PDDL) in robotics task planning with LLMs. However, some publications can belong to multiple categories.

Structured prompting for code generation in task planning: Publications structure prompts using action primitives, system roles or environment information to improve LLM-generated code accuracy [16-20]. Strategies include XML-tagged outputs and corrective dialogues for better task planning and error handling [19], JSON templates [20] or VLM to interpret human demonstration videos and generate robot task planning [21].

Grounding the environment: Grounding robots in the real world and verifying action feasibility through environmental feedback are essential for reliable task execution [22,23]. LLMs use textual descriptions, perception APIs, or vision models to interpret environments and ensure feasibility [24]. Methods such as affordance functions [22], Q-function verification [23], and iterative feedback loops improve plan quality and execution accuracy [25]. Luan et al. employ Q&A-based few-shot prompting supported by VLM object detection; one LLM decomposes user requests, while another refines them into grounded tasks [26]. Yang et al. introduce a re-planning prompt that adapts task plans using environmental feedback via CoT reasoning, while ToT offers no performance benefit and increases latency [27]. C. Wang et al. translate multimodal cues—human behaviour, position, gaze, dialogue, and scene information—into language for LLM-based action coordination [28]. Other VLM approaches focus on replanning [29], or replace separate vision and reasoning systems with a unified multi-agent VLM framework [30].

Granular robot actions: LLMs often focus on high-level planning and invoke a limited repertoire of action primitive and struggle with low-level motion planning [31,32]. To bridge this gap, methods integrate kinematic-aware prompting to generate motion trajectory waypoints [32] or structured action token for a policy network to generate precise movement commands [33]. Other approaches use 3D~value maps for motion planning [31] and visual marks to images to enhance grounding in multimodal models [34]. Brohan et al. highlights the promise of VLMs and LLMs but note that their high-level focus results in separate low-level controllers missing rich semantics. Vision language action models (VLA) are introduced to decode actions from images and language. VLAs are fine-tuned VLMs and can be integrated with LLMs in robotics and improved using techniques like CoT reasoning [35].

Retrieval-augmented prompting: The quality of few-shot examples is crucial [36,37]. Retrieval-augmented prompting selects relevant examples from databases to improve few-shot learning and task execution [2]. Enhancements like k-nearest-neighbor retrieval and logit biases [36], and human feedback [37] improve example selection.

Plannnig-Domain-Definition-Language: PDDL is a standardized language for defining classical planning problems using logical expressions and structures. Combining PDDL with LLMs can enhance the performance of planners by ensuring actions follow a clear, predefined structure. LLMs translate natural language requests into structured PDDL action plans [38-40] and a classical solver to generate accurate plans [41].

3. Approach

3.1 Problem Statement

Generic LLMs cannot produce executable robot task plans without adaptation, typically via prompt engineering or fine-tuning. This work focuses on prompt engineering. Although integrating an LLM agent with external robotic tools is feasible, achieving reliable performance remains challenging. Improvements can come from stronger models, better tools, or tailored prompts, but model and tool development requires significant resources, and heavily customized prompts risk overfitting. Few-shot multimodal examples, such as video demonstrations, may support more efficient generalization, though their use in robotics is still under investigation [5]. This experiment compares zero-shot performance with (multimodal) few-shot prompting.

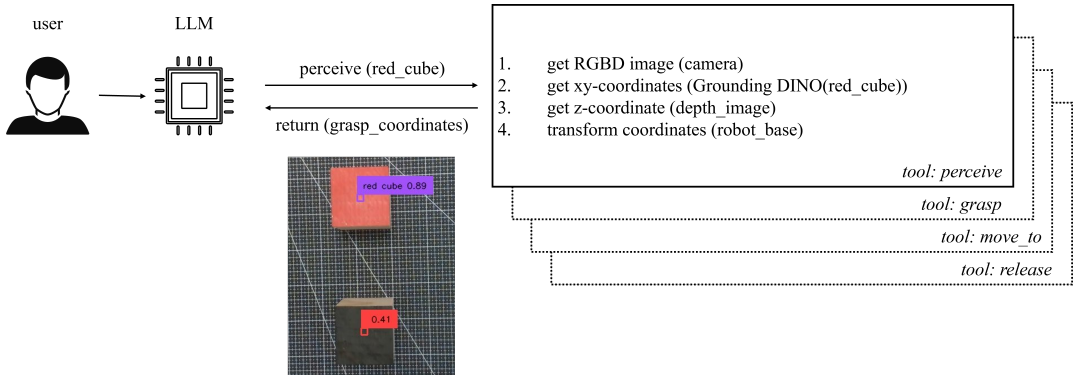


Figure 1: Object detection module: The perceive function uses an RGB-D camera with Grounding-DINO

3.2 Architecture

The proposed approach integrates an LLM agent with external tool execution for robotic control. The system consists of the LLM agent, a UR10e robot, a suction gripper, and an object detection module. Alibaba's Qwen-Max model (temperature 0.95) was used via the Alibaba Cloud API, selected for its strong tool-calling and parameter-selection capabilities [42]. The agent receives a *tools.json* file defining four available tools—*move_to(x,y,z)*, *perceive("object")*, *grasp()*, and *release()*—implemented as Python functions mapped to URScript commands. In this work, "tools" refers to these Python functions, although the term can be broader (Section 2.1). Figure 1 shows the object detection module used by *perceive()*, which outputs the top-center grasp point of the target object. GroundingDINO [15] provides the bounding boxes, and an Intel RealSense D435 RGB-D camera supplies the visual data. For image captioning GPT-4o [43] was used, generating textual descriptions of images [44]. The proposed prompting pipeline consists of six elements: input data (user request), instruction (system role), output indicator (desired format), context (e.g., available tools), solution examples (zero-shot or few-shot), and feedback (re-prompting), following prior work [5].

3.3 Experimental Setup

3.3.1 Metrics

Three metrics are introduced and explained below: Plan success rate (PSR), action success rate (ASR) and optimal plan rate (OPR).

PSR: Defined as the fraction of executions in which the agent produces a correct plan, defined as a tool sequence that reaches the task's goal state. Any additional steps hallucinated after achieving the goal render the plan incorrect. Tool parameters and robot execution are assumed perfect, and plan optimality is not considered. This definition aligns with prior work [25, 45, 46].

ASR: The fraction of executions with correct tool parameters. It evaluates only the agent's selection of Python function parameters under the assumption of flawless tool execution, ignoring external disturbances

such as power loss or camera miscalibration. This separation of plan- and action-level evaluation is common in LLM-based robotics, where execution success is often emphasized [25, 46, 17].

OPR: Is defined as the fraction of executions that generate an optimal plan, defined as using the correct order and the minimum number of tool calls. Non-optimal plans include unnecessary steps or incorrect ordering. Because optimality is stricter than correctness, OPR is always ≤ PSR. Related metrics like path length [27] assess similar aspects, however, the tool-call order is additionally taken into account to minimize the duration an object remains in the gripper, thereby reducing the risk of collisions and gripper failures.

Table 1: Generic evaluation model: Task, their generalization and manipulation family

Task	**Generalized task**	**Manipulation family**
Locate the green block	*direct verb*, *direct object*	Specific, direct and single object
Pick-up the green triangle		
Put the red cube on top of the blue cube	*direct verb*, *direct object*, *direct spatial relation*, *direct object*	Specific, direct and multiple objects
Place the cube with the color of the ocean on the paper with the sum of 3 and 4	*direct verb*, *indirect object*, *direct spatial relation*, *indirect object*	Specific, indirect and multi object
Sort the blue and red cube onto the according paper	*indirect verb*, *direct object*, *direct spatial relation*, *indirect object*	
Build a 3-block tower in the order of the Kenya flag	*indirect verb*, *indirect object*, *indirect spatial relation*	
Sort the cubes by color onto the papers	*indirect verb*, *indirect object*, *direct spatial relation*, *indirect object*, *unspecific*	Unspecific, indirect and multi object
Put the red cube on top of the blue cube	*direct verb*, *direct object*, *direct spatial relation*, *direct object*, *unspecific*	

3.3.2 Tasks

To evaluate the LLM agent, a diverse task set is required. Therefore, a generic evaluation framework in which tasks are constructed from fundamental building blocks (Table 1) is proposed. The manipulation taxonomy distinguishes tasks along three dimensions: *specificity* (specific vs. unspecific), *directness* (direct vs. indirect), and *object multiplicity* (single vs. multi-object). This modular design allows researchers to assemble tasks suited to their setups using predefined components. Indirect and unspecific instructions are considered as they are unavoidable in human–robot interaction, highlighting the importance of the more challenging task types at the bottom of Table 1. For task construction, seven elements are defined: direct verb, direct object, direct spatial relation, indirect verb, indirect object, indirect spatial relation, and unspecific. Each task contains at least a verb and an object; spatial relations appear when multiple objects are involved. Direct tasks provide explicit instructions, whereas indirect tasks require inference or simplification (e.g., “cube with the colour of the ocean” → “blue block”). The final two examples in Table 1 illustrate unspecific tasks where environmental context is essential. In task 7, missing information about object colours or quantities prevents direct execution, as the agent would otherwise guess. In task 8, the red cube is unreachable.

Table 2: Summary of the different scenarios and the respective prompt design

Scenario	**Solution examples**	**Multimodal input**	**Environment perception**
I	None	-	Object detection module
II	Manually created text examples	-	Object detection module
III	Auto-generated text examples from unannotated video frames	Video frames	Object detection module

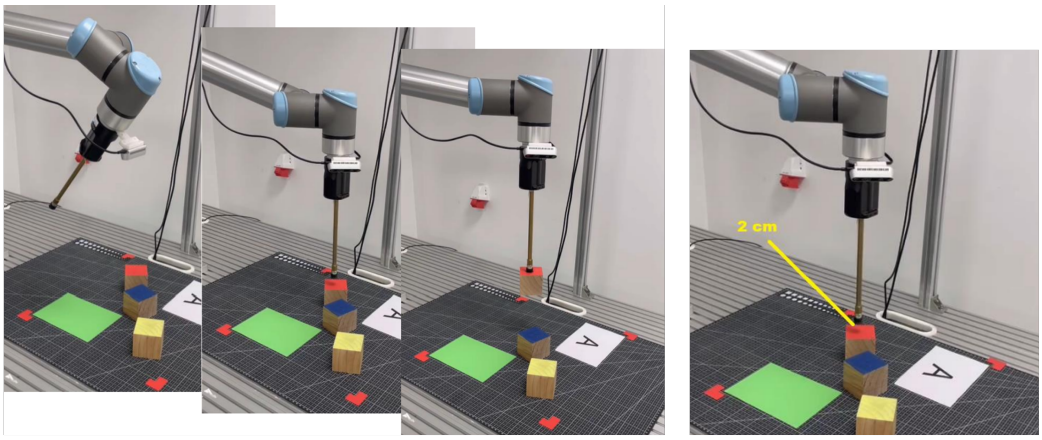
IV	Auto-generated text-examples from annotated video frames	Annotated video frames	Object detection module
V	Manually created text examples	Initial environment image	Object detection module and image captioning

3.3.3 Scenarios

Five performance comparison scenarios are introduced, each with a different prompt for the LLM agent. The scenarios were chosen to explore LLM performance. A zero-shot baseline is established and compared to textual few-shot prompting, as well as three different forms of multimodal few-shot prompting. Table 2 summarizes the different scenarios and the respective prompt designs.

- **Scenario 1** sets a zero-shot baseline where no examples are provided. The LLM agent operates without prior examples, relying solely on the object detection module.
- **Scenario II** evaluates performance using manually created text examples. These examples are provided in pseudo-code format with reasoning steps, as this format performs better than plain text or *json*. Similarly to scenario I, only the object detection module is used.
- **Scenario III** introduces multimodal input in the form of video frames. The LLM auto-generates solution examples based on unannotated video frames of a robot solving a task. Figure 2a illustrates an example sequence of unannotated video frames used in this scenario. The perception of the environment remains based on the object detection module.
- **Scenario IV** builds on scenario III by incorporating annotated video frames. The annotations provide key details, such as correct offsets for grasping objects, as shown in Figure 2b. The LLM uses these annotated frames to generate solution examples, again relying on the object detection module.
- **Scenario V** demonstrates the impact of integrating an image captioning module. This scenario combines the manually created text examples from scenario II with an image captioning module that describes the initial environment, such as available objects. The agent, in this case, has access to both the object detection module and the image captioning system.

In all five scenarios, the agent receives the same *tools.json* file and operates under the same system role.



a) Exemplary frame sequence for picking up the red cube

b) Example annotation of a video frame explaining a 2 cm offset

Figure 2: Visualization of the pick-up process (a) and (b) an annotation highlighting a 2 cm offset

4. Experimental Results

Due to the inherent randomness of LLM-based decision making, the agent's results varied even with the same environment and prompt. Each task was performed 10 times per scenario. The experiments were carried out in August 2024. The results for each metric per respective scenario are shown in Table 3, where

"1.0" indicates success in all 10 cases, and "0.0" indicates complete failure. Among the different scenarios, a brief overview is provided, which will be analyzed in detail in section 4.1.

- **Scenario V** demonstrates the highest overall performance across all three metrics. It particularly outperforms other scenarios in ASR and OPR, demonstrating the most reliable execution, especially for unspecific tasks.
- **Scenarios II** and **scenario IV** rank second, performing well in simpler tasks but struggling with more complex tasks at the end of the table.
- **Scenarios III** and **scenario I** perform similarly badly, with **scenario III** being slightly better than **scenario I**. The weaker performance can be seen particularly in ASR and OPR, where **scenario III** and **scenario I** only perform well for the simplest tasks of locating a single block.

Across all scenarios, several recurring issues were observed: hallucinated plans with unnecessary or missing steps, offset errors in vertical positioning during grasps, stacking failures caused by neglecting object heights, delayed object detection that forces repeated camera movements, and a general lack of environmental awareness in unspecific tasks where additional context is required but not inferred.

Table 3: Experiment Results for the metrics PSR, ASR and OPR for respective scenarios I-V

Task	Plan Success Rate (PSR)					Action Success Rate (ASR)					Optimal Plan Rate (OPR)				
Scenario	I	II	III	IV	V	I	II	III	IV	V	I	II	III	IV	V
Locate the green block.	0.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	1.0	1.0	1.0	1.0
Pick-up the green triangle.	0.2	1.0	1.0	1.0	1.0	0.1	1.0	0.0	1.0	1.0	0.2	1.0	1.0	1.0	1.0
Put the red cube on top of the blue cube.	0.8	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0
Place the cube with the colour of the ocean on the paper with the sum of 3 and 4.	0.6	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0
Sort the blue and red cube onto the according paper.	0.7	1.0	1.0	0.9	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.8	0.0	0.7	0.9
Build a 3-block tower in the order of the Kenya flag.	0.7	0.8	0.9	0.9	0.9	0.0	0.8	0.0	1.0	1.0	0.0	0.8	0.0	0.8	0.9
Sort the cubes by colour onto the papers.	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.9
Put the red cube on top of the blue cube.	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0

4.1 Findings

Scenario I: The zero-shot agent generates plausible task plans but lacks reliability and parameter accuracy. It achieves a PSR > 0 for 6 of 8 tasks. However, PSR ranges from 0 to 1, indicating high variance in plan generation. Parameter selection is frequently incorrect, particularly for z-coordinates during grasping and stacking, resulting in an ASR of 0 for 6 tasks. Task plans are suboptimal, as objects are not fully localized at initialization and existing structures are not reused, leading to unnecessary tool calls. Unspecific tasks fail due to missing environment information.

Scenario II: Manually defined text examples substantially improve performance. All simple tasks are solved correctly and optimally (ASR = 1). In complex long-horizon tasks, the number of errors decreases significantly, although failures such as hallucinations, delayed error detection, and incorrect stacking still occur. These errors are attributed to long-horizon dependencies, where available information is not consistently applied. While the agent learns effective parameter patterns, it exhibits limited spatial

understanding and mainly reproduces few-shot examples. Unspecific tasks remain unsolved (PSR = 0 for tasks 7 and 8).

Scenario III: Scenario III performs better than the zero-shot baseline but worse than Scenario II. Reliability improves, hallucinations are reduced, and task plans are more coherent, reflected in higher PSR values. For example, tower-building plans correctly reuse existing blocks. However, generated examples lack critical parameter details, such as accurate z-coordinates and initial object localization, resulting in offset and detection failures. Consequently, ASR remains comparable to Scenario I. Unspecific tasks again fail due to insufficient environment information.

Scenario IV: Providing annotated frames yields performance comparable to Scenario II. Differences between both scenarios are minor and attributed to LLM randomness rather than systematic effects. Visual annotations improve performance over Scenarios I and III, particularly in ASR and OPR. However, limitations observed in Scenario II persist, including overfitting to provided examples and reduced robustness in long-horizon tasks. Unspecific tasks still fail (PSR = 0 for tasks 7 and 8).

Scenario V: Scenario V performs on par with Scenarios II and IV for specific tasks but shows a clear improvement for unspecific tasks. Image captioning supplies essential environmental information, resulting in a PSR of 1 for tasks 7 and 8. This scenario demonstrates that multimodal perception is necessary to handle underspecified tasks and cannot be compensated by prompting strategies alone.

4.1.1 Limitations

Experimental findings are limited to the setup used, affected by factors like model choice, temperature setting, and *tools.json* configuration. Testing was done with simple cubes in obstacle-free environments, which may reduce their applicability to more complex settings. The agent relies heavily on the few-shot examples provided and might overfit, which restricts its ability to adapt and generalize across new, untested tasks. Creating and annotating effective examples manually is time-consuming and lacks standardization, making the process dependent on manual effort and expertise. Another limitation is the fact that the agent does not have a true understanding, and instead just successfully copies patterns from examples.

4.1.2 Outlook

To further improve the LLM-integrated robotic system, several experiments and architectural enhancements are proposed. A retrieval-augmented generation database with task-specific examples could reduce reliance on generic responses, mitigate overfitting, and improve adaptability. Limitations in 2D spatial reasoning may be addressed through refined prompting, multimodal inputs such as visual cues with coordinate systems, or the integration of simulation and digital twins that provide explicit spatial references and enable adaptive corrections. For complex tasks, mechanisms to detect and resolve contradictory user instructions are needed, particularly in human–robot interaction, potentially extending the introduced manipulation taxonomy. Although few-shot learning reduces hallucinations, model unpredictability remains a safety concern, requiring constraints and safety checks to ensure reliable operation. Given the rapid progress in VLMs and reasoning models, further improvements are expected in handling ambiguous task descriptions, spatial understanding, and long-horizon planning, including through VLM-enabled video-based learning.

5. Conclusion

This work explores the integration of LLMs into pick-and-place robots, focusing on the effectiveness of zero-shot and multimodal few-shot prompting. Key findings highlight performance improvement of few-shot prompting with multimodal inputs, such as annotated video frames. Image captioning helps robots to be grounded in their environment and solve unspecific tasks. Few-shot prompting faces challenges like trial and error and overfitting. Spatial reasoning remains a key issue for LLM agents. Nevertheless, LLMs enable human-robot interaction through language capabilities and multimodal input processing. This highlights the

potential for future production systems, deploying and adapting robots primarily through natural language instructions and extensive prior knowledge, reducing the need for task-specific programming.

Acknowledgements

This research and development project is funded by the Federal Ministry of Research, Technology and Space (BMFTR) within the "SME-innovative: The Future of Value Creation" funding measure (funding number 02K24K021) and managed by the Project Management Agency Karlsruhe (PTKA). It is also funded by the National Key R&D Program of China, under grant no. 2022YFE0114100. The authors are responsible for the content of this publication.

References

[1] Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., et al., 2024. A survey of large language models. arXiv preprint arXiv:2303.18223.

[2] Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A., 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927.

[3] Zhang, C., Chen, J., Li, J., Peng, Y., Mao, Z., 2023. Large language models for human–robot interaction: A review. Biomimetic Intelligence and Robotics 3 (4), 100131.

[4] Liu, H., Zhu, Y., Kato, K., Tsukahara, A., Kondo, I., Aoyama, T., et al., 2024. Enhancing the LLM-based robot manipulation through human-robot collaboration. IEEE Robotics and Automation Letters 9, 6904–6911.

[5] Wolber, J., Muyang, L., Koch, D., Bretz, L., Lanza, G., Baer, F., 2025. Large Language Models for Robotics: A systematic literature review on prompt engineering. In: Min, J., Zhang, W., Fleischer, J., Lanza, G. (Eds.), Sustainable Manufacturing Innovations: Focus on New Energy Vehicles, Production Robots, and Software-Defined Manufacturing. Lecture Notes in Production Engineering, ICSM 2024. Springer, Cham.

[6] Shanahan, M., 2023. Talking about large language models. arXiv preprint arXiv:2212.03551.

[7] Amaratunga, T., 2023. Understanding Large Language Models: Learning Their Underlying Concepts and Technologies. Apress, Berkeley, CA.

[8] Aljanabi, M., Yaseen, M.G., Ali, A.H., Mohammed, M.A., 2023. Prompt engineering: Guiding the way to effective large language models. Iraqi Journal for Computer Science and Mathematics, 151–155.

[9] Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y., 2023. Large language models are zero-shot reasoners. arXiv preprint arXiv:2205.11916.

[10] Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., et al., 2023. Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761.

[11] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., et al., 2024. A survey on large language model based autonomous agents. Frontiers of Computer Science 18 (6).

[12] Huang, D., Yan, C., Li, Q., Peng, X., 2024. From large language models to large multimodal models: A literature review. Applied Sciences 14 (12).

[13] Zhang, J., Huang, J., Jin, S., Lu, S., 2024. Vision-language models for vision tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (8), 5625–5644.

[14] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., et al., 2021. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020.

[15] Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., et al., 2024. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499.

[16] Karli, U.B., Chen, J.-T., Antony, V.N., Huang, C.-M., 2024. Alchemist: LLM-aided end-user development of robot applications. Association for Computing Machinery, New York.

[17] Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., et al., 2022. ProgPrompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302.

[18] Jin, Y., Li, D., A, Y., Shi, J., Hao, P., Sun, F., et al., 2023. RobotGPT: Robot manipulation learning from ChatGPT. arXiv preprint arXiv:2312.01421.

[19] Vemprala, S., Bonatti, R., Bucker, A., Kapoor, A., 2023. ChatGPT for robotics: Design principles and model abilities. arXiv preprint arXiv:2306.17582.

[20] Wake, N., Kanehira, A., Sasabuchi, K., Takamatsu, J., Ikeuchi, K., 2023. ChatGPT empowered long-step robot control in various environments: A case application. IEEE Access 11, 95060–95078.

[21] Wang, B., Zhang, J., Dong, S., Fang, I., Feng, C., 2024. VLM see, robot do: Human demo video to robot action plan via vision-language model. arXiv preprint arXiv:2410.08792.

[22] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., et al., 2022. Do as I can, not as I say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691.

[23] Lin, K., Agia, C., Migimatsu, T., Pavone, M., Bogh, J., 2023. Text2Motion: From natural language instructions to feasible plans. Autonomous Robots, 1345–1365.

[24] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., et al., 2022. Inner Monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608.

[25] Bhat, V., Kaypak, A.U., Krishnamurthy, P., Karri, R., Khorrami, F., 2024. Grounding LLMs for robot task planning using closed-loop state feedback. arXiv preprint arXiv:2402.08546.

[26] Luan, Z., Lai, Y., Huang, R., Bai, S., Zhang, Y., Zhang, H., et al., 2024. Enhancing robot task planning and execution through multi-layer large language models. Sensors 24 (5).

[27] Yang, Z., Ning, L., Wang, H., Jiang, T., Zhang, S., Cui, S., et al., 2024. Text2Reaction: Enabling reactive task planning using large language models. IEEE Robotics and Automation Letters 9 (5), 4003–4010.

[28] Wang, C., Hasler, S., Tanneberg, D., Ocker, F., Joublin, F., Ceravola, A., et al., 2024. LaMI: Large language models for multi-modal human-robot interaction. In: CHI Conference on Human Factors in Computing Systems Extended Abstracts. ACM, New York.

[29] Mei, A., Zhu, G.-N., Zhang, H., Gan, Z., 2024. ReplanVLM: Replanning robotic tasks with visual language models. IEEE Robotics and Automation Letters 9 (11), 10201–10208.

[30] Kannan, S.S., Venkatesh, V.L.N., Min, B.-C., 2024. SMART-LLM: Smart multi-agent robot task planning using large language models. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 12140–12147.

[31] Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., Fei-Fei, L., 2023. VoxPoser: Composable 3D value maps for robotic manipulation with language models. arXiv preprint arXiv:2307.05973.

[32] Xia, W., Wang, D., Pang, X., Wang, Z., Zhao, B., Hu, D., et al., 2024. Kinematic-aware prompting for generalizable articulated object manipulation with LLMs. arXiv preprint arXiv:2311.02847.

[33] Shentu, Y., Wu, P., Rajeswaran, A., Abbeel, P., 2024. From LLMs to actions: Latent codes as bridges in hierarchical robot control. arXiv preprint arXiv:2405.0479.

[34] Yang, J., Zhang, H., Li, F., Zou, X., Li, C., Gao, J., 2023. Set-of-Mark Prompting unleashes extraordinary visual grounding in GPT-4V. arXiv preprint arXiv:2310.11441.

[35] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., et al., 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818.

[36] Song, C.H., Wu, J., Washington, C., Sadler, B.M., Chao, W.-L., Su, Y., 2023. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. arXiv preprint arXiv:2212.04088.

[37] Sarch, G., Wu, Y., Tarr, M.J., Fragkiadaki, K., 2023. Open-ended instructable embodied agents with memory-augmented large language models. arXiv preprint arXiv:2310.15127.

[38] Zhou, Z., Song, J., Yao, K., Shu, Z., Ma, L., 2023. ISR-LLM: Iterative self-refined large language model for long-horizon sequential task planning. arXiv preprint arXiv:2308.13724.

[39] Sakib, M.S., Sun, Y., 2024. Consolidating trees of robotic plans generated using large language models to improve reliability. International Journal of Artificial Intelligence and Robotics Research 1 (1), 2450002.

[40] Zhang, Y., Wang, Z., Zhang, S., Peng, Y., Chen, M., 2023. Boosting robot intelligence in practice: Enhancing robot task planning with large language models. In: International Conference on Robotics and Automation Engineering (ICRAE), 90–94.

[41] Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., Stone, P., 2023. LLM+P: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477.

[42] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., et al., 2023. Qwen technical report. arXiv preprint arXiv:2309.16609.

[43] OpenAI, 2024. GPT-4o.

[44] Aryan, S., Subrahmanya, S., Jagtap, P., Mani, L., R., S.M., 2023. Image captioner: Captioning for visual impact. In: International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), 1–8.

[45] Zhao, C., Yuan, S., Jiang, C., Cai, J., Yu, H., Wang, M.Y., et al., 2023. ERRA: An embodied representation and reasoning architecture for long-horizon language-conditioned manipulation tasks. arXiv preprint arXiv:2304.02251.

[46] Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., Suenderhauf, N., 2023. SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning. arXiv preprint arXiv:2307.06135.

[47] Chen, L., Liu, Z., He, W., Li, Y., Luo, R., Yang, M., 2024. Long context is not long at all: A prospector of long-dependency data for large language models. arXiv preprint arXiv:2405.17915.

[48] Chen, Y., Pesaranghader, A., Sadhu, T., Yi, D.H., 2024. Can we rely on LLM agents to draft long-horizon plans? Let's take TravelPlanner as an example. arXiv preprint arXiv:2408.06318.

Biography

Dominik Koch (*1996) received his B.Sc. in Mechanical Engineering and his M.Sc. in Mechatronics and Information Technology from Karlsruhe Institute of Technology. He is currently a research associate at the wbk Institute of Production Science, pursuing his doctoral degree. His work focuses on robot-based optical inspection in circular manufacturing.

Jakob Wolber (*1999) is working as a Consultant at the Boston Consulting Group in Munich and is part of BCG's Operations practice. He holds a bachelor's and a master's degree in industrial engineering from the Karlsruhe Institute of Technology (KIT), with a focus on production engineering.

Zhou Shi (*1997) has been applied AI researcher at Siemens Ltd., China since 2022. Zhuo's work mainly focuses on researching and applying cutting edge AI technology in the automation area, including computer vision solution for pick & place and anomaly detection use cases, Large Language Model for domain knowledge retrieval, Vision Language Action model for dexterous robot control.

Ji Bo Cheng (*1995) has been researcher at Siemens Ltd., China since 2023. Bo Cheng's work mainly focuses on researching and transforming cutting edge technology into real scenarios, including computer vision solution for pick & place, whole body control of humanoids robot. His research interest consists of kinematic, dynamic of industry robot, generate AI and VLA.

Lucas Bretz (*1992) is General Manager of the Global Advanced Manufacturing Institute (GAMI) in Suzhou, China. He holds a PhD in Mechanical Engineering from the Karlsruhe Institute of Technology. His work focuses on operational excellence, digital transformation, and AI driven production systems. He develops robotics and automation applications for factory optimization and integrates large language models into secure industrial environments.

Alexander Geiser (*1994) received his M.Sc. in Mechatronics from HS Mannheim in 2022. Previously, he worked as a Technical Project Leader in special machine construction. Currently, he is a research associate at the wbk Institute of Production Science, pursuing his doctoral degree (Dr.-Ing.).

Felix Baer (*1992) has been the Head of the Autonomous Factory Department at Siemens Ltd., China since 2022. He has conducted research in advanced robotics for 14 years and holds several patents in this field. As a PhD candidate, he focuses on business transformation driven by generative AI products in industry.

Martin Benfer (*1994) is Deputy Head of Production Systems at wbk Institute of Production Science at Karlsruhe Institute of Technology. He leads the Production Systems Planning and Quality Assurance groups. His research focuses on data and model-based decision-making, digital twins, and resilience in production systems.

Florian Stamer (*1991) has been Professor of Production Management at Leuphana University Lüneburg since 2025. Before that, he led a junior research group and served as chief engineer in quality assurance at Karlsruhe Institute of Technology. His research links AI with production and quality management across the product life cycle. He focuses on digital production systems, production networks, and data-driven decision support for sustainable and efficient manufacturing

Gisela Lanza (*1970) is Professor of Production Systems and a member of the board of directors at the wbk Institute of Production Science at Karlsruhe Institute of Technology. She leads the Production Systems division, focusing on global production strategies, production system planning, and quality assurance. Her expertise centers on data-driven design and evaluation of production systems, ensuring robust and efficient manufacturing operations.