Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-023-02294-y

Embodied intelligence in manufacturing: leveraging large language models for autonomous industrial robotics

Haolin Fan[1] · Xuan Liu[1] · Jerry Ying Hsi Fuh[1] · Wen Feng Lu[1] · Bingbing Li[1,2]

Received: 1 October 2023 / Accepted: 23 November 2023

Abstract
This paper delves into the potential of Large Language Model (LLM) agents for industrial robotics, with an emphasis on autonomous design, decision-making, and task execution within manufacturing contexts. We propose a comprehensive framework that includes three core components: (1) matches manufacturing tasks with process parameters, emphasizing the challenges in LLM agents' understanding of human-imposed constraints; (2) autonomously designs tool paths, highlighting the LLM agents' proficiency in planar tasks and challenges in 3D spatial tasks; and (3) integrates embodied intelligence within industrial robotics simulations, showcasing the adaptability of LLM agents like GPT-4. Our experimental results underscore the distinctive performance of the GPT-4 agent, especially in Component 3, where it is outstanding in task planning and achieved a success rate of 81.88% across 10 samples in task completion. In conclusion, our study accentuates the transformative potential of LLM agents in industrial robotics and suggests specific avenues, such as visual semantic control and real-time feedback loops, for their enhancement.

Keywords Large language models (LLMs) agents · Industrial robotics · Autonomous design · Decision-making · Embodied intelligence

Introduction

The dawn of Industry 4.0 marks a transformative era in which industrial robotics becomes integral to automated manufacturing systems, promising improved productivity and flexibility. Despite these advances, the complexity in programming these robots remains a significant barrier, often requiring specialized knowledge and hindering the ease of modification and re-programming (Buerkle et al., 2023). This challenge is compounded in the face of changing demands from mass production to small-batch and customized production (Perzylo et al., 2016), necessitating robots with advanced sensing, learning, and interaction abilities (Goel & Gupta, 2020).

Prior initiatives aimed at improving robots' ability to interpret human commands and autonomously plan tasks have made some headway. Techniques like semantic parsing have been leveraged to transform human commands into structured task descriptions (Choi et al., 2021; Paul et al., 2018), and various task planning languages have been created to facilitate the automatic generation of action sequences (Hoebert et al., 2021). Despite advances, these approaches remain unsatisfactory for use in dynamic, unstructured manufacturing environments due to their weak generalization and adaptivity, and are further hampered by their reliance on large volumes of training data and significant computational resources, rendering them less viable for smaller-scale, agile manufacturing setups. These limitations underscore the

✉ Bingbing Li
bingbing.li@csun.edu

Haolin Fan
e0816265@u.nus.edu

Xuan Liu
xuanliu@u.nus.edu

Jerry Ying Hsi Fuh
jerry.fuh@nus.edu.sg

Wen Feng Lu
mpelwf@nus.edu.sg

[1] Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117575, Singapore

[2] Department of Manufacturing Systems Engineering and Management, California State University Northridge, Northridge, CA 91330, USA

Published online: 09 January 2024

urgent need for a more nuanced and responsive paradigm for instruction parsing and task execution, precisely where large language models (LLMs) can play a transformative role.

In this context, the advent of LLMs, epitomized by Generative Pretrained Transformer 4 (GPT-4), presents a groundbreaking opportunity (Wei et al., 2022; Zhao et al., 2023) to address these limitations and bridge the gap between human instructions and robotic execution, with capabilities such as multi-domain knowledge internalization (OpenAI, 2023; Petroni et al., 2019), natural language decomposition using the Chain of Thought (CoT) approach (Wei et al., 2023), and advanced code-generation ability (Austin et al., 2021; Poesia et al., 2022). Furthermore, the interactive skills of LLM allow iterative refinement of tasks through user feedback (Bang et al., 2023; Chen & Chang, 2023), while their impressive zero-shot and few-shot learning capabilities enable adaptation to a range of tasks with minimal input even without training.

Building upon the identified limitations and outstanding abilities of LLMs, the concept of "embodied intelligence" (Yoneda et al., 2023) in robotics heralds a pivotal shift towards truly autonomous systems. Here, LLMs are not merely language tools but artificial intelligence (AI) agents capable of interpreting, learning from, and adapting to their environment. This capability becomes increasingly feasible through advancements in LLMs, and has been evidenced in recent research, demonstrating the potential of LLMs to autonomously navigate complex operational tasks, from design to decision-making and path planning (Driess et al., 2023; Huang et al., 2023b; Neunzig et al., 2023; Ren et al., 2023; Singh et al., 2023). Our work aims to further this trajectory, leveraging "embodied intelligence" into industrial robotics, addressing the adaptability and interaction challenges in autonomous personalized manufacturing.

The synthesis of robotics with LLMs has shown that robots can generate trajectories and perform manipulations directly from natural language instructions (Huang et al., 2023c; Liu et al., 2023; Yang et al., 2023). These capabilities represent a significant stride in robotic autonomy and flexibility, aligning with the dynamic requirements of personalized manufacturing. However, despite the burgeoning research on routine robot operations, the intricate integration of "embodied intelligence" within the domain of industrial manufacturing remains relatively unexplored. Industrial manufacturing is governed by distinct industrial constraints and standards that present unique challenges. For example, parsing the numerous parameters and task-specified configurations in manufacturing processes is demanding. Generating an intricate toolpath for 3D printing without access to complete 3D models is challenging. Our work seeks to address these challenges and difficulties by exploring a structured framework that harnesses LLMs to empower industrial robotics, thus contributing a novel perspective to the field of autonomous manufacturing.

This paper presents an LLM-centric framework for industrial robotics control that addresses key manufacturing tasks: task formulation, resource selection, environment initialization, path planning, and manipulation. This framework introduces the concept of LLM-powered agents, capable of parsing human instructions, planning, and executing tasks, and engaging in goal-oriented interactions with humans and their environment. We demonstrate the practicality of our approach through simulations in diverse manufacturing scenarios. The components of our framework are as follows:

1. *Component 1* Match tasks and process parameters—Here, LLMs are employed to extract process parameters from natural language descriptions or commands within a manufacturing context.
2. *Component 2* Path design—This involves generating motion paths for end-effectors based on predefined conditions and subsequently evaluating and verifying the generated path's efficacy.
3. *Component 3* Preliminary embodied intelligence for manufacturing—At this advanced level, the focus is on autonomous design, decision making, and execution of manufacturing tasks. This is achieved by leveraging resources from the code library and benchmark tasks in the task library.

Our contributions can be summarized as follows:

- We present a novel hierarchical and LLM-centric framework for industrial robotics control. This framework is designed to mitigate the prevalent challenges in adaptability, human instruction parsing, and the demands of small-batch and customized production. Using the advanced language processing and in-context learning capabilities of LLMs, our framework efficiently interprets natural language instructions and translates them into precise robotic actions without extensive training. This integration marks a significant step towards intelligent and flexible robotic systems suitable for a range of manufacturing contexts.
- Our work pioneers the practical integration of LLM agents with a focus on embodied intelligence within the domain of manufacturing, an area that has not been extensively explored. We demonstrate the capacity of these agents to understand complex commands and autonomously navigate the intricacies of manufacturing tasks, from design to decision-making and execution. This marks a significant advancement in realizing truly autonomous and intelligent robotic systems that can interact and adapt within their operational environment.

- Through rigorous experimentation, we provide a comprehensive evaluation of different LLM agents. This evaluation highlights their strengths, weaknesses, and dependencies on external and internal sources. Our findings offer valuable insights for future research and practical applications of LLM agents in the industrial domain.

In summary, our research contributes to the advancement of intelligent and adaptable industrial robots by integrating LLMs into robotics. We present a unique framework that bridges natural language processing (NLP) and robotics control, showcasing the potential of LLMs to improve the intelligence and flexibility of manufacturing operations.

The rest of the paper is structured as follows: section "Related works" delves into existing work that intersects LLMs with robotic control. Section "Methodology" offers a detailed exposition of our methodology, emphasizing its three core components. Section "Experiments" presents our experiments and results, providing a nuanced understanding of LLM agents' performance across tasks. In section "Discussion", we discuss our findings, drawing attention to the challenges and potential future directions. We conclude by summarizing our contributions and reflecting on the broader implications of combining industrial robotics control with LLMs-oriented agents.

Related works

The evolution of robotics and AI has been marked by the convergence of diverse research domains, each contributing significantly to the advancement of human–robot interactions. At the heart of this evolution is NLP, which has revolutionized the way we communicate with robotic systems. LLMs like GPT-4 have particularly transformed our approach, extending their utility beyond basic text processing. This section provides an overview of the journey from traditional robotics programming and task planning to the innovative applications of LLMs in robotics and manufacturing, highlighting both advancements and limitations.

Advancements in robotics programming and task planning

Robotics programming, especially in industrial contexts, has transitioned from rigid coding to more adaptable, human-centric methods. This shift, primarily driven by the need to move from mass production to small batch production, involves two key components: human instruction parsing and high-level robotic task planning.

Human instructions parsing Early methods in robotic programming focused on interpreting human instructions using graphical models (Kollar et al., 2010; Tellex et al., 2011), grammar-based interpreters (Wächter et al., 2018), and semantic parsing techniques (Kollar et al., 2014; Thomason et al., 2015), to extract high-level tasks or attributes. This method expands the gap between human instruction and robotic execution. The advent of advanced language models such as BERT and GPT-2 brought a nuanced understanding and execution of human instructions (Choi et al., 2021). Recent trends tend to end-to-end methodologies, where robots are trained using natural language through model learning (Nair et al., 2022), imitation learning (Brohan et al., 2023; Jang et al., 2022), and reinforcement learning (Jiang et al., 2019; Luketina et al., 2019; Misra et al., 2017). However, these methods often require extensive training data and computational resources and can struggle with complex or ambiguous instructions.

High-level task planning The development of sophisticated algorithms has significantly advanced task planning in dynamic environments. Techniques like translating language instructions into 2D cost maps enable the generation of a trajectory with enhanced safety and efficiency (Sharma et al., 2022). Meanwhile, the Planning Domain Definition Language (PDDL) has become a standard for autonomous task planning problems (Heuss et al., 2023), supported by frameworks based on fine-tuning (Capitanelli & Mastrogiovanni, 2023) and knowledge integration (Hoebert et al., 2021). These systems maintain an internal knowledge base for task planning and action sequence formulation based on defined problems and domains. However, these systems, while robust in various applications like mobile manipulators (Bezrucav & Corves, 2022) and robotic assembly (Rovida et al., 2017), sometimes struggle to adapt to rapidly changing or unforeseen scenarios, indicating the need for more adaptable algorithms.

Reasoning in LLMs

In the realm of LLMs, GPT-4 currently stands as the most advanced, surpassing its predecessors and competitors in a multitude of tasks, including coding, reasoning, mathematics, and context understanding (OpenAI, 2023). This model, along with other notable models such as GPT-3.5 (Ye et al., 2023), PaLM 2 (Anil et al., 2023), Claude 2 (Anthropic, 2023), and the LLaMA family models (Touvron et al., 2023), including LLaMA 2 and Guanaco (Dettmers et al., 2023), have been selected for our research due to their exceptional performance in language-related tasks.

With the emergence of LLMs, researchers are beginning to explore their capabilities beyond traditional text processing. LLMs showcase proficiency in tasks with few-shot or

zero-shot learning, which supports high generalization ability with proper instructions (Peng et al., 2023) and enables these pre-trained models to achieve competitive results near the state-of-the-art in various tasks without fine-tuning or retraining (Brown et al., 2020). To further exploit their reasoning prowess, instruction methods such as step-by-step thinking (Kojima et al., 2023) and chain-of-thought (Wei et al., 2023) have been developed. Observations from (Wei et al., 2022) highlight that LLMs, when adequately scaled, significantly outperform previous models in tasks spanning analogical reasoning to instruction following. This led to their application in diverse areas such as code generation (Chen et al., 2021; Drori et al., 2022), and action planning (Huang et al., 2022). Our current exploration is a natural progression from these studies, focusing on the untapped potential of LLMs in manufacturing and industrial robotics.

Leveraging the abilities of LLMs

The demonstrated capabilities of LLMs have led the robotics community to begin to envision their application without the need for additional model training. Most of the work focuses on using LLMs to simplify robotic tasks or reduce the barrier for an untrained person to use robots. The semantic parsing ability of LLMs allows robots to be controlled via natural language instruction. LLMs like GPT-4 and fine-tuned GPT-2 are used to translate human instructions into robot goals or plans (Choi et al., 2021; Huang et al., 2023c), while GPT-3 is employed to extract landmark sequences to help the robot understand the user-specified routine (Shah et al., 2023). Generating code with LLMs changes the paradigm of robot programming. Recent studies focus on calling predefined APIs, functions, or primitives by LLM to generate programs for task planning and robot control (Liang et al., 2023; Singh et al., 2023), but the forgetfulness of LLMs limits the effectiveness of these approaches (Chen & Huang, 2023). Other research explores the employment of LLMs in reward design and reinforcement learning (Colas et al., 2020; Kwon et al., 2023; Mu et al., 2022).

Numerous studies employ LLMs for robot trajectory planning in the actual environment (Ahn et al., 2022; Huang et al., 2023d; Raman et al., 2023). To further refine robot interactions, some research provides textual environment descriptions (Huang et al., 2023e; Singh et al., 2023), while others integrate Visual Language Models (VLMs) with LLMs (Huang et al., 2023a; Shah et al., 2023; Tombari et al., 2022). A notable advancement is by Huang et al. (2023c), who combined LLMs and VLMs to map language instructions into 3D value maps to guide robot trajectories. Another impressive work by Liang et al. (2023) uses LLMs to invoke parameterized control primitive APIs to control robot movement, where primitives are designed manually, and LLMs have been verified to possess the ability to compose sequential policies. Our research diverges from these by using the reasoning, context understanding, and code generation abilities of LLMs for robotic tasks in manufacturing.

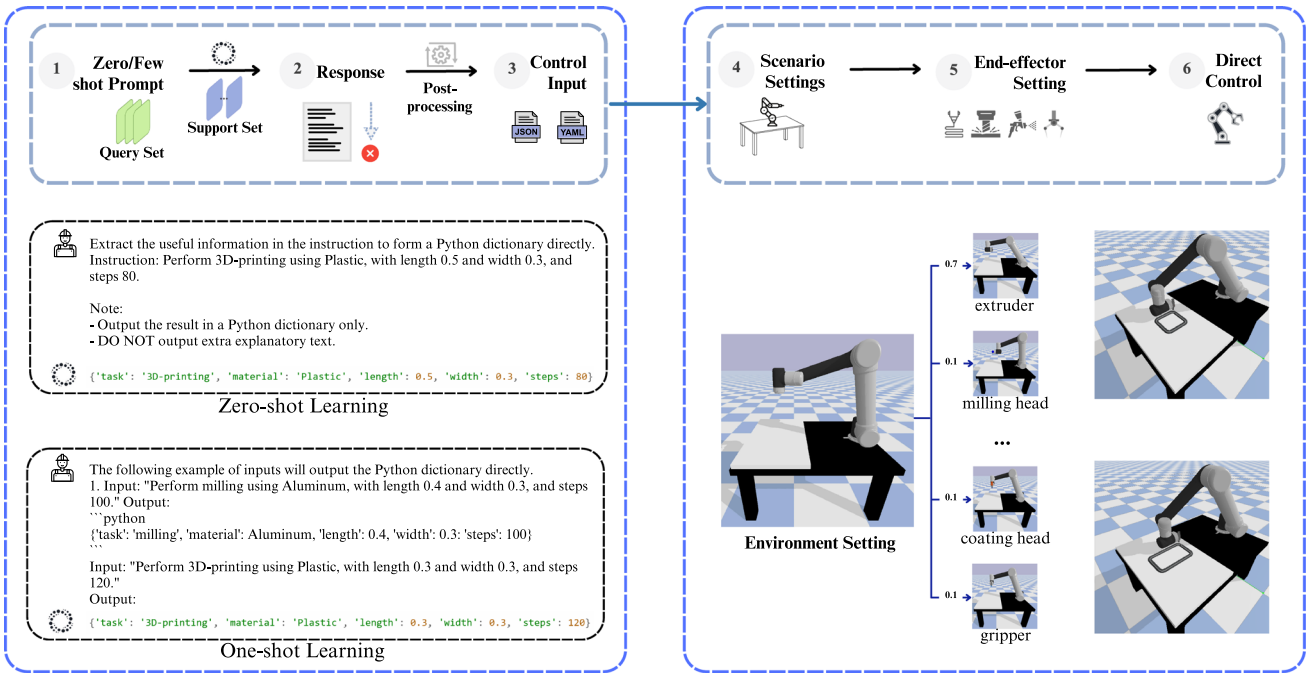
In the context of industrial robotics, our research takes a novel approach by integrating the advanced capabilities of LLMs. Unlike general-purpose robots, industrial robots typically feature fixed-base manipulators and operate within confined spaces. Although they are known for their precision and speed, their lack of flexibility presents a challenge in adapting to new tasks without extensive reprogramming (Hägele et al., 2016). Here, the potential of LLMs becomes particularly valuable. Through this LLM-centric framework, our aim is to bridge the gap between the advanced capabilities of LLMs and the practical requirements of industrial robotics. This synergy promises to unlock new possibilities in manufacturing, enabling robots to perform more complex, varied, and customized tasks with minimal human intervention.

Methodology

Our methodology comprises three primary components: (a) Matching manufacturing tasks with process parameters; (b) Designing tool paths to achieve the desired geometry; and (c) Initial implementation of embodied intelligence. These components exhibit a hierarchical relationship, with each subsequent task building upon the previous.

To better understand the intricacies of task planning and execution in the context of industrial robotics, we introduce a tuple, represented as $\langle \mathcal{O}, \mathcal{P}, \mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{G}, \mathbf{t} \rangle$. This tuple offers a structured approach to planning and executing tasks in industrial robotics. Specifically, $\mathcal{O}$ represents the available assets. $\mathcal{P}$ characterizes the properties of these assets and the process parameters of specific manufacturing operations. The action space $\mathcal{A}$ consists of discrete high-level actions $a \in \mathcal{A}$ that dictate the type of interaction the robot will have with its environment and realize the goal of the task. These actions vary depending on the current state of the assets, with $s \in \mathcal{S}$ denoting the current state and $\mathcal{S}$ the state space. $\mathcal{T}\left(s' \mid s, a\right)$ is the deterministic transition model, symbolizing the state changes for the assets. $\mathcal{I}$ denotes the initial state in which all environmental settings are in place. Meanwhile, $\mathcal{G}$ contains the desired states or sequences of operations. The LLM agents, while not having explicit knowledge of the exact goal state $\mathrm{g} \in \mathcal{G}$, operate based on high-level tasks described in natural language $\mathbf{t}$.

In our simulations, $\mathcal{O}$ mainly consists of assets described in the Unified Robot Description Format (URDF), which is an XML specification designed for multi-body systems. Assets within $\mathcal{O}$ encompass typical work environment equipment and specialized manufacturing end-effectors like extruders, milling heads, and various types of grippers. In addition,



(a) Phase I: Instruction Parsing **(b) Phase II: Command Execution**

Fig. 1 Flowchart of Component 1: Phase I involves extracting information from natural language prompts. In Phase II, the emphasis shifts to task execution and parameter adjustment; values such as 0.7 indicate the probability of selecting the end-effector

the action space is $\mathcal{A} = \{$`select tool, couple, decouple, activate, deactivate, rotate, …, move, 3D print, mill, coat, grip`$\}$.

The initial state $\mathcal{I}$, represents the state where the robot is positioned at the workspace's origin, with all its joints at default angles, and the end-effector is decoupled. This standardized initial state ensures consistency across different runs and experiments.

The goal states in $\mathcal{G}$ are operation-specific; for example, a goal state of 3D printing includes an extruder selected as the end-effector, and 3D printing is performed to form the desired shape. A specific tool path ψ will be followed during the desired shape generation process. These states are critical benchmarks for evaluating the robot's performance.

Lastly, the task directive $\mathbf{t}$ provides high-level instructions in natural language, such as "3D print a cube" or "mill grooves on the cube's surface." This approach facilitates intuitive interactions, allowing operators without extensive technical knowledge to instruct the robot effectively.

Task and process parameter matching

This component focuses on matching tasks and process parameters. It consists of two phases: instruction parsing and command execution. In Phase I, we leverage LLM agents' zero-shot and few-shot learning abilities to extract specific manufacturing tasks $\mathbf{t}$ (e.g., 3D printing, milling, gripping) and their associated process parameters (e.g., material type, particle size, milling depth) from natural language descriptions. These extracted parameters form a set of properties, denoted as $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. The detailed flow is illustrated in Fig. 1.

In Phase II, LLM agents set the scene layout. Based on the identified manufacturing tasks, they then select the most suitable end-effector by performing $a =$ `select_tool` from the set of end effectors that includes options such as extruders, milling heads, coating heads, and various grippers. This will transit state from $s =$ `environment_set` to $s' =$ `tool_selected`. Here, as modeled in Eq. 1, the agent selects the end-effector with the highest probability at Step 5. This is conceptually similar to the sequential token prediction in LLM.

$$e^* = \underset{e \in \mathcal{O}_e}{\text{argmax}}\ P(e \mid g) \tag{1}$$

Here, e^* is the most probable end-effector for the goal state $g \in \mathcal{G}$. $P(e \mid g)$ is the probability of selecting a specific end-effector e from the set of available end-effectors $\mathcal{O}_e \in \mathcal{O}$, given the goal state g. For instance, in Fig. 1b, given the t as '3D print a box', a goal state g could contain the conditions `select_tool`(extruder) $\in$ g, the possibilities of selecting various end-effectors are [0.7, 0.1, 0.1, 0.1], indicating the agent's likelihood of choosing each end-effector, with the first end-effector (the extruder for 3D printing) being

the most probable with a selection likelihood of 0.7. Guided by the CoT prompt strategy, the agents then execute the tasks using the identified process parameters. A subsequent post-processing step refines the agent's response, eliminating descriptive words, and producing outputs in suitable formats.

Autonomous tool path design

Building on Component 1, LLM agents organize the manufacturing scene, select the end-effector, and complete the manufacturing tasks as instructed. This component emphasizes the autonomous generation of the end-effector's path after a certain action $a \in \mathcal{A}$ is selected, denoted as ψ_e. Once the end-effector is coupled with the robot, given a starting state z_{start} and a target state z_{target} for the end-effector, the path ψ_e can be represented as a sequence of kinematic states comprising the positions and orientations of the end-effector, as described by Eq. 2:

$$\psi_e = \left\{z_{\text{start}}, z_1, z_2, \ldots, z_n, z_{\text{target}}\right\} \tag{2}$$

Component 2 engages in continuous decision-making to construct ψ_e. The interpolation functions, denoted by $I(x)$, where x is a parameter, allow the generation of a smooth and continuous path between the defined states. Notably, the velocity of the end-effector is determined by the interpolation frequency. The specific methods, including $I_{\text{linear}}(x)$, $I_{\text{spline}}(x)$, and $I_{\text{circular}}(x)$, furnish the system with a repertoire of tools to generate the continuous sequence of positions and orientations required along ψ_e, as depicted in Fig. 2.

In greater detail, given z_{start} and z_{target}, LLM agents determine the optimal combination of interpolation methods to formulate ψ_e, ensuring that the end-effector produces a part that aligns with design specifications and spatial constraints.

We integrate basic APIs, including interpolation methods, as input prompts to guide agents and as agent memory. Guided by CoT, LLM agents understand the target shape, analyze available APIs, and generate the target tool paths. We categorize tasks as either planar (e.g., generating words and alphabets, creating geometric shapes) or spatial (e.g., constructing a cube, cylinder, or sphere).

Following our initial experiments, we have introduced an error module and a reflection mechanism in addition to the standard API and agent memory modules, as shown in Fig. 3a. The error module is designed to identify common input errors. For example, if the path ψ_e generated using $I_{\text{linear}}(x)$ exceeds the robot workspace, the error module will flag this error. It also addresses issues related to setting layer height in processes like 3D printing, as well as other common errors such as only considering the positions of the path points set without considering the orientations. The reflection module enables LLM agents to perform self-validation

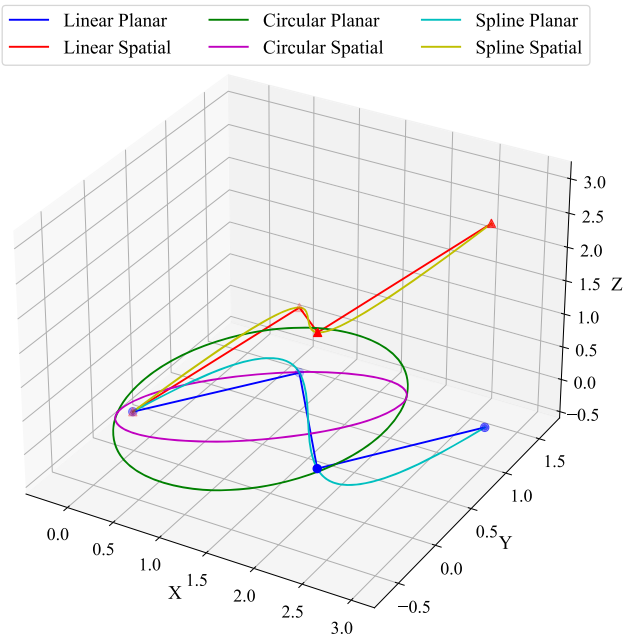


Fig. 2 Interpolation methods employed for tool path generation

of generated paths to refine the path and avoid possible errors, though it might occasionally alter correct paths.

To better align with industrial practices, we also develop a G-code parsing class for indirectly generating ψ_e. G-code is a language that instructs machines on specific tasks, such as movements and operations. To further support this, we provide example tasks that guide LLM agents in generating G-codes autonomously for the desired shapes. Examples of G-code generation can be seen in Fig. 3c.

Initial realization of embodied intelligence

Highlighting the potential of LLM agents in the design and control of industrial robotics, we present a framework, demonstrated in Fig. 4, that enables industrial robots to design, decide and execute tasks autonomously within manufacturing contexts. This framework transforms task-program synthesis into an agent-prompting mechanism, integrating an agent with task design and execution capabilities seamlessly. Each component of this framework leverages few-shot and CoT prompts on LLMs, granting it the capability to perform reasoning and exploration.

We represent this framework as $F(m)$, where m denotes various modules: task description and design m_{t}, API reference m_{API}, common error review m_{e}, and code generation m_{c}. Each module is activated by its corresponding prompt within the set $\mathbb{P} = \{p_{\text{t}}, p_{\text{API}}, p_{\text{e}}, p_{\text{c}}\}$.

Modules such as m_{t} and m_{c} have dedicated memory buffers to utilize the in-context memory of the LLMs. LLM agents handle aspects that traditionally require human inter-

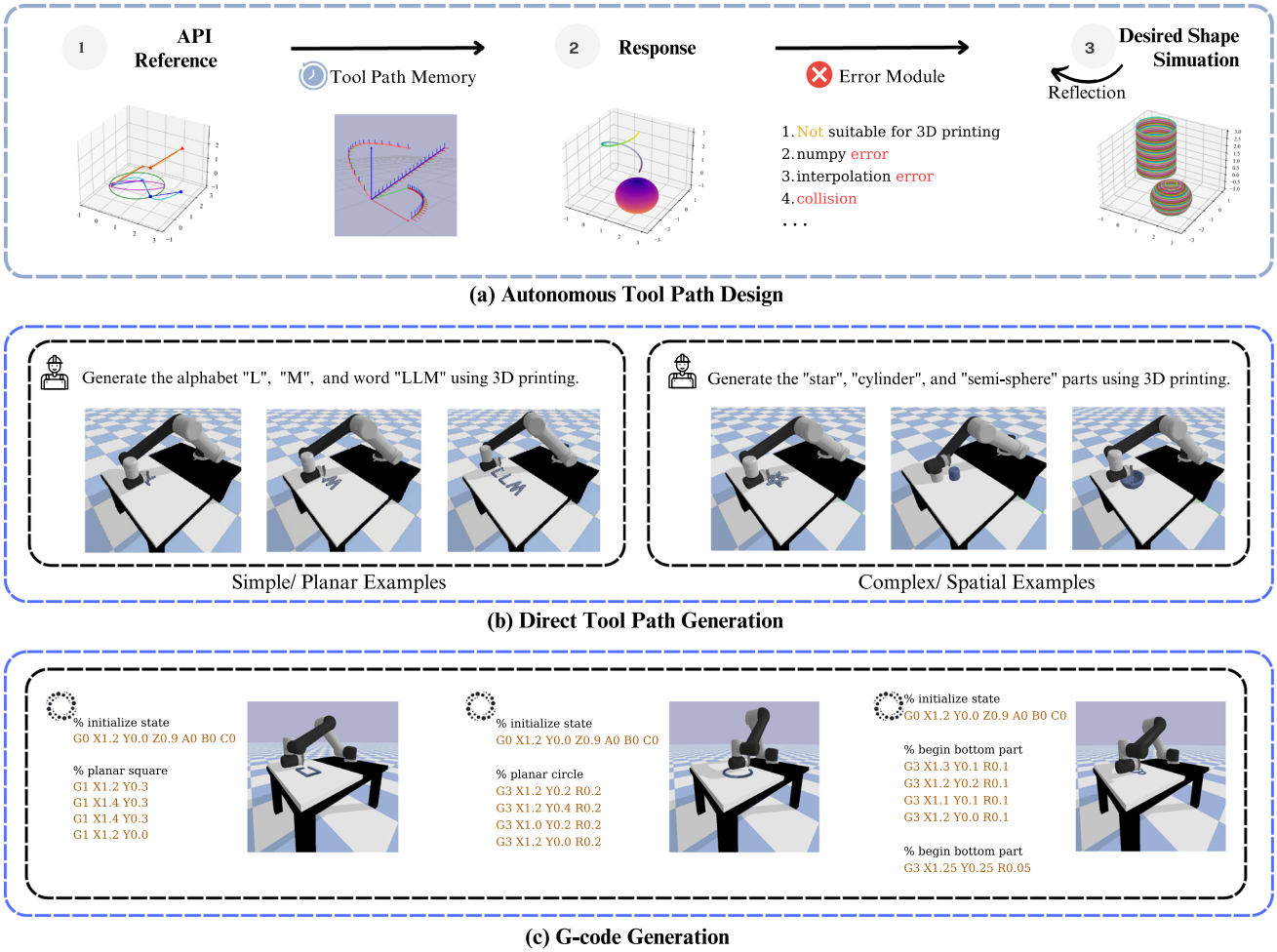


Fig. 3 **a** Flowchart illustrating the tool path generation process for Component 2, highlighting our error and self-reflection modules. This includes an example demonstrating the error module's correction of incorrect path generation for cuboids and spheres. **b** Examples of both planar and spatial cases. **c** G-code generation samples for shapes: square, circle, and gourd

vention, such as path planning. In parallel, the execution module operates on the code generated by LLMs to accomplish specific tasks. To activate LLM agents, we employ a dual approach: top-down and bottom-up prompt strategies. The top-down strategy breaks down complex tasks into simpler sub-tasks, while the bottom-up strategy refines prompts by incrementally adding details and constraints. This dual approach allows the development and refinement of prompt strategies based on intricate manufacturing details.

The task generation and design module is denoted as m_{t}, processes task example inputs, and guides the LLM to produce novel, non-redundant tasks related to manufacturing. The agents' task generation can be represented as a function mapping from a prompt to a specific task: $F(m_{\mathrm{t}}) : p_t \rightarrow t$, where $p_t \in \mathbb{P}$. The input prompts for agent task generation, p_t, are multifaceted, encompassing task names, descriptions, goal states, and required assets. The generated tasks are subsequently stored in the task memory buffer in a structured format for future reference, using the format $\{t_g, \mathrm{g}_g, \mathcal{O}_g\}$.

The task memory buffer also houses various assets from $\mathcal{O}$, reference examples from the task library, existing task descriptions, and suboptimal task examples with identified drawbacks. This arrangement ensures that the generated tasks are unique and adhere to specific constraints, such as the resource options and operational boundaries of the robot.

Once task t is generated, API prompts p_{API} give the agent access to m_{API}, which offers foundational knowledge on the main classes and functions of manufacturing-related tasks, covering actions in $\mathcal{A}$, and basics of physical simulation and rendering. To avoid historical errors, we include p_{e} to guide the agent into m_{e} that summarizes common high-level mistakes.

In the final stage, the task code formulated by $F(m_{\mathrm{c}})$ is stored in the memory buffer, presenting the generated task name to the LLM agents. Through the selection of refer-

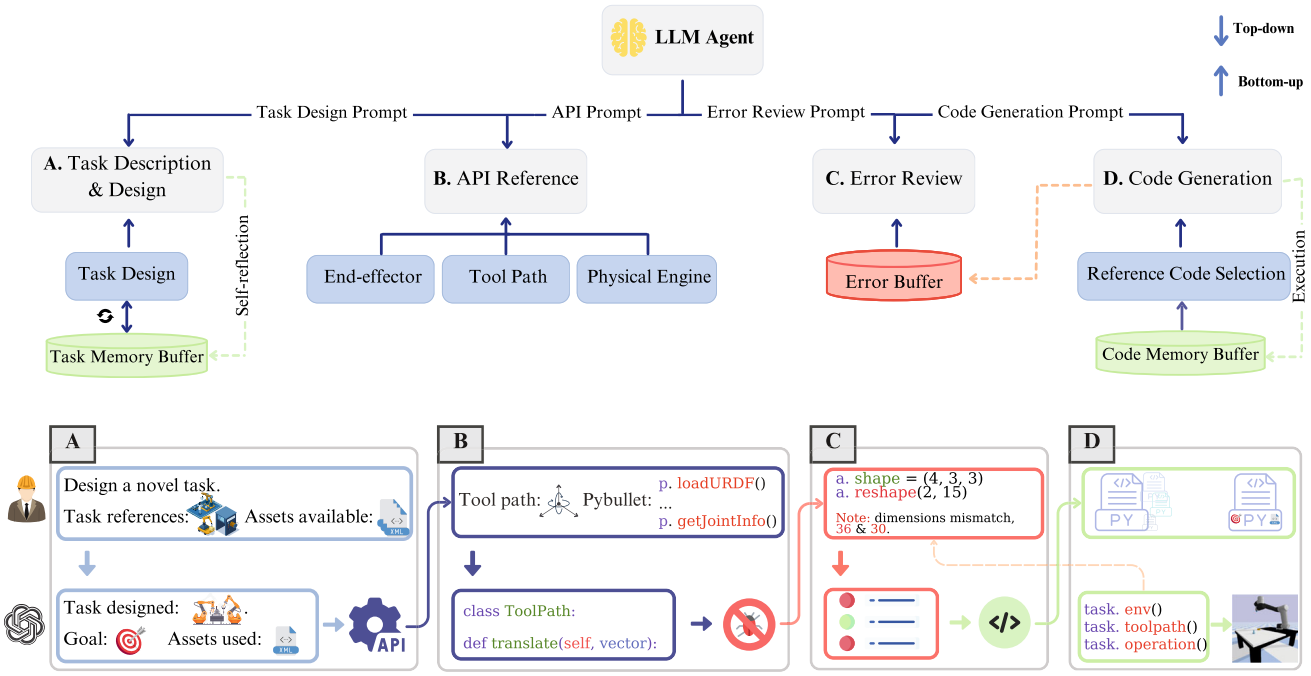


Fig. 4 Flowchart of Component 3, showcasing our **A** task description and design module, **B** API reference module, **C** error reviewer module, **D** code generation module, and essential components like the memory buffer and error buffer

ence codes, LLM agents identify helpful codes for realizing the current task. This ensures that LLM agents comprehend task descriptions accurately and implement tasks logically from experience, considering factors like scene construction sequence and goal setting for planes or spaces.

To better illustrate this process, one use case is depicted in Algorithm 1. Here, we assume that the task generated by the LLM agent involves 3D printing with one robot and milling with another, executed sequentially and cooperatively.

Experiments

To bridge our simulation setup and evaluation approach, we designed experiments that test the robot's performance in various tasks. Regarding our experimental setup, we used Pybullet (Coumans & Bai, 2021) as our simulation platform. We performed these experiments in different components using diverse example samples. To enhance the robustness of our results, we employed parallel input prompts and maintained consistency in hyperparameters, such as temperature and the top k value, across all samples. In addition to the deployment of LLaMa2 family models on a server equipped with eight NVIDIA A100 GPUs with 80 GB of memory, the rest of LLM agents were activated via API.

Task and process parameters matching

To compare the performance of different LLM agents in terms of instruction comprehension and parsing, we conducted 20 experiments using consistent prompts for each group of the desired shape. During this phase, the output from LLM agents was not subject to any post-processing, presenting their capabilities in raw form. After completing both Phase I and Phase II, we evaluated the overall success rate of the tasks by calculating the ratio of samples that completed both phases to the total number of tasks.

Figure 5a presents the results for Component 1. During Phase I, the PaML-2 agent exhibits a lower success rate, primarily due to its inclination to provide detailed reasoning during response generation. Adjusting the prompt struggles to mitigate this descriptive output. In contrast, the Claude-2 agent excels, achieving a success rate of 100% even without post-processing. This success might be attributed to the optimization of the Claude-2 model using CoT during its training phase. Notably, the LLaMa2-13B agent only attains an approximate 80% success rate after post-processing. While the average success rate for most LLM agents during Phase II hovers around 85%, the cumulative rate is a mere 10%. This highlights the cascading nature of the tasks: a lower success rate in Phase I has a cascading effect on overall performance. Despite this challenge, decomposing tasks using CoT has proven to be more effective than direct prompts. We observed

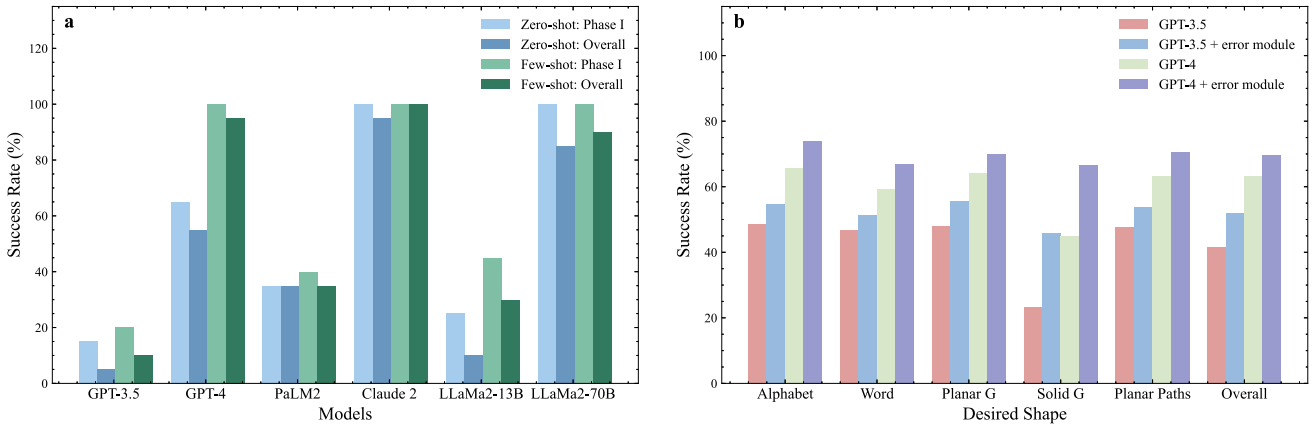


Fig. 5 **a** LLM agents' success rate for Component 1. **b** The success rate of the GPT−3.5 and GPT-4 agents for Component 2, both with and without the error module

Algorithm 1 LLM agent's decision process for generated task: 3D printing and milling using two robots

Task Generation:
Generate task $t \leftarrow F(m_t(p_t))$
API Reference:
Access m_{API} using p_{API}
Initialization:
Identify goal state g from task description
Set initial state $s_0 \leftarrow \mathcal{I}$
Planning:
Select robots r_1, r_2
Select end-effectors e_1, e_2 for 3D printing and milling
Plan high-level actions $a_1, a_2, ..., a_n \in \mathcal{A}$
Interpolate points for tool paths ψ_{e1}, ψ_{e2} using $I(x)$
Error Review:
Access m_e using p_e
Execution:
$a_1 =$ `couple(`r_1, e_1`)` ▷ Couple r_1, e_1
$s_1 \leftarrow \mathcal{T}(s_0, a_1)$
for each point z_i in ψ_{e1} **do**
    $a_2 =$ `3d_print(`z_i`)`
end for
$s_2 \leftarrow \mathcal{T}(s_1, a_2)$ ▷ State update after full print
if $s_2 =$ `print_done` **then**
    $a_3 =$ `decouple(`r_1, e_1`)` ▷ Decouple r_1, e_1
end if
// Milling steps (coupling, milling, decoupling)
Goal Checking:
if $s_n \notin \mathcal{G}$ **then**
    Return to planning step
end if
Code Generation:
Identify helpful codes for references
Generate code for task using $F(m_c(p_c)) \rightarrow$ `code`
Store the generated code in the memory buffer

an average improvement of 43.2% with a simple CoT prompt structure that encouraged a step-by-step approach.

Autonomous tool path design

For the second component, we employed both the GPT-3.5 and GPT-4 agents, with each task executed 20 times. The planar tasks covered 26 alphabets, 15 letter combinations, and 6 planar geometries. In contrast, the 10 spatial tasks incorporated shapes such as hollow and solid cubes, cylinders, and semi-spheres. Subsequently, we allowed the agents to execute the generated tool paths and manually assessed the success rate in achieving the desired shapes.

Our results are shown in Fig. 5b, indicating that LLM agents are proficient in generating regular paths. Without the error module, the GPT-4 agent achieved a success rate of 65.77% for alphabet generation, 59.41% for letter combinations, 64.29% for other planar geometric shapes, and 45% for three-dimensional shapes. These rates represent the highest average success across all categories. However, certain alphabets and shapes posed challenges. For instance, paths for alphabets with multiple key points, such as the letter M, or those lacking symmetry, such as the letter G, were error-prone. Occasionally, the agent might choose an unsuitable margin, causing overlapping characters or exceeding the executable space.

While LLM agents excel in planar shape generation, their performance in spatial shape generation is less impressive. The GPT-3.5 and GPT-4 agents only achieved average success rates of 23.33% and 45%, respectively. The primary issues are: (a) LLM agents frequently misinterpreted the spatial relationships between points during path generation. An example of this is when attempting to create a semi-sphere for 3D printing; the output paths began from the top with the largest radius. (b) The agents tended to perceive 3D objects with a bias towards planar interpretations, resulting in hollow structures. A case in point is where the objective is a solid cylinder, yet the generated paths only represent the bottom and top surfaces, accompanied by two parallel lines. (c)

Table 1 The success rate for G-code generation of the desired shape (standard deviation in parentheses)

Agent	Alphabet	Word	Planar	Overall
GPT-3.5	39.2%	41.2%	42.9%	41.1% (±21.6%)
GPT-4	56.2%	45.9%	45.7%	49.3% (±23.1%)

For 3D printing, the agents often overlooked the significance of layer heights to the extruder's attributes. This oversight can lead to neglecting the influence of path layer height on the object's formability. These challenges reflect the current limitations of LLM agents in navigating 3D spaces.

To address these issues, we incorporated the error module discussed in section "Autonomous tool path design". This addition enhanced the agents' performance in both planar and 3D geometries. For planar tasks, the success rate of the GPT-3.5 agent increased from 47.74 to 53.93%, while the GPT-4 agent rate increased from 63.16 to 70.56%. Among agents, the GPT-4 agent combined with the error module achieved the best overall performance, with an average success rate of 69.59%. This was a notable improvement over the GPT-3.5 agent's 67.12%. Integration of the error module also improved the stability of the LLM agent's responses, as evidenced by the reduction in the standard deviation for the GPT-4 agent from 25 to 18%.

For the G-code generation task, we conducted five experiments for each desired shape, only evaluating the tasks executed without the error module. The results are presented in Table 1.

The results of G-code generation show a decline in success rate compared to direct tool path generation. Specifically, the GPT-3.5 agent's overall success rate dropped from 41.6 to 41.1%, while the GPT-4 agent's rate decreased from 63.2 to 49.3%. This decline is reasonable given the complexity of G-code generation and the need for extensive memory to enhance performance.

Initial realization of embodied intelligence

Task design

Our experiments prompted five agents—GPT-3.5, GPT-4, Claude-2, LLaMa2-70b, and the fine-tuned Guanaco-33b—to generate 32 tasks each, resulting in a total of 160 diverse tasks. These tasks typically entailed multi-step manufacturing procedures, such as the intricate process of coating three cubes and stacking them at the desired location.

We initially evaluated these tasks using an automated method, focusing on novelty level (NLev), realization possibility (RPos), collision-free possibility (CFP), and relevance to the manufacturing context. For example, if a task involves various objects or involves multiple robots cooperating, it is possible to obtain a higher NLev score and lower RPos and CFP scores.

Table 2 Average scores achieved by LLM agents in terms of novelty, realization, collision-free, and relevance to manufacturing

Agent	NLev	RPos	CFP	Relevance	N
Guanaco-33b	4.22	**8.17**	5.13	3.83	1
Claude-2	7.02	6.88	**5.22**	5.26	5
LLaMa2-70b	5.91	7.01	4.99	5.10	2
GPT-3.5	5.75	7.12	5.16	4.91	2
GPT-4	**7.19**	7.25	5.03	**6.44**	6

Bold values represent the highest average score achieved under each scoring criterion

For an objective assessment, GPT-4 was tasked with scoring each criterion on a score of zero to ten and providing qualitative explanations to ensure transparency. To address the potential bias due to GPT-4 being also a subject of the study, independent human evaluators reviewed tasks to confirm or revise GPT-4's scores.

Table 2 presents the average scores of the tasks generated and the distribution of selected task candidates (N) generated by each agent. The results showed the proficiency and bias of each model in all the criteria evaluated. In greater detail, we chose 16 task candidates for detailed analysis using a balanced approach, selecting tasks that scored above the median in at least two criteria and exhibited a reasonable score distribution across all criteria.

Notably, while the GPT-4 agent excelled in novelty and relevance, it produced more complex tasks, which pose greater implementation challenges due to increased spatial collision risks and moderate realization scores. On the contrary, the GPT-3.5 agent demonstrates average performance in all metrics. These results informed our selection of 16 tasks, emphasizing the need to challenge the models with tasks that align with real-world manufacturing scenarios.

High-level task planning

After the self-guided task building and selection, our study compared the abstract task planning abilities of two systems: the Teriyaki framework, which uses a planning description language (PDDL) as detailed by (Capitanelli & Mastrogiovanni, 2023), and LLM agents. We established a training dataset of 3000 entries for Teriyaki, constructed from problem-planning pairs generated by GPT-3.5. These pairs were specifically prompted to cover our extensive action space. To ensure a fair comparison, we matched the experimental conditions of the referenced study by keeping all hyperparameters the same. We evaluated both systems using two metrics: planning accuracy, which checks if the plans are correct and achieve the goal, and planning time, where shorter times indicate better efficiency.

Table 3 Performance comparison for accuracy and average task planning time between previous work and LLM agents

Solver	Acc. (%)	Avg. planning time (s)
Teriyaki NO-MACRO	81.1	9.2
0-shot GPT-4 agent	79.2 (↓ 2.3%)	7.6 (↓ 17.4%)
2-shot GPT-4 agent	84.6 (↑ 4.3%)	10.3 (↑ 12.0%)
5-shot GPT-4 agent	90.5 (↑ 11.6%)	16.7 (↑ 81.5%)

The arrow indicates the absolute value change in percentage

Table 3 presents our findings, showing a clear trend: more context ("shots") leads to better planning accuracy in LLM agents. Remarkably, the 5-shot GPT-4 agent reached an accuracy of 90.5%, surpassing Teriyaki's performance of 81.1%. Interestingly, even without additional examples or fine-tuning, the 0-shot GPT-4 agent performed nearly as well as Teriyaki with an accuracy of 79.2%.

However, there is a clear trade-off between accuracy and efficiency. The Teriyaki framework consistently shows a moderate planning time of 9.2 s. In contrast, the planning time for LLM agents increases with the number of shots, with the 5-shot GPT-4 agent taking 81.5% longer than Teriyaki at 16.7 s.

The conclusion drawn from these data is that LLM agents can outperform traditional frameworks like Teriyaki in accuracy even when given a few contextual data without training, but this comes at the cost of increased planning time due to the increased size of input tokens, which will raise the burden for LLM agents. Hence, the choice between using a Teriyaki-like framework or an LLM agent depends on the specific needs of the task: accuracy or efficiency.

Completion and evaluation

Next, we guided LLM agents to execute and complete the 16 selected tasks and evaluate their success rate. Additionally, we conducted ablation experiments on Guanaco-33b, Claude-2, LLaMa2-70b, GPT-3.5, and GPT-4 agents. These experiments focused on the error module (EM), API reference (API), and code reference (CR).

We found that the performance of the GPT-4 agent is noteworthy, achieving a remarkable accuracy of 93.75% on a single sample and an average of 81.88% on 10 samples. To determine whether these performance levels were significantly different between LLM agents, we used one-way analysis of variance (ANOVA) for statistical analysis. The results indicate significant performance variations (p-value $4.24 \cdot 10^{-12}$, F-value 29.64). Further examination using Tukey's Honestly Significant Difference test reveals that Claude-2's performance is comparable to that of GPT-3.5 and LLaMa2-70b ($p > 0.05$). However, there were larger performance gaps between the other agents, aligning with the trends of varying success rates shown in Table 4.

Our ablation experiments provide insight into the dependence of LLM agents on external references and internal modules. Remarkably, even without the error module, the GPT-4 agent maintains robust performance, achieving 81.25% (single sample) and 70.63% (10 samples). These results indicate that the GPT-4 agent can effectively process tasks without over-reliance on this specific module.

The significance of the API reference is evident across all models. The performance of the GPT-4 agent without the API is 62.50% (single sample) and 59.38% (10 samples), still leading the pack. Guanaco-33b agent, in this scenario, faced a steeper decline, especially when averaged over 10 samples, dropping to 19.44%. The most notable dependency is observed with the Guanaco-33b agent in the absence of the code reference, where its performance plummets to 0% for a single sample. On the contrary, the GPT-4 agent showcases its adaptability once again with scores of 37.50% (single sample) and 51.88% (10 samples). An interesting pattern emerges with the performance of the LLaMa2-70b agent without the reflection module: It scores 37.50% for a single sample but improves to 45.00% over 10 samples, suggesting potential scalability or adaptability in repeated tests.

To quantify the impact of each module, we conducted a repeated measures ANOVA. We verified that the data met the assumptions of normality and homogeneity before performing multiple comparisons with the Bonferroni correction. For

Table 4 Performance comparison of LLM agents using a single random sample versus an average of 10 samples, accompanied by results from the ablation study (standard deviation in parentheses)

Agent	Overall		W/O EM		W/O API		W/O CR	
	1	10	1	10	1	10	1	10
Guanaco-33b	50.00%	36.25% (±9.22%)	31.25%	32.50%	31.25%	19.44%	0.00%	11.11%
Claude-2	75.00%	61.25% (±11.33%)	43.75%	60.00%	37.50%	48.75%	25.00%	44.38%
LLaMa2-70b	56.25%	50.63% (±9.06%)	37.50%	45.00%	31.25%	40.63%	12.50%	37.50%
GPT-3.5	81.25%	63.75% (±11.33%)	50.00%	56.88%	31.25%	49.38%	18.75%	40.63%
GPT-4	**93.75%**	**81.88%** (±7.48%)	**81.25%**	**70.63%**	**62.50%**	**59.38%**	**37.50%**	**51.88%**

Bold entries highlight the GPT-4 agent achieved the best performance in our ablation study
The "W/O" prefix indicates respective modules are disabled

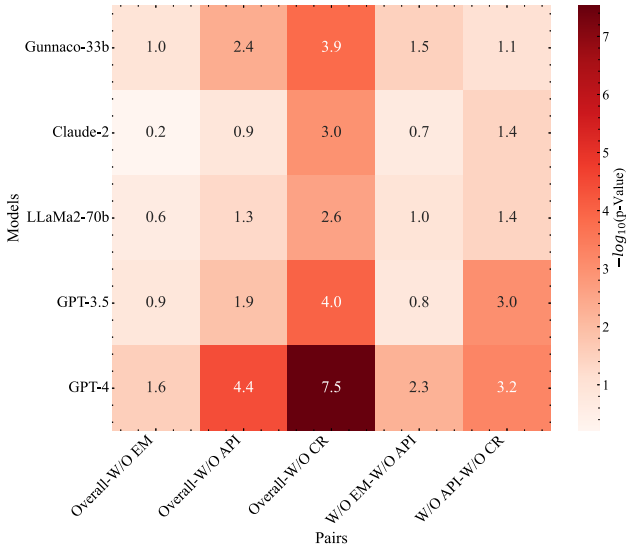


Fig. 6 Heatmap represents $-log_{10}$ transformed p-values for post hoc comparisons between conditions within each LLM. Higher values indicate more significant differences ($p < 0.05$ corresponds to a value of 1.3 on this scale)

clarity, we converted the p-values into their negative logarithms $-log_{10}$ and displayed them in a heatmap, as illustrated in Fig. 6, making it easier to discern which differences were statistically significant.

This heatmap simplifies the interpretation of our post hoc comparisons; deeper shades indicate stronger significance. For instance, we can quickly identify the most significant effects, such as GPT-4 agent's dependency on the "Code Reference" feature, which recorded a high significance value reflected by the deeper red color. In contrast, the "Error Module" appeared to have a lesser impact on all models. These findings are consistent with those of our earlier ablation study, offering a nuanced understanding of the influence of each module.

In conclusion, the GPT-4 agent consistently outperforms under various experimental conditions, demonstrating its resilience and adaptability. The Guanaco-33b agent displays specific dependencies, while the unique behaviors exhibited by the LLaMa2-70b agent merit further investigation.

To provide a clearer picture, we also illustrate some examples of agent-generated tasks in Fig. 7. These include tasks such as picking up and aligning a nut with a bolt, sorting cubes based on material into corresponding containers, and orchestrating 3D printing and milling tasks using two robots in sequence.

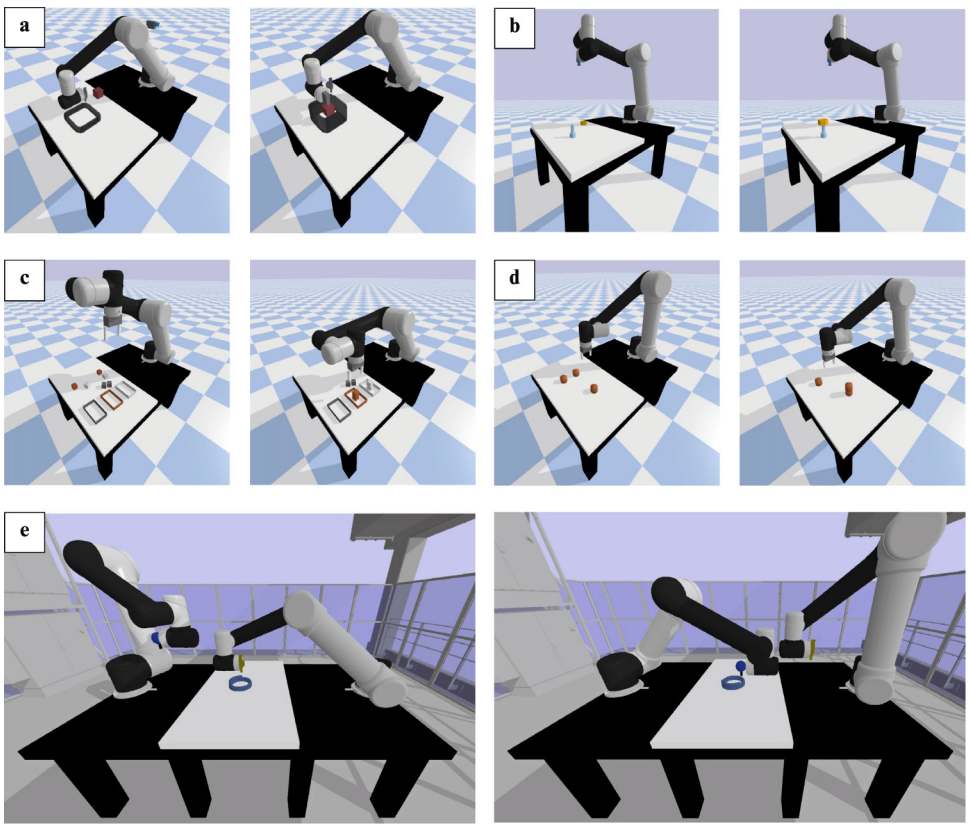


Fig. 7 Several examples of generated tasks designed and executed by LLM agents: **a** grip the small cube and drop it into the 3D-printed box. **b** Grip the nut and place it at the top of the bolt. **c** Grip the cubes with different materials in corresponding containers with matching materials. **d** Stack the cylinders. **e** Use two robots to perform 3D printing and milling in sequence

Discussion

In this section, we discuss the performance of LLM agents across various components, highlighting strengths and potential improvements. We also discuss the broader implications of our findings and suggest potential avenues for future research.

Component 1 The primary challenge is the agent's understanding of human-imposed constraints. For instance, even when explicitly instructed to avoid descriptive text, the agent tends to produce unnecessary output. Interestingly, this issue is not solely dependent on the size of the LLMs. Models with larger parameter sizes are more prone to this problem, whereas smaller models may simply produce incorrect answers.

Specific failure modes for Component 1 include mismatches in output data types, such as producing a string when a numerical value is expected. This suggests that LLM agents might occasionally misinterpret the finer requirements of input prompts. Moreover, the PaML-2 agent often generates lengthy descriptions, leading to increased processing times and potentially distorted outputs. Another concern is the model's choice of end-effector, which might arise from its challenges in understanding the full context of a task. We have introduced solutions like format conversion and removing extra information to tackle these problems. However, these fixes point out the model's intrinsic limitations.

Component 2 Within this component, the challenges center on the control capabilities of the LLM agent. We identified two main issues:

1. Agent's limited grasp of spatial relationships, especially in specific manufacturing processes.
2. Its diminished capability in few-shot learning for complex tasks, results in a greater reliance on external data references.

In Component 2, LLM agents excel in planar tasks but struggle with spatial tasks. Their difficulties with 3D printing tasks, where they often miss critical spatial relationships, result in impractical or incorrect tool paths. In addition, generating and interpreting G-code, due to its industrial complexity, remains a challenge. These shortcomings underscore the limitations of the component in industrial applications.

As we transition to *Component 3*, we see that LLM agents show a promising ability in advanced task planning, outperforming the conventional PDDL framework even with minimal examples and no training. This demonstrates the potential of LLM agents in sequence-to-sequence planning tasks. However, the structured nature of PDDL poses challenges in translating unstructured human instructions into task plans. Meanwhile, the performance of the GPT-4 agent in various scenarios showcases advances in LLM adaptability. In contrast, the performance drop of the Guanaco-33b agent without a code reference indicates a strong dependence on coded references. On the other hand, the improved performance of the LLaMa2-70b agent in repeated trials without the reflection module might suggest inherent resilience or alternate strategies.

Although our study is comprehensive, we acknowledge several limitations. Due to the constraints of the physical engine, every small object is treated as a rigid body. This complicates simulation tasks like milling and prevents achieving Digital-Twin level simulation. Additionally, in the 3D printing simulation, the particles are combined by adding constraints without considering mass, focusing purely on visual feasibility. We also simplified the 3D printing simulation process. Specifically, we overlooked the impact of different slicing and filling methods on path generation in real-world scenarios, due to the agents' poor performance in understanding spatial relationships. Finally, although we possess a real robot intended for 3D printing to achieve greater spatial freedom, it remains in the debugging phase.

Conclusion

In this study, we explore the potential of LLM agents to guide industrial robotics. Our methodology encompasses three components: task and process parameter matching, autonomous tool path design, and exploration of embodied intelligence. This methodology illuminates both the strengths and weaknesses of LLM agents.

While some agents, such as Claude-2, perform well in Component 1, others, particularly in complex 3D tasks, yield less accurate outputs. Component 2 underscores the agents' proficiency in planar tasks but also reveals their struggles with 3D spatial relationships. Component 3 is particularly insightful. The GPT-4 agent exhibits remarkable adaptability, with an accuracy rate of 93.75% for a single sample and an average of 81.88% across 10 samples. However, agents such as Guanaco-33b show a significant dependence on external sources, such as code references. In particular, the accuracy of task planning with the GPT-4 agent surpasses traditional frameworks like Teriyaki in accuracy, showcasing how its strong few-shot learning capabilities contribute to the intelligence level.

We observe challenges in the agents' handling of complex manufacturing processes, notably in 3D path planning. The introduction of modules like the error module and the reflection mechanism improves performance, emphasizing the importance of memory buffers and external references. Furthermore, agents guided by the CoT method outperform those relying on direct prompts, which aligns with the human logic chain.

These findings have broader implications for the field of manufacturing. As LLM agents become more integrated into manufacturing processes, their ability to optimize and automate tasks can lead to significant advances in efficiency and precision. However, the challenges identified also underscore the need for continuous refinement and adaptation. In conclusion, while LLM agents hold significant potential for advancing industrial robotics, there is room for improvement. Looking ahead, we encourage researchers to delve into areas like visual semantic control, refining memory modules, and implementing real-time feedback loops. As technology continues to evolve, LLM agents stand at the forefront of transforming the automation and optimization of manufacturing processes, and further exploration in this domain is paramount.

Future outlook

Our study has laid the groundwork for expanding the capabilities of LLM agents in industrial tasks, providing a baseline for further research in this growing domain. The avenues proposed here are designed not only to enhance efficacy, but also to broaden the applicability of LLM agents to handle complex industrial tasks with greater robustness and adaptivity. These improvements are poised to: (1) improve the LLM agent's understanding of three-dimensional structures and optimize the path generation algorithm; (2) explore the prospects of our framework in solving the field of multi-robot cooperation in industry; (3) in addition to desktop robots, consider using mobile robots to cover actual industrial needs more broadly; and (4) empower agents with multi-modal processing capabilities.

Building on the challenges identified in our study, one critical advance involves the refinement of memory modules based on task-related feedback to improve spatial understanding. Such an iterative learning approach, highlighted in a recent study (Liang et al., 2023), is essential for adapting LLM agents to a broader spectrum of challenges. Path optimization remains an area ripe for development. The current trajectory of the end-effector, which is constrained to collision avoidance and task completion, can be made more efficient by incorporating a cost function. Such a function would allow for optimization of the path in terms of energy consumption, time, or other relevant metrics, leading to a more refined and economical operation.

Moreover, the prospect of collaborative strategies, wherein multiple LLM agents work in concert, emerges as a promising frontier to tackle multifaceted tasks. This synergy could harness the collective intelligence and capabilities of agents, leading to more effective solutions (Hong et al., 2023; Nascimento et al., 2023). Implementing a real-time feedback mechanism is another cornerstone in achieving optimal performance. Such a system would allow agents to immediately recalibrate their actions, aligning with the dynamic demands of the task at hand (Zhang et al., 2023).

In the realm of practical applications, our current simulations with desktop robots offer a glimpse into the potential of LLM agents. Including mobile robots in our framework could extend its applicability to logistics and supply chain management (Heuss et al., 2023). This expansion allows us to explore the framework's application value further in these sectors.

Lastly, inspired by (Driess et al., 2023; Hu et al., 2022), we see the potential in converting images from industrial scenes into robotics planning and programming. Given our recent work, integrating VLMs into our framework could be transformative. The ability of VLMs to leverage multimodal information makes the realization of embodied intelligence more promising. This conversion process enhances our framework's practical utility and paves the way for a more intuitive and effective human–robot interaction.

In conclusion, the proposed enhancements are poised to significantly advance the adaptability, efficiency, and practicality of LLM agents within and beyond manufacturing. These focal areas will form the cornerstone of our future research endeavors, each contributing to the overarching goal of creating more autonomous, intelligent, and versatile industrial robotic agents.

Acknowledgements The two Ph.D. students Mr. Haolin Fan and Mr. Xuan Liu are supported by the National University of Singapore Chongqing Research Institute under the NUS Research Scholarship. This research is partially supported by the project "The Sustainable Manufacturing Alliances for Research and Training Industry Assessment Center (The SMART IAC)" funded by the U.S. Department of Energy's Office of Manufacturing and Energy Supply Chains (Award Number: DE-EE0009726). We acknowledge Prof. Larry Smarr and Dr. Ilkay Altintas from University of California San Diego for HyperCluster computing support of National Research Platform (NRP) Nautilus.

Author contributions Conceptualization, HF and BL; methodology, HF and BL; case study, HF and XL; validation, HF and XL; formal analysis, HF; investigation, HF and BL; resources, BL; data curation, HF and XL; writing—original draft preparation, HF, XL and BL; writing—review and editing, HF, XL, JYHF, WFL and BL; visualization, HF and XL; supervision, JYHF, WFL, and BL; project administration, BL; funding acquisition, BL. All authors have read and agreed to the published version of the manuscript.

Data availability Not applicable.

Declarations

Competing interests The authors declare that they have no competing interests.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., ... Zeng, A. (2022). Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., El Shafey, L., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., ..., Wu, Y. (2023). Palm 2 technical report. arXiv:2305.10403

Anthropic. (2023). Model card and evaluations for Claude models. https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., & Sutton, C. (2021). Program synthesis with large language models. arXiv:2108.07732

Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., Lovenia, H., Ji, Z., Yu, T., Chung, W., Do, Q. V., Xu, Y., & Fung, P. (2023). A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. arXiv:2302.04023

Bezrucav, S.-O., & Corves, B. (2022). Modelling automated planning problems for teams of mobile manipulators in a generic industrial scenario. *Applied Sciences, 12*(5), 2319. https://doi.org/10.3390/app12052319

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., ..., Zitkovich, B. (2023). RT-1: Robotics transformer for real-world control at scale. arXiv:2212.06817

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Arvind, N., Pranav, S., Girish, S., Amanda, A., Sandhini, A., Ariel, H.-V., Gretchen, K., Tom, H., Rewon, C., Aditya, R., Daniel, Z., Jeffrey, W., Clemens, W., ..., Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates, Inc.

Buerkle, A., Eaton, W., Al-Yacoub, A., Zimmer, M., Kinnell, P., Henshaw, M., Coombes, M., Chen, W.-H., & Lohse, N. (2023). Towards industrial robots as a service (IRAAS): Flexibility, usability, safety and business models. *Robotics and Computer-Integrated Manufacturing, 81*, 102484. https://doi.org/10.1016/j.rcim.2022.102484

Capitanelli, A., & Mastrogiovanni, F. (2023). A framework to generate neurosymbolic PDDL-compliant planners. arXiv:2303.00438

Chen, J.-T., & Huang, C.-M. (2023). Forgetful large language models: Lessons learned from using LLMS in robot programming. arXiv:2310.06646

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., ..., Zaremba, W. (2021). Evaluating large language models trained on code. arXiv:2107.03374

Chen, P.-L., & Chang, C.-S. (2023). Interact: Exploring the potentials of ChatGPT as a cooperative agent. arXiv:2308.01552

Choi, D., Shi, W., Liang, Y. S., Yeo, K. H., & Kim, J. -J. (2021). Controlling industrial robots with high-level verbal commands. In *Social robotics* (pp. 216–226). Springer International Publishing.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P., & Oudeyer, P.-Y. (2020). Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems, 33*, 3761–3774.

Coumans, E., & Bai, Y. (2016–2021). PyBullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMS. arXiv:2305.14314

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., ..., Florence, P. (2023). PaLM-E: An embodied multimodal language model. arXiv:2303.03378

Drori, I., Zhang, S., Shuttleworth, R., Tang, L., Lu, A., Ke, E., Liu, K., Chen, L., Tran, S., Cheng, N., Wang, R., Singh, N., Patti, T. L., Lynch, J., Shporer, A., Verma, N., Wu, E., & Strang, G. (2022). A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences, 119*(32), e2123433119. https://doi.org/10.1073/pnas.2123433119

Goel, R., & Gupta, P. (2020). *Robotics and industry 4.0. A roadmap to industry 4.0: Smart production, sharp business and sustainable development* (pp. 157–169). https://doi.org/10.1007/978-3-030-14544-6_9

Hägele, M., Nilsson, K., Pires, J. N., & Bischoff, R. (2016). Industrial robotics. In *Springer handbook of robotics* (pp. 1385–1422). https://doi.org/10.1007/978-3-319-32552-1_54

Heuss, L., Gebauer, D., & Reinhart, G. (2023). Concept for the automated adaption of abstract planning domains for specific application cases in skillsbased industrial robotics. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02211-3

Hoebert, T., Lepuschitz, W., Vincze, M., & Merdan, M. (2021). Knowledge-driven framework for industrial robotic systems. *Journal of Intelligent Manufacturing, 34*(2), 771–788. https://doi.org/10.1007/s10845-021-01826-8

Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Ka, S., Yau, S., Lin, Z., Zhou, L., Ran, C., Xiao, L., & Wu, C. (2023). MetaGPT: Meta programming for multi-agent collaborative framework. arXiv:2308.00352

Hu, H., Chen, J., Liu, H., Li, Z., & Huang, L. (2022). Natural language-based automatic programming for industrial robots. *Journal of Grid Computing, 20*(3), 26–44. https://doi.org/10.1007/s10723-022-09618-x

Huang, C., Mees, O., Zeng, A., & Burgard, W. (2023a). Visual language maps for robot navigation. *IEEE International Conference on Robotics and Automation (ICRA), 2023*, 10608–10615. https://doi.org/10.1109/ICRA48891.2023.10160969

Huang, S., Jiang, Z., Dong, H., Qiao, Y., Gao, P., & Li, H. (2023b). Instruct2Act: Mapping multimodality instructions to robotic actions with large language model. arXiv:2305.11176

Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning* (pp. 9118–9147).

Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., & Fei-Fei, L. (2023c). VoxPoser: Composable 3D value maps for robotic manipulation with language models. arXiv:2307.05973

Huang, W., Xia, F., Shah, D., Driess, D., Zeng, A., Lu, Y., Florence, P., Mordatch, I., Levine, S., Hausman, K., & Ichter, B. (2023d). Grounded decoding: Guiding text generation with grounded models for robot control. arXiv:2303.00855

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., & Ichter, B. (2023e). Inner monologue: Embodied reasoning through planning with language models. In K. Liu, D. Kulic, & J. Ichnowski (Eds.),

Proceedings of the 6th conference on robot learning (Vol. 205, pp. 1769–1782). PMLR.

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., Finn, C., & Finn, C. (2022). BC-Z: Zeroshot task generalization with robotic imitation learning. In *Proceedings of the 5th conference on robot learning* (pp. 991–1002).

Jiang, Y., Gu, S., Murphy, K., & Finn, C. (2019). Language as an abstraction for hierarchical deep reinforcement learning. arXiv:1906.07343

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2023). Large language models are zero-shot reasoners. arXiv:2205.11916

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *2010 5th ACM/IEEE international conference on human–robot interaction (HRI)* (pp. 259–266). https://doi.org/10.1109/HRI.2010.5453186

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2014). Grounding verbs of motion in natural language commands to robots. In *Experimental robotics: The 12th international symposium on experimental robotics* (pp. 31–47). https://doi.org/10.1007/978-3-642-28572-1_3

Kwon, M., Xie, S. M., Bullard, K., & Sadigh, D. (2023). Reward design with language models. arXiv:2303.00001

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., & Zeng, A. (2023). Code as policies: Language model programs for embodied control. *IEEE International Conference on Robotics and Automation (ICRA), 2023*, 9493–9500. https://doi.org/10.1109/ICRA48891.2023.10160591

Liang, K.-H., Davidson, S., Yuan, X., Panditharatne, S., Chen, C.-Y., Shea, R., Pham, D., Tan, Y., Voss, E., & Fryer, L. (2023). ChatBack: Investigating methods of providing grammatical error feedback in a GUI-based language learning chatbot. In *Proceedings of the 18th workshop on innovative use of NLP for building educational applications (BEA 2023)* (pp. 83–99). https://doi.org/10.18653/v1/2023.bea-1.7

Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). Visual instruction tuning. arXiv:2304.08485

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., & Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language. arXiv:1906.03926

Misra, D., Langford, J., & Artzi, Y. (2017). Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 1004–1015). https://doi.org/10.18653/v1/D17-1106

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N., Rocktäschel, T., & Grefenstette, E. (2022). Improving intrinsic exploration with language abstractions. *Advances in Neural Information Processing Systems, 35*, 33947–33960.

Nair, S., Mitchell, E., Chen, K., Ichter, B., Savarese, S., & Finn, C. (2022). Learning language-conditioned robot behavior from offline data and crowdsourced annotation. In *Proceedings of the 5th conference on robot learning* (Vol. 164, pp. 1303–1315).

Nascimento, N., Alencar, P., & Cowan, D. (2023). Self-adaptive large language model (LLM)-based multiagent systems. arXiv:2307.06187

Neunzig, C., Möllensiep, D., Kuhlenkötter, B., & Möller, M. (2023). ML Pro: Digital assistance system for interactive machine learning in production. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02214-0

OpenAI. (2023). GPT-4 technical report. arXiv:2303.08774

Paul, R., Arkin, J., Aksaray, D., Roy, N., & Howard, T. M. (2018). Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms. *The International Journal of Robotics Research, 37*(10), 1269–1299. https://doi.org/10.1177/0278364918777627

Peng, B., Li, C., He, P., Galley, M., & Gao, J. (2023). Instruction tuning with GPT-4. arXiv:2304.03277

Perzylo, A., Somani, N., Profanter, S., Kessler, I., Rickert, M., & Knoll, A. (2016). Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016*, 2293–2300. https://doi.org/10.1109/IROS.2016.7759358

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? arXiv:1909.01066

Poesia, G., Polozov, O., Le, V., Tiwari, A., Soares, G., Meek, C., & Gulwani, S. (2022). Synchromesh: Reliable code generation from pre-trained language models. arXiv:2201.11227

Raman, S. S., Cohen, V., Paulius, D., Idrees, I., Rosen, E., Mooney, R., & Tellex, S. (2023). Cape: Corrective actions from precondition errors using large language models. arXiv:2211.09935

Ren, P., Zhang, K., Zheng, H., Li, Z., Wen, Y., Zhu, F., Ma, M., & Liang, X. (2023). RM-PRT: Realistic robotic manipulation simulator and benchmark with progressive reasoning tasks. arXiv:2306.11335

Rovida, F., Crosby, M., Holz, D., Polydoros, A. S., Großmann, B., Petrick, R. P. A., & Krüger, V. (2017). SkiROS—A skill-based robot control platform on top of ROS. *Robot Operating System (ROS) The Complete Reference (Volume 2)*, 121–160. https://doi.org/10.1007/978-3-319-54927-9_4

Shah, D., Osiński, B., Ichter, B., & Levine, S. (2023). LM-NAV: Robotic navigation with large pretrained models of language, vision, and action. In *Proceedings of the 6th conference on robot learning* (pp. 492–504).

Sharma, P., Sundaralingam, B., Blukis, V., Paxton, C., Hermans, T., Torralba, A., Andreas, J., & Fox, D. (2022). Correcting robot plans with natural language feedback. arXiv:2204.05186

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2023). ProgPrompt: Generating situated robot task plans using large language models. *IEEE International Conference on Robotics and Automation (ICRA), 2023*, 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., & Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 25, pp. 1507–1514).

Thomason, J., Zhang, S., Mooney, R., & Stone, P. (2015). Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th international conference on artificial intelligence* (pp. 1923–1929). https://doi.org/10.5555/2832415.2832516

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., ..., Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288

Wächter, M., Ovchinnikova, E., Wittenbeck, V., Kaiser, P., Szedmak, S., Mustafa, W., Kraft, D., Krüger, N., Piater, J., & Asfour, T. (2018). Integrating multi-purpose natural language understanding, robot's memory, and symbolic planning for task execution in humanoid robots. *Robotics and Autonomous Systems, 99*, 148–165. https://doi.org/10.1016/j.robot.2017.10.012

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent abilities of large language models. arXiv:2206.07682

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models. arXiv:2201.11903

Yang, Y., Zhang, X., & Han, W. (2023). Enhance reasoning ability of visual-language models via large language models. arXiv:2305.13267

Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., Zhou, J., Chen, S., Gui, T., Zhang, Q., & Huang, X. (2023). A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. arXiv:2303.10420

Yoneda, T., Fang, J., Li, P., Zhang, H., Jiang, T., Lin, S., Picker, B., Yunis, D., Mei, H., & Walter, M. R. (2023). Statler: State-maintaining language models for embodied reasoning. arXiv:2306.17840

Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., & Florence, P. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. arXiv:2204.00598

Zhang, D., Chen, L., Zhao, Z., Cao, R., & Yu, K. (2023). Mobile-Env: An evaluation platform and benchmark for interactive agents in LLM era. arXiv:2305.08144

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ..., Wen, J.-R. (2023). A survey of large language models. arXiv:2303.18223