1 - Large Language Models for Robotics: Opportunities, Challenges, and Perspectives

Abstract—Large language models (LLMs) have undergone significant expansion and have been increasingly integrated across various domains. Notably, in the realm of robot task planning, LLMs harness their advanced reasoning and language comprehension capabilities to formulate precise and efficient action plans based on natural language instructions. However, for embodied tasks, where robots interact with complex environments, text-only LLMs often face challenges due to a lack of compatibility with robotic visual perception. This study provides a comprehensive overview of the emerging integration of LLMs and multimodal LLMs into various robotic tasks. Additionally, we propose a framework that utilizes multimodal GPT-4V to enhance embodied task planning through the combination of natural language instructions and robot visual perceptions. Our results, based on diverse datasets, indicate that GPT-4V effectively enhances robot performance in embodied tasks. This extensive survey and evaluation of LLMs and multimodal LLMs across a variety of robotic tasks enriches the understanding of LLM-centric embodied intelligence and provides forward-looking insights toward bridging the gap in Human-Robot-Environment interaction.

III. SCOPE OF ROBOTIC TASKS

A. Planning

1) Natural Language Understanding : In robot planning, Large Language Models excel due to their advanced natural language comprehension. They translate natural language instructions into executable action sequences for robots, a crucial aspect of robot planning [61], [68]. This study reveals that LLMs can generate accurate action sequences based on linguistic instructions alone, even without visual input [69]. However, their performance is enhanced significantly with a modest amount of visual information, enabling them to create precise visual-semantic plans. These plans transform highlevel natural language instructions into actionable guidance for virtual agents to undertake complex tasks. This ability underscores the potential of LLMs to integrate multimodal information, thereby improving their comprehension. It also demonstrates their capacity to interpret and incorporate information from various modalities, leading to a more comprehensive task understanding [70]. Moreover, research in generating action sequences from a large language model for natural language understanding further confirms the efficacy of LLMs in robot planning. LLMs also show great promise in interpreting natural language commands in sync with the physical environment. Employing the Grounded Decoding approach, they can produce behavior sequences that align with the probabilities of the physical model, showcasing this method's effectiveness in robot planning tasks [71].

The research in complex sequential task planning has highlighted significant advancements in the capabilities of LLMs. Text2Motion's studies demonstrate that LLMs are adept not only at processing linguistic information but also at addressing dependencies in skill sequences [72]. This is achieved through geometrically feasible planning, marking a crucial advancement in the interpretation of abstract instructions and the comprehension of intricate task structures. In addition, LLMPlanner research enhances the natural language understanding abilities of LLMs in robotic planning by integrating them with conventional planners [73]. This synergy illustrates how the NLP proficiencies of LLMs can be harnessed to boost the efficiency and precision of planning tasks. Moreover, LLM+P harnesses the

capabilities of classical planners, employing the Planning Domain Definition Language (PDDL) and problem cues to create task-specific problem files for LLMs [44]. This integration significantly amplifies LLMs' efficacy in addressing long-term planning tasks. Also, SayPlan addresses the issue of planning horizon by integrating a classical path planner. By doing so, SayPlan is capable of grounding largescale, long-horizon task plans derived from abstract and natural language instructions, enabling a mobile manipulator robot to execute them successfully [74]. Furthermore, LLMs have shown promise in acting as heuristic strategies within search algorithms, while also serving as reservoirs of common-sense knowledge. This dual role of LLMs not only enhances the reasoning capabilities within these algorithms but also aids in forecasting potential outcomes. Such an approach harnesses the full potential of LLMs, leveraging their advanced reasoning capabilities for the effective planning of complex tasks [66].

This dual application underscores the extensive and versatile potential of large language models in task planning and problem-solving.

The research conducted on LLMs has showcased their remarkable ability to parse and comprehend natural language understanding. This capability extends beyond mere text matching to a profound semantic understanding, encompassing the tasks' purpose and context. A critical aspect of LLMs is translating the instructions they comprehend into executable action sequences for robots, an essential feature in robot task planning. LLMs significantly enhance the quality and adaptability of instruction generation, enabling the creation of complex action sequences that are both context-aware and environment-specific. These models demonstrate versatility in managing various task-planning complexities and types, from straightforward physical interactions to intricate, long-term sequence planning. The studies highlight LLMs' potential as both independent decision-makers and collaborators with other modalities and planning algorithms. This collaboration is pivotal in interpreting natural language and advancing robotic planning. As research progresses, LLMs are expected to play an increasingly vital role in the fields of robotics and automated systems.

2) Complex Task Reasoning and Decision-making: In the realm of complex task reasoning and decision-making, robots empowered by LLMs have shown remarkable proficiency. These LLM-based robotic planning tasks have significantly transcended the realms of mere text generation and language comprehension. Recent research highlights the immense capabilities of Language Models in managing intricate tasks, engaging in logical reasoning, making informed decisions, and partaking in interactive learning. [3], [75] These breakthroughs have not only expanded our comprehension of LLM-based robotic planning's potential but also opened the door to innovative practical applications.

In exploring the application of pre-trained language models (PLMs) in interactive decision-making, research has demonstrated how targets and observations are transformed into embedding sequences, initializing the network with PLMs. This strategy's generalization ability is particularly effective in multivariate environments and supervised modalities [76]. A notable advancement in the multimodal domain is the development of the LM-Nav system [59]. This system, grounded in PLMs, integrates language, vision, and action models to guide robotic navigation via high-level natural language commands. Significantly, it reduces dependency on costly trajectory annotation supervision by merging pre-trained visual navigation, image-verbal correlation, and language understanding models. Focusing on

LLMs in specific environments, researchers [65] have examined their capacity for reasoning with natural language feedback and complex task planning. This capability is crucial for following high-level task instructions and enhancing the model's applicability in real-world scenarios. Addressing the issue of consistency fault-tolerance in natural language understanding and decision-making, the innovative ReAct model [77] overcomes prior limitations of linguistic reasoning in interactive settings. It tackles challenges like hallucination generation and misinformation propagation. By

leveraging LLMs' potential to maintain working memory and abstractly conceptualize high-level goals, the ReAct model achieves significant performance improvements across various tasks. In parallel, to address confidently hallucinated predictions in large language models (LLMs) applied to robotics, KnowNo [78] provides statistical guarantees for task completion while minimizing the need for human assistance in complex multi-step planning scenarios. Notably, KnowNo seamlessly integrates with LLMs without requiring modelfinetuning, offering a lightweight and promising method to model uncertainty. This approach aligns with the constantly evolving capabilities of foundation models, providing a scalable solution. Further, a strategy involving preconditioned error cues has been proposed, enabling LLMs to extract executable plans. This approach offers a fresh perspective on the independence and adaptability of agents in task execution. In terms of multi-agent collaboration, the integration of language models with action agents is increasingly being explored. By pairing LLMs with agents executing tasks in specific environments, a system comprising planners, executors, and reporters is established. This arrangement markedly enhances the efficiency of reasoning and execution in complex tasks.

The burgeoning field of large pre-trained LMs is witnessing a notable trend: these models are increasingly adept at understanding and performing complex tasks, closely aligning with real-world scenarios. This advancement not only underscores the adaptability and versatility of pre-trained models but also heralds the advent of next-generation AI. As these technologies evolve, we anticipate a surge in innovative applications, poised to revolutionize various industries. A key aspect of these tasks is the utilization of LLMs' robust language comprehension and generation capabilities for intricate reasoning and decision-making processes. Each study in this domain explores the potential of LLMs in complex cognitive functions. Many models employ self-supervised learning, with some incorporating fine-tuning to better align with specific tasks. This approach enables LLMs to excel in downstream task-assisted reasoning, leading to more precise and tailored decisions. Despite the widespread use of LLMs in complex reasoning and decision-making, the specific techniques and approaches vary, particularly in terms of task handling, learning strategies, and feedback mechanisms. These models find applications in diverse real-world contexts, including home automation, robot navigation, and task planning, demonstrating their broad and evolving utility.

3) Human-robot interaction: In the realm of human-robot interaction, the advanced reasoning capabilities of AGI language models empower robots with a significant degree of generalization ability. [79] This enables them to adapt to new task planning in previously unseen environments and tasks. Furthermore, the natural language understanding interface of LLMs facilitates communication with humans, opening new possibilities for human-robot interactions. [80] Extensive research has underscored the progress made by LLMs in aiding intelligent task planning, which in turn enhances multi-intelligence collaborative communication. Studies have found that using natural language to boost

the efficiency of multi-intelligence cooperation is an effective method to enhance communication efficiency. A notable example of this is OpenAI's ChatGPT, whose capabilities in robotics applications were evaluated through rigorous experiments. The findings revealed that ChatGPT excels in complex tasks such as logical, geometric, and mathematical reasoning, along with airborne navigation, manipulation, and controlling embodied agents [48]. It achieves this through techniques like freeform dialogue, parsing XML tags, and synthesizing code. Furthermore, ChatGPT allows user interaction via natural language commands, providing vital guidance and insights for the development of innovative robotic systems that interact with humans in a natural and intuitive way. In a similar vein, there is a proposed framework that leverages large-scale language models for collaborative embodied intelligence [81]. This framework enables the use of language models for efficient planning and communication, facilitating collaboration between various intelligences and humans to tackle complex tasks. Experimental results demonstrate that this approach significantly outperforms traditional methods in the field.

B. Manipulation

1) Natural Language Understanding: In the field of robot control, the natural language understanding capabilities of LLM can help robots make common-sense analyses. For example, LLM-GROP demonstrates how semantic information can be extracted from LLM and used as a way to make common-sense, semantically valid decisions about object placement as part of a task and motion planner that performs multistep tasks in complex environments in response to natural language commands [82]. The research proposes a framework for placing language at the core of an intelligent body [83]. By utilizing the prior knowledge contained in these models, better robotic agents can be designed that are able to solve challenging tasks directly in the real world. Through a series of experiments, it is demonstrated how the framework can be used to solve a variety of problems with greater efficiency and versatility by utilizing the knowledge and functionality of the underlying models. At the same time, the study introduces Linguistically Conditional Collision Function (LACO), a novel method to learn collision functions using only single-view image, language prompt, and robot configuration. LACO predicts collisions between robots and the environment, enabling flexible conditional path planning [84].

Outside of natural language understanding capabilities, the powerful reasoning capabilities of LLM also have a prominent role. For example, in the VIMA work [85], a novel multimodal cueing formulation is introduced to convert different robot manipulation tasks into a unified sequence modeling problem and instantiated in a diverse benchmark with multimodal tasks and system generalization evaluation protocols. Experiments show that VIMA is capable of solving tasks such as visual goal realization, one-off video imitation, and novel conceptual foundations using a single model, with robust model scalability and zero-sample generalization. Similarly, TIP proposes Text-Image Cueing [86], a bimodal cueing framework that connects LLMs to multimodal generative models for rational multimodal program plan generation.

In addition to prompt methods, fine-tuning downstream tasks based on pre-trained LMs is also a common approach in the field of robot control. For example, the work demonstrated that pre-trained

visual language representations can effectively improve the sample efficiency of existing exploratory methods [87]. R3M investigates how pre-trained visual representations on different human video data can enable data-efficient learning of downstream robot manipulation tasks [88]. The LIV is trained on a large generalized human video dataset, and fine-tuned on a small robot data set, is fine-tuned to outperform state-of-the-art methods in three different evaluation settings, and successfully performs real-world robotics tasks [89].

This collection of studies collectively illustrates the significant role of LLMs and Natural Language Understanding techniques in advancing robotic intelligence, particularly in comprehending and executing complex, language-based tasks. A key emphasis of these studies is on the importance of model generalization and the ability to apply these models across various domains. Each study, while sharing this common theme, diverges in its specific focus and application methodology. For instance, LLM-GROP is dedicated to the extraction and application of semantic information. In contrast, VIMA and TIP concentrate on multimodal processing and learning without prior examples. Furthermore, methodologies that finetune pre-trained LMs are directed toward enhancing application efficiency and task-specific optimization. Collectively, these studies demonstrate that integrating sophisticated NLP techniques with machine learning strategies can substantially enhance the efficiency of robotic systems, particularly in their ability to understand and perform intricate tasks. This advancement is a crucial stride towards achieving greater intelligence and autonomy in robotic manipulation.

2) Interactive Strategies: In the realm of interactive strategies, the TEXT2REWARD framework introduces an innovative approach for generating interactive reward codes using LLMs [83]. This method automatically produces dense reward codes, enhancing reinforcement learning. Also, By utilizing Large Language Models to define reward parameters that can be optimized to accomplish a variety of robotic tasks, the gap between high-level language instructions or corrections and low-level robot actions can be effectively bridged. The rewards generated by the language models serve as an intermediate interface, enabling seamless communication and coordination between high-level instructions and low-level actions of the robot [90]. Furthermore, VoxPoser presents a versatile framework for robot manipulation [64], distinct in its ability to extract manipulability and constraints directly from LLMs. This approach significantly enhances the adaptability of robots to open-set instructions and diverse objects. By integrating LLMs with vision-language models and leveraging online interactions, VoxPoser efficiently learns to interact with complex task dynamics models. The application of LLMs extends to human-robot interaction as well. The LILAC system exemplifies this through a scalable [63], language-driven interaction mechanism between humans and robots. It translates natural language discourse into actionable commands within a low-dimensional control space, enabling precise and userfriendly guidance of robots. Importantly, each user correction refines this control space, allowing for increasingly targeted and accurate commands. InstructRL offers another innovative framework designed to enhance human-AI collaboration [91]. It focuses on training reinforcement learning agents to interpret and act on natural language instructions provided by humans. This system employs LLMs to formulate initial policies based on these instructions, guiding reinforcement learning agents toward achieving an optimal balance in coordination. Lastly, for language-based human-machine interfaces, a novel, flexible interface LILAC has been developed. It permits users to alter robot trajectories using textual input and scene imagery [92].

This system synergizes pre-trained language and image models like BERT and CLIP, employing transformer encoders and decoders to manipulate robot trajectories in both 3D and velocity spaces. Proving effective in simulated environments, this approach has also demonstrated its practicality through real-world applications.

All of these techniques and approaches depend, to varying degrees, on advanced language modeling to enhance humanrobot interaction and robot control. They collectively underscore the crucial role of LLMs in interpreting and executing human intentions. Each method aims to boost the adaptability and flexibility of robots, enabling them to handle diverse tasks and environments more effectively. Specifically, TEXT2REWARD centers on generating and optimizing reward codes. This enhances the efficacy of reinforcement learning strategies. Conversely, VoxPoser focuses on extracting operants and constraints from LLMs. Meanwhile, LILAC and InstructRL adopt distinct approaches to interpreting and executing natural language commands. LILAC prioritizes mapping discourse to a control space, whereas StructRL dedicates itself to training reinforcement learning agents to comprehend and follow natural language instructions. Additionally, the last discussed language-based Human-Machine Interaction research investigates how to directly extract user intentions from text and images, applying them across various robot platforms. This aspect sets it apart from other approaches that might not incorporate this feature. Collectively, these studies mark substantial advancements in integrating LLMs techniques into robotics. While their application areas and methodologies have distinct focal points, they collectively demonstrate the potential for innovation in artificial intelligence. Furthermore, they pave the way for future explorations in human-robot interaction.

3) Modular Approaches: Recent advancements in robot control emphasize modular approaches, allowing the creation of more complex and feature-rich robotic systems. Key aspects of this trend have been highlighted in recent research. PROGRAMPORT proposes a program-based modular framework focused on robot manipulation [93]. It interprets and executes linguistic concepts by translating natural language's semantic structure into programming elements. The framework comprises neural modules that excel in learning both general visual concepts and task-specific operational strategies. This structured approach distinctly enhances learning of visual foundations and operational strategies, improving generalization to unseen samples and synthetic environments.

Next, researchers have explored the use of LLMs to expedite strategy adaptation in robotic systems [94], particularly when

encountering new tools. By generating geometrical shapes and descriptive tool models, and then converting these into vector representations, LLMs facilitate rapid adaptation. This integration of linguistic information and meta-learning has shown significant performance improvements in adapting to unfamiliar tools.

In addition, Combining NLMap [95], a visual language model based on ViLD and CLIP, with the SayCan framework, has led to a more flexible scene representation. This combination is particularly effective for long-term planning, especially when processing natural language commands in open-world scenarios. NLMap enhances the capability of LLM-based planners to understand their environments.

The ”Scaling Up and Distilling Down” framework combines the advantages of LLMs [96], sampling-based planners, and policy learning. It automates the generation, labeling, and extraction of rich robot exploration experiences into a versatile visual-linguistic motion strategy. This multi-task strategy not only inherits long-term behavior and robust manipulation skills but also shows improved performance in scenarios outside the training distribution.

MetaMorph introduces a Transformer-based method for learning a generalized controller applicable to a vast modular robotic design space [97]. This approach enables the use of robot morphology as a Transformer model output. By pretraining on a diverse range of morphologies, strategies generated through this approach demonstrate broad generalizability to new morphologies and tasks. This showcases the potential for extensive pre-training and fine-tuning in robotics, akin to developments in vision and language fields.

In each of these studies, a modular approach has been adopted, enhancing the system’s flexibility and adaptability to new tasks and environments. These works extensively utilize deep learning techniques, notably in synergy with LLMs, to augment the robotic system’s understanding and decision-making abilities. Moreover, a significant focus of these studies is the application of NLP. This is evident either through the direct interpretation of linguistic commands or via linguistically enriched learning and adaptation processes. The primary objective is to enhance the robot’s capability for quick generalization and adaptation in novel environments and tasks. While all the studies employ deep learning and LLMs, their specific implementations and applications are diverse. Some are centered on linguistic description and comprehension, while others explore the fusion of vision and language. The research goals are varied, addressing challenges from adapting to new tools, to long-term strategic planning, to polymorphic robot control. Despite differences in technical approaches, application areas, and targeted tasks, each study significantly contributes to advancing the intelligence and adaptive capabilities of robotic systems.

C. Reasoning

1) Natural Language Understanding: In the realm of robotic reasoning tasks, LLMs based on natural language understanding serve as an essential knowledge base, providing common sense insights crucial for various tasks. Extensive research has shown that LLMs effectively simulate human-like states and behaviors, especially relevant in the study of robots performing household cleaning functions. This approach deviates from traditional methods, which typically require costly data gathering and model training. Instead, LLMs leverage off-the-shelf methods for generalization in robotics, benefiting from their robust summarization abilities honed from extensive textual data analysis. Moreover, the common sense reasoning and code comprehension capabilities of LLMs foster connections between robots and the physical world. For instance, Progprompt introducing programming language features in LLMs has been shown to enhance task performance. This approach is not only intuitive but also sufficiently flexible to adapt to new scenarios, agents, and tasks, including actual robot deployments [98]. Concurrently, GIRAF harnesses the power of large language models to more flexibly interpret gestures and language commands, enabling accurate inference of human intentions and contextualization of gesture meanings for more effective human-machine collaboration [99].

One innovative development in this field is Cap (Code as Policies) [47], which advocates for robot-centric language model generation programs. These programs can be adapted to specific layers of the robot's operational stack: interpreting natural language commands, processing perceptual data, and parameterizing low-dimensional inputs for the original language control. The underlying principle of this approach is that layered code generation facilitates the creation of more intricate code, thereby advancing the state-of-the-art in this area.

Both the home cleaning application and the robot-centric language model generation programs in Cap highlight the strengths of LLMs in providing common sense knowledge and interpreting natural language instructions. Traditional robotics often necessitates extensive data collection and specialized model training. In contrast, LLMs mitigate this need by utilizing their extensive training on textual data. The code comprehension and generation abilities of LLMs are particularly crucial, enabling robots to interact more effectively with the physical world and execute complex tasks. However, there is a distinction in application focus: the home cleaning function tends to emphasize everyday tasks and environmental adaptability, whereas Cap centers on programming and controlling the robot's more technical behaviors through Language Model Generation Programs (LMPs).

In summary, the integration of LLMs into robotic reasoning tasks underscores their remarkable capabilities in natural language understanding, common sense knowledge provision, and code comprehension and generation. These features not only alleviate the data collection and model training burdens typically associated with traditional robotics but also enhance robots' generalization and flexibility. With adequate training and adjustment, LLMs can be applied across various scenarios and tasks, demonstrating their vast potential and wide-ranging applicability in the future of robotics and artificial intelligence.

2) Complex Task Reasoning and Decision-making: In the realm of complex task reasoning and decision-making, various studies have leveraged the reasoning abilities of LLMs to augment the refinement of specific downstream tasks. For instance, SayCan utilizes the extensive knowledge embedded in LLMs for concretization tasks alongside reinforcement learning [61]. This method involves using reinforcement learning to uncover insights about an individual's skill value function. It then employs textual labels of these skills as potential responses, while the LLM provides overarching semantic guidance for task completion.

Another notable development is the Instruct2Act framework [100]. It offers a user-friendly, general-purpose robotics system that employs LLMs to translate multimodal commands into a sequence of actions in the robotics field. This system uses policy code generated by LLMs, which make API calls to various visual base models, thus attaining a visual comprehension of the task set.

The usage of LLMs for self-planning and in PDDL (Planning Domain Definition Language) planning has also been explored [101]. It has been shown that LLM outputs can guide heuristic search planners effectively.

In the domain of failure explanation and correction tasks, the REFLECT framework leverages a hierarchical summary of the robot's past experiences generated from multisensory observations to

query an LLM for failure reasoning [102]. The failure explanation obtained can then guide a language-based planner to correct the failure and successfully accomplish the task.

Furthermore, the adaptation of pre-trained multimodal models is a common strategy. By integrating the pre-training of vision-language models with robot data to train VisualLinguistic-Action (VLA) models [62], researchers have found that models trained on internet data with up to 55 billion parameters can generate efficient robot strategies. These models exhibit enhanced generalization performance and benefit from the extensive visual-linguistic pre-training capabilities available on the web.

Socratic Models represent another approach [67], where structured dialogues between multiple large pre-trained models facilitate joint predictions for new multimodal tasks. This method has achieved zero-shot performance across multiple tasks.

In these studies, the primary focus has been on harnessing LLMs for automating reasoning and decision-making processes. This is achieved by leveraging LLMs' capacity to provide or utilize high-level semantic knowledge, thereby enhancing task execution. Some approaches integrate LLMs with other modalities, like vision and action, to deepen task understanding and execution. Others demonstrate effective performance on previously unseen tasks, showcasing zero-shot or few-shot learning capabilities.

Each study adopts a unique approach to integrate LLMs. For example, SayCan incorporates reinforcement learning, whereas Instruct2Act is centered on the direct mapping of multimodal instructions. The techniques employed—ranging from reinforcement learning and heuristic search to multimodal pretraining—vary significantly across different application domains like robot manipulation, planning, and automated decision-making. These studies collectively illustrate LLMs' vast potential in managing complex task reasoning and decision-making. By amalgamating LLMs with other techniques, such as reinforcement learning and multimodal data processing, a deeper semantic understanding and more effective decision support can be achieved. This is particularly evident in robotics and automation, where such integrated approaches are paving the way for novel applications. However, the efficacy of these methods is highly contingent on the specific nature of the task, the data utilized, and the model training approach. Hence, the selection and application of each method must be meticulously tailored to the specific context.

3) Interactive Strategies: The recent advancements in LLMs have significantly contributed to the development of interactive strategies, showcasing impressive capabilities in language generation and human-like reasoning. Matcha [103], utilizing LLMs, enhances interactive multimodal perception, illustrating the potential of LLMs in understanding various types of input data, such as visual and auditory. This approach proposes an augmented LLM multimodal interactive agent. This agent not only leverages commonsense knowledge inherent in LLMs for more plausible interactive multimodal perception but also demonstrates the practical application of LLMs in conducting such perception and interpreting behavior.

Generative agents, as introduced are interactive computational agents designed to simulate human behavior [104]. The architecture of these agents is engineered to store, synthesize, and apply relevant memories, thereby generating plausible behaviors using large language models. The integration of LLMs

with these computational agents facilitates the creation of advanced architectures and interaction patterns. This combination enables more realistic simulations of human behavior, extending the potential applications of LLMs.

The emphasis in LLM-based interactive strategies is on the fusion of LLMs with other perceptual systems, such as image recognition and speech processing. This amalgamation aims to mimic or augment human abilities, enhancing cognitive and processing capabilities. Such advancements have profound implications in the realms of intelligent assistants, robotics, and augmented reality systems.

In the discussed work, a notable emphasis is placed on multimodal perception, focusing on improving the system's ability to understand and interact with its environment. Additionally, the simulation of human behavior seeks to replicate human thought and action processes in AI. The convergence of these two directions holds the promise of creating more powerful and versatile intelligent systems. These systems are envisioned to interact with humans at a more complex and humanized level, presenting significant technical challenges as well as raising crucial ethical and social adaptation questions.

B. Prompt Design

The design of prompts plays a crucial role in querying LLMs. A meticulously crafted prompt, rich in information and structured clearly, yields more precise and consistent outputs aligned with the given instructions. Here we update the text prompt from [114] by incorporating images, creating a multimodal prompt that guides GPT-4V to produce robot task plans. The multimodal prompt consists of five parts:

• System Role Explanation: Specifies the task and the

persona GPT-4V adopts in its responses.

• Predefined Action Pool: A set of predefined robot

actions from which GPT-4V can select and sequence

to complete tasks step-by-step. To address vocabulary

limitations, GPT-4V is prompted to create new actions

if necessary.

• Example Output: An example in JSON format to illustrate the expected output and ensure consistency.

• Case-by-Case Environment Image and Natural Language Instruction: Includes the first frame extracted

from the video demonstration as the environment image.

• Evaluation: GPT-4V is tasked to assess the generated

task plan against the ground truth video demonstration,

scoring the plan based on its alignment with the video

and providing an explanation.

The first three components are input as system messages for each query, while the last two as user messages vary according to the test data. The complete prompt is depicted in Fig. 4 of Appendix.

2 - Parsing Natural Language Sentences into Robot Actions

Abstract. In this paper we propose a Natural Language Processing engine that allows the NAO humanoid robot to execute natural language actions spoken by the user. To this aim we created an ontology that describes body parts, actions, and incompatibilities between actions. The system can work in two modes: stateless and stateful. In stateless mode, the robot returns to its default position after each action. In stateful mode, it performs the actions sequentially and may refuse a command if incompatible with the robot current state. Our system handles compound and multiple expressions that the robot understands and performs.

System Architecture

The Zora robot is completely programmable and the Choregraphe suite7 allows (i) combining and create different behaviours using a visual programming approach, (ii) creating animations by means of an intuitive and dedicated user interface, and (iii) testing behaviors and animations on simulated virtual robots, or directly on the real one. It is equipped with four microphones and can easily record the human voice, which is contextually analyzed and turned into text by a speech recognition module powered by Nuance8. Figure 1 shows the architecture of the system. The user interacts with the robot, which sends his/her input to the cloud through the Internet for the highly expensive computations. In the cloud we have the NLP engine running, which takes a text as input and uses a pipeline of NLP tools and semantic technologies to return a list of commands for Zora to perform. In presence of a partial command (when the user does not indicate some elements in his/her command to identify a unique action), the NLP engine is able to detect and process further those statements. As an example, in the text Hello Zora, now raise your arm, the adjective left or right has not been specified. In such a case, the robot is informed by the NLP engine and interacts with the user by asking whether he/she meant the right or left arm of the robot.

Understanding Natural Language Expressions

After the speech to text, the user's sentence is used as input for the NLP Engine we have developed. The engine uses Stanford CoreNLP9 and returns an RDF that contains the base forms of words, their part of speech (POS), dependency and parse trees.

The first thing performed by the NLP engine is to look for those tokens that have been labeled as verbs by the POS tagger. In addition, it exploits the extracted syntactic dependencies relationships for finding tokens that are linked to the identified verbs (e.g., phrasal verbs) to better capture the semantics of the command. For example in the sentence Zora, now stand up, the robot action is generated by using the extracted phrasal verb stand up. Then, the detected expression is compared with a list containing verbs and synonyms. If a match is detected, then the NLP engine looks for syntactic dependencies to find possible direct object dependency. The list of robot body parts, their related synonyms, and adjectives

(e.g., up, down, left, right) are searched using the object as a key to find a match. When the user indicates plural body parts (e.g., arms, legs), as in Zora, move your hands down, possible adjectives are ignored, as there is no need to specify either left or right. On the other hand, when the identified body part is a hand, arm or leg, the associated adjective (left or right), must be identified to detect one action only to be executed by the robot. For such a purpose, the NLP engine adopts the adjectival dependency of the extracted verb. Therefore, with all this extracted information, the engine can query an OWL ontology that includes the robot body parts and all the related actions.

The ontology10 consists of two parts. The first part includes two main classes, RobotAction and RobotBody. RobotAction includes three subclass: BaseAction, describing all the basic movements that Zora performs with its parts, CompoundAction, which define complex action including multiple basic actions, and SimpleAction defining general movements Zora performs with its entire body such as walking, rotating, lying back, and so on. Hands, arms, legs and all physical parts of Zora that can be involved in basic actions are defined within the RobotBody class. The ontology includes 48 individuals of type RobotBody and RobotAction. The understanding of natural language expressions has been included in the second part of the ontology. The list of alternative keywords (individuals) for all possible actions that Zora can perform and all its body parts are respectively defined within the ActionWord and BodyPartWord classes. For example, the individual sit of the class ActionWord is related to the sit action for the robot and includes different synonym relations to the list of expressions that convey the same meaning (i.e. sit down, seat, and so on).

If the part of the body has not been correctly specified, the robot can ask the user to specify what is missing. For example, after the command Zora, please raise your leg, the robot will reply back asking the user to specify which leg (if left or right) he/she meant. Zora will perform the action only once the user specifies the missing information. There are two modes our system can work: stateless and stateful. In stateless mode, an identified action is performed by the robot which returns then to its default position. In stateful mode, the robot performs the actions sequentially without returning to its default posture and may refuse a command if it is incompatible with its current state.

Let us suppose that the robot is standing on its left leg and the user asks the robot to raise its right leg. Then, the robot will reject the user's request and will warn the user about the incompatibility issue.

The current state of Zora is checked by the NLP engine. The compatibility of the next action to be performed is then checked as well as soon as the current state of the robot is identified. In presence of incompatibility issues, the robot informs the user with a vocal message. All combinations of actions and postures have been tested to generate the list of incompatibilities.

Two subsequent actions were defined as incompatible if they resulted in the robot falling down or being physically unable to perform the second action.

The source code of the NLP engine, the Choregraphe program we developed and the entire list of incompatibilities are publicly available11.

3 - A Review of Natural-Language-Instructed Robot Execution Systems

Abstract: It is natural and efficient to use human natural language (NL) directly to instruct robot task executions without prior user knowledge of instruction patterns. Currently, NL-instructed robot execution (NLexe) is employed in various robotic scenarios, including manufacturing, daily assistance, and health caregiving. It is imperative to summarize the current NLexe systems and discuss future development trends to provide valuable insights for upcoming NLexe research. This review categorizes NLexe systems into four types based on the robot's cognition level during task execution: NL-based execution control systems, NL-based execution training systems, NL-based interactive execution systems, and NL-based social execution systems. For each type of NLexe system, typical application scenarios with advantages, disadvantages, and open problems are introduced. Then, typical implementation methods and future research trends of NLexe systems are discussed to guide the future NLexe research.

Systematic Overview of NLexe Research

Advancements of NLP, such as modeling semantic meanings for complex NL expressions [60,61], interpreting implicit NL expressions by extracting the logic like "I need food" from commands like "I am hungry" [62,63], and enriching abstract NL expressions by associating them with extra commonsense knowledge [64,65], support an accurate task understanding in NLexe. Advancements of robot execution capabilities, such as replicating or representing the action of human [50,66], human intention inference and reaction based on reinforcement learning [67–69], and the ability to understand human's implicit intention with the environment and external knowledge [70,71], support intuitive task execution in NLexe. A Common architecture for an NLexe system includes three key parts: instruction understanding, decision-making, and knowledge-world mapping. In instruction understanding, the user's verbal instructions were acquired by sensors and processed by speech recognition and language understanding systems to perform a comprehensive semantic analysis. In the decision-making phase, various algorithms such as deep neural networks were used to collect and construct the task-related knowledge using information from the robot knowledge base or NL knowledge and supported robot decision-making in various manners. Through the process of knowledge-world mapping, information patterns were accurately contextualized within real-world scenarios, allowing for the resolution of incomplete knowledge gaps. This facilitated the successful execution of the NLexe process. With supporting techniques from both NLP and robot execution, NLexe has been developed from a low-cognition-level symbol matching control, such as using "yes" or "no", to control robotic arms, to a high-cognition-level task understanding, such as identifying a plan from the description "go straight and turn left at the second cross". Combining the advancements of both NLP and robot executions improves the effectiveness of both communication and execution in NLexe.

As a result of NLexe research, a substantial number of NLexe projects were launched, including the following. (1) "collaborative research: jointly learning language and affordances" from Cornell University, which interprets objects by natural-language-described affordances, such as 'water: drinkable; cup: containable', to help robot with its manipulation [72]; (2) "robots that learn to communicate with human through natural dialog" from University of Texas at Austin, which enables robots to directly learn task executions from NL instructions with a user-friendly manner [73]; (3) "collaborative research: modeling

and verification of language-based interaction" from MIT, which uses NL to interpret human interactions and understand human perspective in physical world for finally integrating humans and robots [74]; (4) "language grounding in robotics" in University of Washington, which maps theoretical knowledge, such as object-related descriptions, to practical sensor values, such as sensor-captured object attributes [75]; (5) "semantic systems" from Lund University, which uses NL to describe industrial assembly processes [76]. NLexe research is frequently disseminated through prestigious international journals, including International Journal of Robotics Research (IJRR), IEEE Transactions on Robotics (IEEE TRO), The Journal of Artificial Intelligence (AIJ) and IEEE Transactions on Human-machine Systems (IEEE THMS), and international conferences, like International Conference on Robotics and Automation (ICRA), International Conference on Intelligent Robots and Systems (IROS) and AAAI Conference on Artificial Intelligence (AAAI). By using the following keywords, NLP, human, robot, execution, speech, dialog, and natural language, human-robot interaction, HRI, social robotics, about 4410 papers were retrieved from Google Scholar. Then with a focus on NL-instructed robot executions, about 1390 papers were finally kept for plotting the publication trend, shown in Figure 2. The steadily increasing publication numbers demonstrate the increasing significance of NLexe research.

NL-Based Execution Control Systems Using Execution Logic

To flexibly adjust execution plans, NL-based execution control systems using execution logic were designed, in which hierarchical logic is verbally specified to define execution preconditions and procedures. In this system, speech recognition and logic level NL understanding are required, which brings the difficulty of accurately understanding the coordination of consecutive instructions. This system is designed for tasks with an ordered execution or a conditional execution. Typical tasks include instructing with logic words like "turn left, then go forward", where robots figure the order and turn left to go straight. Despite the improvements in flexibility brought about by the integration of semantic correlation in NL execution mapping, control performance in dynamic situations remains limited due to the neglect of control logic within these semantic correlations. This oversight renders robots unable to adapt to environmental changes and hampers their ability to intuitively reason about execution plans. For instance, the NL instruction "fill the cup with water, deliver it to the user" inherently includes the logical sequence: "search for the cup, then use the water pot to fill the cup, and finally deliver the cup". Ignoring this logic during the control process can lead to incorrect executions, such as "use the water pot to fill the cup, then search for the cup", or can restrict proper executions, such as "deliver the cup, then use the water pot to fill the cup", in dynamic environments. This results in the incorrect removal or addition of execution steps. To address this issue, the study explores logic correlations—encompassing temporal logic, spatial logic, and ontological logic—among controlling symbols to enhance the adaptability of NL-based execution control methods.

Typical NLexe systems that employ NL-described logic relations include the following: (1) Using NL commands, such as "if you enter a room, ask the light to turn on and then release the gripper", to modify the robot trajectory in different situations with self-reflection on task executability [99–101]. (2) Designing robot manipulation posts based on fuzzy action type and speed requirements, exemplified by commands like "move up, then move down" [88,94]. (3) Serving meals by considering object function

cooperation logic, such as “foodType—vesselShape” relation [82,102]. (4) Assembling industrial parts by considering assembly logic, such as “first pick part, then place part” [81,103].

To scale up robot knowledge for executing multiple tasks under various working environments, NL was used to train a robot. During the training, knowledge about task execution methods was transferred from a robot or a human expert to targeted robots. An NL-based execution training system uses human knowledge in the form of NL to train an inexperienced robot for task understanding and planning. In NLexe training systems, with the support of speech recognition technologies, robots have the capability of initial language understanding based on interpreting multiple sentences. User instructions include simple words like the name and property of an object and the action of robot motion and grasping. Human users give verbal descriptions, and robots extract information and task steps as well as their meaning according to the way instructions are interpreted. A robot also proactively asks its human users questions for object or action disambiguation, bringing the instruction complexity to a higher level since the robots need both language understanding and generation during training. Typical NL-based execution training systems are shown in Figure 5. Human commonsense knowledge was organized into executable task plans for robots with consideration of the robot’s physical capabilities, such as force, strength, physical structure, and speed, human preferences, such as motion and emotion, and realworld conditions, such as object availability, object distributions, and object locations. With the executable knowledge, a robot’s capabilities in task understanding, environment interpretation, and human-request reasoning were improved. Different from NL-based execution control, where robots were not involved in advanced reasoning, in NL-based execution training, robots were required to reason about human requirements during human-guided executions.

To teach a robot with high-level execution plans, execution training systems that only use verbal instructions were designed. In this way, robots learn from these instructions without any physical participation or help from their users. In this type of system, speech recognition, NL understanding, learning, and reasoning mechanisms are required, which brings the difficulty of transforming high-level human instructions to real-world implementation. They are designed for training robots on how to execute certain tasks using command sequences. Typical tasks include training robots with action combinations like “lift, then move to location”, where robots learn this plan and execute it in the future. During the high-level knowledge grounding, human cognition processes on task planning and performance were modeled. The advantage of the training using human NL instructions is that a robot’s reasoning mechanisms during NLexe are initially developed; the disadvantage is that the execution methods directly learned from the human NL instructions are still abstract in that the sensor-value-level specifications for the NL commands are lacking, limiting the knowledge implementations in real-world situations.

Given that human verbal instructions can deliver basic information about task execution methods, NL-based execution training initially started with defining simple commonsense by using human NL instructions, enabling a robot with initial reasoning capability during execution [111]. Typical NLexe execution training systems using human NL instructions include (1) daily assistive robots with NL-trained object identities, such as “cup, mug, apple, laptop” [102,112]; (2) robots with NL-trained object physical

attributes, such as “color, shape, weight” [107,113,114]; (3) robotic grippers with NL-trained actions, such as “grasp, move, lift” [115,116].

Instead of modeling correlations among task execution procedures, the knowledge involved in the low-level reasoning was merely piecemeal with only separate knowledge entities, such as “cup, cup color, grasp action”. Piecemeal knowledge enables robots with a shallow understanding of motivations and logic in task executions. As information and automation techniques improved, the low-level reasoning method was then evolved into a high-level reasoning method, in which complex NL expressions were grounded into hierarchical knowledge structures for motivation and logical understanding. With hierarchical knowledge, NLexe systems were enabled to learn complex task executions. Typical NLexe systems include the following. (1) Industrial robot grippers with NL-trained object grasping methods, such as “raise, then move-closer” [117]. (2) Executing tasks in unfamiliar situations with NL-trained spatial and temporal correlations, such as “landmark—trajectory— object, computer—on—table, mouse—leftOf—computer” [118–121]. (3) Daily assistive robots with NL-trained object delivery methods, such as “moved forward → moved right → moved forward” [122,123].

4 - Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming

Abstract. Industrial robots are applied in a widening range of industries, but robot programming mostly remains a task limited to programming experts. We propose a natural language-based assistant for programming of advanced, industrial robotic applications and investigate strategies for domain-specific fine-tuning of foundation models with limited data and compute.

Methods

As a testbed, we use the ArtiMinds Robot Programming Suite (RPS), an integrated development environment (IDE) for industrial robots. With it, robots are programmed using parameterizable robot skills (called “templates”) such as “Grasp”, “Insert”, etc. The resulting robot program is a tree of parameterized templates, which is compiled into executable code. The assistant answers questions of three types:

Providing high-level explanations of templates. Given a question like “What does a ‘Move to State’ template do?”, it should answer something like “It moves the end-effector on a collision-free trajectory to a goal specified in configuration space.” The complexity of answers range from simple, as illustrated here, to highly complex, particularly for force-controlled skills whose behavior depends in part on interactions with the environment.

Providing examples for the usage of templates. To a question like “When should I use a ‘Move to State’ template?”, it should answer “A ‘Move to State’ template can be used to efficiently move the robot arm through environments in which collisions can occur.” Examples can involve concrete application domains, such as painting, gluing or welding for a “Path Loader” template.

Providing step-by-step explanations of expected robot behavior. To a question like “What motions will a robot make when executing a ‘Grasp’ template?”, the assistant should answer “It will execute a collision-

free approach motion, open the gripper, move closer to the object, close the gripper, and depart with a collision-free motion." Step-by-step explanations provide useful information for very high-level templates such as "Peg in Hole".

We investigate how LLMs can be fine-tuned under the compute and data constraints typical for industrial small- or medium-sized enterprises (SMEs). This implies limitations to a single server-grade graphics processing unit (GPU), e.g., Nvidia A100, with 80GB VRAM. Traditional fine-tuning requires up to 780GB given 65B parameters [2]. To reduce model size, we leverage QLoRA adaptation with 4-bit NormalFloat quantization and double quantization [2]. We investigate three data sparse training regimes: Fine-tuning an instructionfollowing model on domain-specific data, which has been pretrained in different domains; fine-tuning a streaming model for instruction-following on domainspecific data; and fine-tuning a streaming model on a domain-specific streaming data, succeeded by domain-specific instruction-following training. Comparing these training regimes determines whether the use of general-purpose pre-trained models provides useful priors to simplify fine-tuning. Moreover, we investigate the impact of prefix fine-tuning [3] on model quality.

To obtain feedback about real-life performance, we conduct a survey with domain experts. The evaluation set contains 40 instructions, some of which outside the question types seen in training (e.g., "What is the difference between a 'Move to Point' template and a 'Move to State' template?"). A total of 33 engineers participated in the survey, each completing at least one questionnaire containing 10 randomly assigned prompts and model responses. Each prompt-response pair was evaluated on factual correctness ("The answer is precise and factually correct. (yes/no)"), domain adherence ("The answer remains in the domain of ArtiMinds RPS. (yes/no)") and perceived helpfulness ("Rate the helpfulness of the response on a scale from 1 to 5."). The results are shown in Table 1. Echoing the BERTScore results, Alpaca models performed best with respect to correctness, domain adherence and helpfulness. It must be noted, however, that all trained models have noticeable shortcomings with respect to correctness and overall helpfulness. The LLaMA-based models in particular struggled with sentence structure, often repeating words and predicting unknown tokens. In contrast, Alpaca-based models generated responses with satisfactory format. Prefix fine-tuning did not have a discernible impact on model performance.

5 – RT-Grasp: Reasoning Tuning Robotic Grasping via Multi-modal Large Language Model

The growth of artificial intelligence in recent years has been significantly driven by the emergence of large language models (LLMs). These models, packed with vast knowledge and advanced reasoning ability, have revolutionized our approach to various tasks, especially those involving language processing. In robotics, LLMs play a crucial role in facilitating direct interactions between robots and humans. For instance, in tasks such as robot manipulation planning, many studies [1], [2], [3] have utilized LLMs to interpret natural language commands from users and translate them into feasible multi-step plans for robots. However, despite their potential in robotics, LLMs' application has predominantly been limited to such planning tasks. A notable bottleneck lies in the textual nature of LLM outputs, which often pose challenges for tasks requiring precise numerical outputs. Recently, multi-modal LLMs have expanded LLM capabilities by understanding both text and images. In robotics, they bridge the gap between

perception and planning, addressing a variety of embodied reasoning tasks [4], [5]. However, their image understanding lacks precision, for

In summary, our work focuses on adapting multi-modal LLMs for numerical prediction tasks, specifically in the domain of robotic grasping. In contrast to deterministic traditional methods, our approach not only incorporates advanced reasoning capabilities but also introduces a novel paradigm for refining predictions, as illustrated in Fig. 1. The main contributions can be summarized as follows

Traditional robotic grasping methods typically rely on deterministic predictions, which often fail in real-world scenarios due to their lack of reasoning capabilities. Most existing methods [7], [8], using CNN-based architectures, excel in experimental accuracy on benchmark datasets, but struggle in practical applications. For example, these traditional models may produce theoretically correct predictions that prove impractical in execution, as shown in Fig. 1 and labeled as invalid. Such predictions are hard to apply across robot arms due to varying gripper constraints. Additionally, some theoretically correct grasps may result in unsafe actions, such as targeting the sharp ends of screwdrivers during grasping. Hence, adopting a non-deterministic approach equipped with reasoning ability is crucial. This capability not only allows the model to generate practical grasp poses applicable across various settings but also allows the refinement of predictions based on user commands. Here a question is posed: can the reasoning capabilities inherent in LLMs be utilized for numerical prediction tasks in robotics? This paper offers a positive answer, showcasing an adaptation of multi-modal LLMs to robotic grasping tasks

IV. RT-GRASP In this section, we introduce Reasoning Tuning for robotic grasping (RT-Grasp), a novel method designed to bridge the gap between the inherent text-centric nature of LLMs and the precise numerical requirements of robotic tasks. Its primary objective is to facilitate multi-modal LLMs for numerical prediction by leveraging their extensive encapsulated prior knowledge. A pre-trained multi-model LLM, such as LLaVA [37], can be directly fine-tuned in a fully supervised manner when given the image and the text instruction. The model is trained by predicting each token in the text output sequentially. The proposed Reasoning Tuning introduces a structured text output, which includes a reasoning phase and a subsequent numerical prediction. We created our image-text dataset for robotic grasping, named Reasoning Tuning VLM (Visual Language Model) Grasp dataset, used for fine-tuning multimodal LLMs. Additionally, we introduce a method that automatically generates such image-text datasets using GPT3.5 [38], which can be applied to datasets for tasks beyond robotic grasping. Further details are presented in Section IV-A. Furthermore, we discuss two cost-efficient training strategies employed in our experiments in Section IV-B

reasoning_templates = { [“The image shows a baseball, which is spherical. Grasping it as its center will ensure optimal balance.” , “The object is a baseball. Its round shape requires a center grip for stability.” , …] , [“The object is a pair of sunglasses, with lenses and a frame. Grasping the frame, away from the lenses, is safest.” , “Recognized sunglasses. It's essential to grasp its frame while avoiding the delicate lenses.” , …] , “baseball”: “sunglasses”: [“The object is a cup, which is generally cylindrical. An secure way to grasp a cup is by its edge from the top and the rotation angle should be such that the gripper is orthogonal to the edge of the cup.” , “Recognized object is a cup. Its best grasped by targeting its top edge, ensuring the gripper is perpendicular to its circular opening.” , …] , “cup”: … } “object category”: [

“Description of the object type and general shape. Grasping strategy.”, …] Fig. 4. Examples of reasoning templates within the Reasoning Tuning VLM dataset.

This research underscores the potential of LLMs beyond their conventional text-centric applications. Our proposed method utilizes the extensive prior knowledge of LLMs for numerical predictions, specifically in robotic grasping. Through comprehensive experiments conducted on both benchmark datasets and real-world scenarios, we have demonstrated the efficacy of our approach. For future work, we plan to extend the validation of our method by applying it to grasping datasets featuring a broader array of objects, such as the Jacquard dataset [42]. Moreover, the adaptation of multi-modal LLMs for numerical predictions in other robotic manipulation tasks is also a promising research direction

6 – ARRC: Advanced Reasoning Robot Control—Knowledge-Driven Autonomous Manipulation Using Retrieval-Augmented Generation

Abstract— We present ARRC (advanced reasoning robot control), a practical system that connects natural language instructions to safe, local robotic control by combining RetrievalAugmented Generation (RAG) with RGB–D perception and guarded execution on an affordable robot arm. The system indexes curated robot knowledge (movement patterns, task templates, and safety heuristics) in a vector database, retrieves task-relevant context for each instruction, and conditions a large language model (LLM) to synthesize JSON-structured action plans. These plans are executed on a UFactory xArm 850 fitted with a Dynamixel-driven parallel gripper and an Intel RealSense D435 camera. Perception uses AprilTags detections fused with depth to produce object-centric metric poses; execution is enforced via a set of software safety gates (workspace bounds, speed/force caps, timeouts, and bounded retries). We describe the architecture, knowledge design, integration choices, and a reproducible evaluation protocol for tabletop scan/approach/pick–place tasks. Experimental results are reported to demonstrate efficacy of the proposed approach. Our design shows that RAG-based planning can substantially improve plan validity and adaptability while keeping perception and low-level control local to the robot.

Abstract— We present ARRC (advanced reasoning robot control), a practical system that connects natural language instructions to safe, local robotic control by combining RetrievalAugmented Generation (RAG) with RGB–D perception and guarded execution on an affordable robot arm. The system indexes curated robot knowledge (movement patterns, task templates, and safety heuristics) in a vector database, retrieves task-relevant context for each instruction, and conditions a large language model (LLM) to synthesize JSON-structured action plans. These plans are executed on a UFactory xArm 850 fitted with a Dynamixel-driven parallel gripper and an Intel RealSense D435 camera. Perception uses AprilTags detections fused with depth to produce object-centric metric poses; execution is enforced via a set of software safety gates (workspace bounds, speed/force caps, timeouts, and bounded retries). We describe the architecture, knowledge design, integration choices, and a reproducible evaluation protocol for tabletop scan/approach/pick–place tasks. Experimental results are reported to demonstrate efficacy of the proposed approach. Our design shows that RAG-based planning can substantially improve plan validity and adaptability while keeping perception and low-level control local to the robot.

In this paper, we bridge these gaps by introducing a RAG-enabled robotic manipulation pipeline–called ARRC (advanced reasoning robot control)–that unifies perception, retrieval, and safe plan execution. We deploy this system on a UFactory xArm 850 equipped with a RealSense D435 and a Dynamixel gripper, integrating retrieval of robot-centric safety heuristics and procedural templates at inference time. This design offers both adaptability and reliability, enabling the injection of new task knowledge or safety rules without retraining. Our contributions are as follows: • We develop a hybrid VLA architecture integrating RAG for dynamic injection of procedural and safety knowledge. • We demonstrate a real-world implementation with RGB-D perception, JSON-structured plan generation, and strict execution safety gates. • We propose a reproducible ablation protocol to quantify contributions of retrieval, vision gating, and safety checks.

II. SYSTEM OVERVIEW A. Perception AprilTags, combined with depth data from the Intel RealSense D435, provide marker-based detections that are fused to recover metric 3D poses in the robot frame. With sub-pixel accuracy, AprilTags enable robust and precise pose estimation, making them well-suited for manipulation tasks [19]. B. Retrieval & Planning We construct a curated robotics knowledge base comprising movement primitives, templates, safety heuristics, short demonstration transcripts, and parameterized affordances. This knowledge base is embedded using a SentenceTransformers model and indexed in ChromaDB (or a FAISS index). At inference time, the retrieval module selects the topk relevant context snippets, which are concatenated with the current observation summary and provided to an LLM (e.g., Gemini, PaLM-E-style models). The LLM then generates a structured JSON plan, enabling downstream execution [20]. C. Execution The JSON plan—represented as a sequence of named actions with bounded parameters—is first validated by a plan checker, synchronized with the latest object observations, and then executed through the XArm Python SDK. Execution is safeguarded by software safety gates, including workspace and joint limits, capped Cartesian speeds and accelerations, gripper torque/time gating, per-step timeouts, and bounded retries. Low-level controllers and perception remain local to the robot, while the LLM planner can be configured to run either locally (on-device) or via cloud APIs. IV. PROPOSED METHOD In this section, we detail each system component and our design choices.

D. Plan Representation, Validation, and Safety Plans follow a restricted JSON schema: { "goal": "place bottle on tray", "steps": [{"action":"SCAN_AREA", "params":{...}}, {"action":"APPROACH_OBJECT", "params":{"label":"bottle", "hover_mm":40}}, {"action":"MOVE_TO_POSE", "params":{"xyz_mm":[...], "rpy_deg":[...]}}] } Before execution, each action step is validated against system constraints, including parameter bounds such as speeds and positions, and overall plan length. Steps that require current environmental information are synchronized with the latest perception data, and high-risk operations may optionally include human-in-the-loop confirmation. During execution, the executor enforces runtime safety through mechanisms such as per-step timeouts, gripper aborts triggered by load or duration limits, and emergency retreat in response to repeated failures.

We have presented a practical, retrieval-augmented manipulation system that integrates local RGB–D perception, a vector-indexed robotics knowledge base, and LLM-driven plan synthesis to translate natural language instructions into validated, executable actions on a UFactory xArm 850. Experimental results demonstrate that retrieval-augmented generation improves plan specificity by providing concise, robot-relevant context, enabling more reliable and parameterized plans compared to LLM-only

approaches. Furthermore, keeping perception and low-level control local proves effective for real-time execution and safety enforcement, while latency introduced by retrieval and LLM inference can be mitigated through capturing strategies. Despite these strengths, the current system has some limitations. The knowledge base is curated and static, limiting adaptability to lifelong updates. Tasks requiring tactile feedback or precise torque control, such as dowel insertion or screw driving, remain out of scope. Additionally, the LLM planner may occasionally generate physically infeasible actions, highlighting the necessity for robust plan validation and symbolic feasibility checks. These limitations underscore important avenues for future research. Future work will focus on scaling the system to more complex manipulation scenarios, including multi-arm coordination, large-scale benchmarking across diverse objects and lighting conditions, and integration of tactile sensing and force-aware motion primitives. On-device acceleration of retrieval and LLM inference is another promising direction to reduce latency and improve autonomy. Overall, our study demonstrates that combining retrieval-augmented language reasoning with local perception and safety-guarded execution provides a practical and reproducible pathway toward robust, adaptable robotic manipulation in real-world environments

D. Safety-Constrained Execution The execution module enforces a set of safety constraints to guarantee reliable operation during plan execution. First, all commanded target positions are validated against the robot's reachable workspace, ensuring all Pa ∈ W. Motion commands are further regulated through velocity limits: Cartesian translations satisfy $\|\dot{P}\| \leq v_{max}$, while joint velocities are constrained by $|\dot{\theta}_i| \leq \dot{\theta}_{i,max}$. Each action step is assigned a maximum allowable duration tmax; exceeding this limit triggers an abort and initiates a return-to-safe configuration. Gripper operations are protected through realtime load monitoring, with an emergency release executed if excessive force or unexpected obstruction is detected. By integrating workspace validation, dynamic constraints, temporal bounds, and force-aware gripper control, the execution framework achieves robust, safe deployment of RAGgenerated plans without sacrificing real-time responsiveness or hardware longevity.

7 - ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning

Abstract We present a framework for intuitive robot programming by non-experts, leveraging natural language prompts and contextual information from the Robot Operating System (ROS). Our system integrates large language models (LLMs), enabling non-experts to articulate task requirements to the system through a chat interface. Key features of the framework include: integration of ROS with an AI agent connected to a plethora of open-source and commercial LLMs, automatic extraction of a behavior from the LLM output and execution of ROS actions/services, support for three behavior modes (sequence, behavior tree, state machine), imitation learning for adding new robot actions to the library of possible actions, and LLM reflection via human and environment feedback. Extensive experiments validate the framework, showcasing robustness, scalability, and versatility in diverse scenarios, including long-horizon tasks, tabletop rearrangements, and remote supervisory control. To facilitate the adoption of our framework and support the reproduction of our results, we have made our code open-source. You can access it at: ROS-LLM-Code.

As we conceptualize more tasks, we subsequently develop more and more atomic actions that we can curate in a library. Each atomic action is ultimately a function: an input variable and parameters that are mapped to a return value. The representation of an atomic action could be simple, for instance, “open gripper” that sends an “open” signal to a parallel gripper attached to a robot arm which then returns the final width from the gripper. The representation could take other forms, for example a planner and feedback controller, a neural network policy trained using RL or imitation learning for the end-effector that feeds into an inverse-kinematic controller to compute target joint states. The planner/controller parameters for a given formulation or the neural network weights can be stored in memory and considered as the atomic action representation in the atomic action library. This library of atomic actions can be in the form of a code API or, if the Robot Operating System (ROS) is being used, the library can be a list of ROS actions and services. Textual descriptions for each atomic action are always assumed to be provided, i.e. documentation.

In general, robot systems are designed in a modular way to allow users to build their own frameworks by easily integrating and modifying existing processes. The most well-known framework used in research and industry is the Robot Operating System (ROS) [41, 32]. Other examples include the Lightweight Communications and Marshaling (LCM) project [16] and the Open Robot Control Software (Orocos) project [5]. The ROS framework provides a well-established ecosystem of packages and libraries that are ready to use and integrated with many common robot systems (e.g. Universal Robots, Robotiq, Clearpath Robotics). Some of most widely used ROS libraries and packages include the TF library [11], MoveIt [9], and the Navigation-stack. The ROS ecosystem of contributors additionally contains many packages for many important requirements such as simulation [26, 34], kinematic modeling [47, 2], and planning and control [19, 36, 8]. Thus, ROS offers many packages providing useful functionalities for both research and commercial applications. These packages include valuable data structures, control interfaces, inverse kinematics (IK) and motion planning tools, perception utilities, and various visualizers. Additionally, with new tools such as the BehaviorTree.ROS library, ROS actions and services enable the generalization of a wide variety of capabilities required by robot systems into a unified execution framework.

As described in the introduction, we can think of an atomic action as a single task that the system can perform. Formally, we frame a single task as a Markov Decision Process (MDP) characterized by the tuple $\langle S, A, r, P, \gamma \rangle$ where S is the state space, A is the action space, $r : S \times A \rightarrow R$ is a reward function defined for any state $s \in S$ and any action $a \in A$, $P(s_{t+1}|s_t, a_t)$ is a transition probability distribution, t is a discrete time step, and the scalar $0 < \gamma \leq 1$ is a discount factor. In contrast, to the standard MDP formulation, we also assume access to a failure flag f that is returned on termination of the MDP, i.e. task completion. The failure flag indicates whether the desired task was completed successfully or not, i.e. f = 0 indicates success, f = 1 otherwise. For example, if the task is for a robot arm to reach to a target, then at termination f = 0 means the target was acquired, and f = 1 implies that the robot finished in a configuration far from the target. Thus, our modified MDP is denoted by $\langle S, A, r, f, P, \gamma \rangle$. In single-task RL, an AI agent generally learns a policy $\mu(a|o)$ choosing an action a given an observation o (e.g. from an image) of the state s. The agent’s objective is to determine the sequence of actions that maximizes the expected return $E [P_t \gamma_t r(s_t, a_t)]$. The specific task and its associated rewards are determined by the reward function r. In our case, we assume access to an atomic action library that corresponds to having a

set of N pre-trained or pre-defined policies {μbi} N i=1 ready to use, each based on an underlying modified MDP. Note, we use hat μb to denote that the policy is trained/defined. Humans exhibit remarkable proficiency in synthesizing complex behaviors by composing various known skills. With regard to a robotic system, this process involves the composition of distinct policies that are executed following a certain structure, such as some appropriate sequential order, or corresponding to a data structure like a behavior tree. In situations when there is access to a team of experts, it is reasonable to assume that they are capable to define some reward function that measures appropriate compositions of atomic actions, i.e. behaviors. With access to experts, we can reasonably assume some reward or fitness function that specifies an appropriate composition of these policies, in which case we could explore methods based on hierarchical reinforcement learning [1]. However, in our case, absent expert guidance, the robotic system must rely on environmental observations o and non-expert human input h to guide policy selection. Also, we assume the input from the human is given by text.

3.3 Human non-expert interface We provide a chat-based interface to our framework to allow easy adoption from non-expert human users. Each environment step is executed after the human feedback is received from the interface, and then once the execution is over on the system (ending either with a success or a failure), we ask the human to input a new textual entry. At the beginning, we let the system interpret the first human input as the task description. The task description should outline the goal or objective to assign to the robot, providing context for the subsequent actions to generate. Thereafter, the system treats human input prompts as a feedback, which may contain suggestions for corrective behavior or suggestions for alternative approaches for the robot to complete the task. Another potential interaction mode could be via speech, which would have the potential to be even more intuitive for non-experts. We actually plan to implement a microphone into our setup and use an off-the-shelf audio-to-text package for parsing the input. This functionality will be incorporated into our main code-base in the future.

3.4 Prompt generation The prompt provided to the language model serves as input to generate behavior representations that can be executed on the system. At each environment step τ , the prompt is updated, ensuring that the language model receives the latest information necessary for decision-making. We show in the central part of Figure 2 the different elements that we expect in a prompt to shape the behavior of the system. The prompt includes a task description that is provided by the user, as described in the previous sub-section. After the first environment step τ , the non-expert provides feedback that the system uses to correct its behavior. A description of the atomic action library is also included to provide context on the admissible behaviors of the system, as described in Section 3.1. Moreover, the prompt contains an observation of the environment that is collected by mapping several sensor readings to text. Several well-known prompt engineering strategies are utilized to aid the language model construct a behavior for the system, namely chain-of-thought and few-shot prompting. These additional portions of the prompt are assumed to be given as part of all observations o. Finally, some additional notes are written in the prompt, such as how the language model should format the behavior output (e.g. Python or XML). Overall, the prompt generation process gathers information from both the ROS environment and the human interface, ensuring that the language model receives comprehensive input to guide its decision-making process. Once the prompt is constructed, it is then

passed to the language model. We consider the output of the language model to represent the desired behavior for the system. We describe next the different formatting options for the output of the language model. 3.5 Behavior representation We call a behavior the combination of atomic actions that is extracted from the textual output of the language model. To represent a behavior, the LLM generates either a Python or XML code. When Python format is used, a Python terminal exposed to the ROS environment executes the LLM output. In the case of XML, the LLM response is interpreted as a behavior tree1 . We use regular expressions to easily identify parse LLM output that should encapsulate the code into ‘‘‘python...’’’, ‘‘‘json...’’’, or ‘‘‘xml...’’’. Python output We expect the use of Python code when the action library is a set of Python function that exposes the various functionality of the system. The library can also contain ROS actions and ROS services that can be interfaced with the script. JSON output When the JSON format is used, a behavior representation called an action sequence is used. In this case, the specified actions in the sequence are executed one after the other, and we expect each action to be a ROS service of the type rosllm_srvs/AtomicAction. This service returns a string called output and takes as input one string argument called input, which takes input per action, and another string argument prev_output, which is the output from the previous action. XML output A behavior tree, represented by XML code, is a hierarchical model that describes the behavior of autonomous agents or robots. It consists of nodes that define specific actions or conditions, and it is organized in a tree-like structure. At the root of the tree, the behavior selector node determines the order in which to evaluate and execute child nodes. These child nodes can include sequences of nodes, which execute their child nodes sequentially until one fails, or parallel nodes, which execute their child nodes simultaneously. Other types of nodes include conditional nodes, action nodes, and decorator nodes, each serving distinct roles in controlling the agent’s behavior. In this case, the language model is tasked with producing the XML code defining a behavior tree.

the capability of our framework to adapt and recover from such changes through continual learning facilitated by human feedback. The primary goal is to showcase the system’s ability to utilize human feedback to recover from unforeseen disruptions in task execution and subsequently learn from these experiences to handle similar changes in the future autonomously. The experimental protocol is as follows. 1. Task Specification: The system is tasked with “pick and place the box,” a common robotic manipulation task. 2. Action Sequence Generation: Using the provided task description, the language model generates an action sequence for execution. 3. Execution and Disruption: As the robot executes the task, the environment is intentionally perturbed by moving the target box, leading to a failure to grasp the object. 4. Human Feedback: Upon observing the failure, the human provides corrective feedback, advising the system to ensure the box’s proximity before grasping and then to retry the task. 5. Recovery and Adaptation: Leveraging the feedback, the system adapts its approach and successfully completes the task, demonstrating its ability to recover from environmental changes through human-guided learning. To evaluate the system’s capability for continual learning, a second trial is conducted under identical conditions, with the box moved simultaneously as in the previous trial. However, this time, the human feedback from the initial trial is incorporated into the task prompt. By doing so, we aim to assess whether the system autonomously applies the learned corrective action to handle similar environmental changes without human intervention

8 Conclusions In this section, we provide an overview of our main conclusions, acknowledge the current limitations of our framework and highlight potential future directions.

8 - ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning

The human ability to learn, generalize, and control complex manipulation tasks through multimodality feedback suggests a unique capability, which we refer to as dexterity intelligence. Understanding and assessing this intelligence is a complex task. Amidst the swift progress and extensive proliferation of large language models (LLMs), their applications in the field of robotics have garnered increasing attention. LLMs possess the ability to process and generate natural language, facilitating efficient interaction and collaboration with robots. Researchers and engineers in the field of robotics have recognized the immense potential of LLMs in enhancing robot intelligence, human-robot interaction, and autonomy. Therefore, this comprehensive review aims to summarize the applications of LLMs in robotics, delving into their impact and contributions to key areas such as robot control, perception, decision-making, and path planning. We first provide an overview of the background and development of LLMs for robotics, followed by a description of the benefits of LLMs for robotics and recent advancements in robotics models based on LLMs. We then delve into the various techniques used in the model, including those employed in perception, decision-making, control, and interaction. Finally, we explore the applications of LLMs in robotics and some potential challenges they may face in the near future. Embodied intelligence is the future of intelligent science, and LLMs-based robotics is one of the promising but challenging paths to achieve this.

In conclusion, the applications of large language models in robotics hold tremendous potential. They provide new paradigms and methods for robot control, path planning, and intelligence. Through more intuitive and natural human-machine interaction, language-based path planning, and intelligent semantic understanding, large language models not only enhance the performance and efficiency of robots but also improve the experience and interaction modes of human-robot interaction. Therefore, this comprehensive review aims to summarize the applications of LLMs in robotics, delving into their impact and contributions to key areas such as robot control, perception, decision-making, and path planning. To summarize, there are four key contributions in this paper, as follows: • We discussed the latest advancements in LLMs and their significant impact on the field of robotics. We highlighted the benefits of LLMs for robots, as well as the emergence of new robot models equipped with LLMs in recent years. • We discussed the current state of robot technology, focusing on advancements in perception, decisionmaking, control, and interaction combined with LLMs. Specifically, we highlighted the critical role of LLMs in decision-making modules, which have enabled robots to make more informed and effective decisions in various applications. • We explored potential applications of current robots equipped with LLMs in the near future. • We discussed several potential challenges that robots may face when integrated with LLMs, as well as the potential impact of future developments in this field on human society.

affordance function. It will perform the most likely action to succeed in the current environment and state. For instance, upon receiving the instruction, “Can you help me get an apple?". LLM may

decompose it into several tasks: “walking to the kitchen, opening the refrigerator, obtaining the apple, and delivering it to the requester.", just like in Figure 2(a). 2.3.2. PaLM-E While LLMs have demonstrated remarkable capabilities in handling complex tasks, integrating them as an interface into robots remains a significant challenge. A major limitation of LLMs is their reliance on text input, which is insufficient for robots that require physical interaction. PaLME [34] boasts an LLM capable of integrating continuous sensory information from the real world, effectively bridging the gap between language and perception. Its multi-modal input encompasses vision, text, and state estimation, like in Figure 2(b), as exemplified by the question "What is it in ?" The model’s processing is end-to-end, whose performance is state-of-the-art in OK-VQA [84]. PaLM-E is a visual-language generalist. PaLM-E treats images and text as multi-modal inputs represented by latent vectors. PaLME is a decoder-only model that generates text completions autonomously when provided with a prefix or hint. The output of PaLM-E is separated into two parts: when tackling text generation tasks (such as embedded question answering or scene description), the model directly produces the final output (i.e., output text or speech). In contrast, when utilized for specific planning and control tasks, PaLM-E generates low-level instruction text (e.g., instructions for controlling robot meta-actions). 2.3.3. LM-Nav Goal-based robot navigation can leverage large, unlabeled datasets for training, resulting in strong generalization capabilities in real-world scenarios. However, in visionbased settings, specifying targets often requires images. Current supervised learning methods are not only expensive but also demand linguistically described and labeled trajectory data, making them impractical for widespread use. How can users communicate with robots more conveniently? To address the challenge, LM-Nav [117] was developed, exploiting the advantages of language to facilitate effective communication between users and robots. The LM-Nav system comprises three components: a vision-navigation model (VNM); a visual-language model (VLM); and a large-scale language model (LLM). Notably, LM-Nav operates without the requirement of labeled data or fine-tuning. By leveraging the VLM and VNM, LM-Nav can extract landmark names from commands and navigate to specified locations. LMNav leverages three pre-trained models to achieve successful navigation in pre-explored environments. First, it employs ViNG [114] as a VNM creates a topological map using observations from a prior exploration of the environment. Subsequently, GPT-3 [27] serves as the LLM [14] processes free-form text instructions to determine the target landmark. Finally, CLIP [99] serves as the VLM to locate the corresponding position in the topology map based on the identified landmark. By combining these models, LMNav can effectively follow natural language instructions to complete navigation tasks. 2.3.4. Expedition A1 Expedition A12 , developed by AGIBot, embodies the company’s commitment to seamlessly integrating advanced AI into robotics and fostering harmonious collaboration between humans and machines. Envisioning a future where robots serve as indispensable assistants to humans, AGIBot’s mission is to create intelligent and versatile robots

3.3. Control Here, we argue that the control module is the key component responsible for regulating robotic actions. This module plays a crucial role in ensuring that the robot’s actions are executed accurately and successfully, with a focus on the hardware aspects of action execution. 3.3.1. How to learn language-conditioned behavior Much of the previous work has focused on enabling robots and other agents to comprehend and execute natural language instructions [19, 35, 81]. There are various approaches to learning linguistically conditioned behaviors, such as image-based behavioral cloning that follows the BCZ [58] method or the MT-Opt [64] reinforcement learning method. Imitation learning

techniques train protocols on demonstration datasets [58, 153], while offline reinforcement learning has also been studied extensively [59, 71, 88]. However, some works suggest that imitation learning on demonstration data performs better than offline reinforcement learning [83], and other studies demonstrate the feasibility of offline reinforcement learning in theory and practice [72, 73]. Many works combine RL and Transformer structures [20, 60], and some works integrate imitation learning with reward conditions, such as Decision Transformer (DT) [20], namely combines imitation learning with reinforcement learning elements. However, DT does not enable the model to learn from the demonstration dataset to have better performance. Deep Skill Graphs (DSG) [5] present a novel approach to skill learning utilizing the option framework. This method leverages graphs to represent discrete aspects of the environment, enabling the model to acquire structured knowledge and learn complex skills within the given domain. CT employs goal-conditioned RL to transform the local skill-learning problem into a goal-conditioned Markov decision process (MDP) [61]. In the context of navigation robots, early approaches to enhancing navigation strategies with the natural language employed static machine translation [80] to discover patterns. The process involves utilizing discovery patterns to translate free-form instructions into formal languages that adhere to specific grammatical rules [19, 85, 127]. However, these methods were limited to structured state spaces. Recent works have also developed the VLN task as a sequence prediction problem [3, 87, 119]. Additionally, there are methods that leverage nearly 1M labeled simulation trajectory demonstration data for training [47], but applying these models in unstructured environments remains a significant challenge. Data-driven approaches for vision-based mobile robot navigation often depend on the utilization of realistic simulation techniques [70, 111, 144] or gathering supervised data to directly learn policies for achieving goals based on observations [38]. Alternatively, self-supervised learning methods can utilize unlabeled datasets or trajectories generated automatically by onboard sensors and hindsight relabeling learning [51, 63, 114]. 3.3.2. How to execute action after parsing nature language To determine whether a skill can be executed in the current state after parsing a natural language command, a temporal-difference-based (TD) reinforcement learning approach can be employed. This method learns a value function to evaluate whether the skill is executable or not [1]. The value function is derived from the corresponding affordance function of reinforcement learning [42]. Additionally, LM-Nav [117] utilizes a self-supervised learning method to enhance the parsing of free-form language instructions leveraging pre-trained VLM in a large number of previous environments. To address the challenges of long-term tasks, hierarchical reinforcement learning (HRL) [55] can be employed, where higher-level policies play a role in setting objectives for lower-level protocols to execute [90, 132]. The process of mapping natural language and observations into robot actions can also be viewed as a sequence modeling problem [9, 10, 29]. Transformer-based robot control, such as the Behavior Transformer [113], focuses on learning demonstrations that correspond to each task. Gato [104] suggests training a model on large datasets including robotic and non-robotic. 3.4. Interaction Interaction serves as a fundamental module that enables robots to engage and interact with both the environment and humans. To enhance robots' ability to interact in the physical world, they are often trained extensively. While some researchers utilize artificial intelligence to interact in virtual environments, such as games or simulations, ultimately, these models must be transferred to the real world.

Large Language Models for Robotics: A Survey However, the accuracy of these models tends to be lower in real-world settings compared to simulated environments.

In this survey, we summarized the methods and technologies currently used for large models in robots. First, we reviewed some basic concepts of large language models and common large models. We explain what improvements will be brought to robots by using large models as brains. We also introduce the representative LLM-based robot models proposed in recent years, such as LM-Nav [117], PaLMSayCan [1], PaLM-E [34], etc. Next, we divide the robot into four modules: perception, decision-making, control, and interaction. For each module, we discuss the relevant technologies and their functions, including the perception module's ability to process the robot's input from the surroundings; the decision-making module's capacity to understand human instructions and plan; the control module's role in processing output actions; the interaction module's ability to interact with the environment. We also explore the potential application scenarios of current robots based on LLMs and discuss the challenges, such as training, safety, shape, deployment, and long-term task performance. Finally, we consider the social and ethical implications of post-intelligent robots and their potential impact on human society. As LLMs continue to evolve, robots may become increasingly intelligent and capable of processing instructions and tasks more efficiently. With advancements in hardware, robots may eventually become reliable assistants for humans, as depicted in science fiction movies. However, we must also be mindful of their potential impact on society and address any concerns proactively. Embodied intelligence is a new paradigm for the development of intelligent science and is of great significance in leading the development of the future. LLM-based robotics represent a potential path to embodied intelligence. We hope this survey can provide some inspiration to the community and facilitate research in related fields.