applied sciences

MDPI

Review

A Survey of Robot Intelligence with Large Language Models

Hyeongyo Jeong [1,†], Haechan Lee [1,†], Changwon Kim [2,*] and Sungtae Shin [1,*]

[1] Department of Mechanical Engineering, Dong-A University, Busan 49315, Republic of Korea
[2] School of Mechanical Engineering, Pukyong National University, Busan 48513, Republic of Korea
* Correspondence: ckim@pknu.ac.kr (C.K.); stshin@dau.ac.kr (S.S.)
† These authors contributed equally to this work.

Abstract: Since the emergence of ChatGPT, research on large language models (LLMs) has actively progressed across various fields. LLMs, pre-trained on vast text datasets, have exhibited exceptional abilities in understanding natural language and planning tasks. These abilities of LLMs are promising in robotics. In general, traditional supervised learning-based robot intelligence systems have a significant lack of adaptability to dynamically changing environments. However, LLMs help a robot intelligence system to improve its generalization ability in dynamic and complex real-world environments. Indeed, findings from ongoing robotics studies indicate that LLMs can significantly improve robots' behavior planning and execution capabilities. Additionally, vision-language models (VLMs), trained on extensive visual and linguistic data for the vision question answering (VQA) problem, excel at integrating computer vision with natural language processing. VLMs can comprehend visual contexts and execute actions through natural language. They also provide descriptions of scenes in natural language. Several studies have explored the enhancement of robot intelligence using multimodal data, including object recognition and description by VLMs, along with the execution of language-driven commands integrated with visual information. This review paper thoroughly investigates how foundation models such as LLMs and VLMs have been employed to boost robot intelligence. For clarity, the research areas are categorized into five topics: reward design in reinforcement learning, low-level control, high-level planning, manipulation, and scene understanding. This review also summarizes studies that show how foundation models, such as the Eureka model for automating reward function design in reinforcement learning, RT-2 for integrating visual data, language, and robot actions in vision-language-action models, and AutoRT for generating feasible tasks and executing robot behavior policies via LLMs, have improved robot intelligence.

Keywords: embodied intelligence; foundation model; large language model (LLM); vision-language model (VLM); vision-language-action (VLA) model; robotics

Citation: Jeong, H.; Lee, H.; Kim, C.; Shin, S. A Survey of Robot Intelligence with Large Language Models. *Appl. Sci.* **2024**, *14*, 8868. https://doi.org/10.3390/app14198868

Academic Editors: Luis Gracia and J. Ernesto Solanes

Received: 6 September 2024
Revised: 24 September 2024
Accepted: 25 September 2024
Published: 2 October 2024

1. Introduction

To enhance the intelligence of robots in real-world environments that interact with humans, developing robots capable of perceiving, acting, and interacting like humans is a crucial goal. The recent advancements in large language models (LLMs) such as GPT-4o [1] have significantly altered the field of robotic AI research. These LLMs, trained on vast amounts of textual data, have shown excellent performance in enabling robots to communicate with humans more naturally and efficiently. Moreover, beyond the impacts on human–robot interaction (HRI), there is ongoing research aimed at surpassing the limitations of traditional low-level robot control techniques and planning algorithms by utilizing the high-level situational awareness and knowledge-based planning capabilities of LLMs. Notably, the programming capabilities of ChatGPT in the research presented by Microsoft's ChatGPT for Robotics [2] have introduced a new paradigm for applying LLMs in the robotics field.

The goal of robot intelligence is to enable robots to operate autonomously in complex environments, interact naturally with humans, and make high-level decisions. To promote

advancements in robot intelligence, the adoption of foundation models, such as LLMs and vision-language models (VLMs), which boast large parameter scales and pre-training on massive datasets, is accelerating. These foundation models can perform various tasks, such as complex language understanding and generation and visual perception, enabling robots to engage with their environment in a more human-like manner.

While traditional robot intelligence systems are highly effective in structured and predictable environments, they are significantly limited in their ability to adapt to dynamically changing and complex real-world scenarios. In general, the intelligence models used in these robotic systems are based on supervised learning, which requires large amounts of labeled data. This process is inherently resource-intensive. Moreover, these models are designed for a specific environment and require reconfiguration whenever the task or environment changes. This renders the robots challenging to adapt and scale to disparate environments [3]. For practical robot systems, it is essential that they are able to flexibly respond to the ever-changing physical environment. From this perspective, the generalization of affordable tasks, environmental adaptability, and the accuracy of execution, planning, and reasoning capabilities remain significant challenges for traditional robotic intelligence systems [4].

However, LLMs and VLMs help a robot intelligence system to enhance its generalization capability in dynamic and complex real-world environments. LLMs can leverage pre-trained knowledge from extensive datasets to augment their ability to generalize to everyday tasks that are typically expected of robots. Unlike the conventional supervised models, LLMs can utilize zero-shot and few-shot learning to help robots quickly adapt to new environments without additional training [5]. This has the advantage of significantly reducing the need for costly data collection and labeling. In addition, robot systems equipped with LLMs can process complex instructions based on their ability to understand and generate natural language, which can improve human–robot interactions. Furthermore, LLMs can be integrated with multimodal sensors such as LiDAR, depth, voice, tactile, proprioception, and visual information, which enables robots to comprehensively understand and adapt to their environment [6].

LLMs have demonstrated exceptional capabilities in processing and understanding text-based information, significantly enhancing robotic communication abilities. For instance, robots can accurately comprehend and execute natural language commands via LLMs, providing scalability and flexibility beyond traditional word-based robotic command systems. Consequently, robots can respond more adaptably and intelligently in interactions with human users, allowing them to engage in complex problem-solving and decision-making processes beyond simple mechanical tasks.

Additionally, LLMs not only enhance a robot's communication skills to improve HRI usability but also boost the robot's planning abilities. Planning involves setting goals and devising a sequence of actions to achieve them, which are essential in determining a robot's autonomy and efficiency. LLMs interpret natural language from users and complex commands, enabling robots to establish and execute suitable plans in various situations. Moreover, LLMs adapt flexibly to new situations through a zero-shot approach and utilize past data for learning. These capabilities indicate that robots can play a vital role in autonomously navigating changing environments and resolving unexpected issues.

Moreover, VLMs such as CLIP [7], which are trained to solve vision question answering (VQA) tasks, have the ability to process visual and linguistic information simultaneously. This ability allows robots to visually perceive their surroundings and integrate this information into linguistic descriptions, enabling more sophisticated situational awareness. For instance, using VLMs, a robot can recognize objects and provide descriptions, as well as understand and execute user commands based on visual cues. This integrated approach significantly enhances a robot's autonomy and interaction capabilities.

In practice, building on the capabilities of predecessors RT-1 [8] and RT-2 [9], which enable low-level actuator control using LLMs and VLMs, Google has introduced AutoRT [10]. AutoRT is a system where robots interact with real-world objects to collect motion data. It

begins by exploring the surrounding space to identify feasible tasks, then uses a VLM to understand the situation and an LLM to propose possible tasks. By inputting the robot's operational guidelines and safety constraints into the LLM as prompts, AutoRT assesses the validity of the proposed tasks and the necessity for human intervention. Throughout this process, AutoRT safely selects and executes feasible tasks while collecting relevant data.

Nvidia has also introduced Eureka (Evolution-driven Universal REward Kit for Agent) [11], a system that automatically designs reward functions for reinforcement learning problems using the capabilities of LLMs, which include understanding physical causality in the real world, problem-solving through trial-and-error feedback, and code generation abilities. Eureka can autonomously generate reward functions for a variety of tasks and robots without needing specific templates for each. This allows for the generation of human-level reward functions for diverse robots and tasks without human input. Furthermore, Eureka has demonstrated the ability to solve complex problems that were previously unsolved by expert-designed reward functions.

Given these research outcomes, integrating language models into robotic intelligence presents significant potential to enhance robot capabilities and applications dramatically, thereby redefining their roles in diverse industries and everyday life. Therefore, this survey paper explores recent research trends in LLM- and VLM-based robot intelligence, aiming to provide a comprehensive understanding of future development possibilities by examining the application of language models in various robotic research fields. It also seeks to highlight research cases, identify current limitations, and suggest future research directions.

To chronicle this advancement in robotics research fields, this review paper presents the following contributions:

- This paper summarizes and introduces the foundational elements and tuning methods of LLM architecture.
- It explores and arranges prompt techniques to enhance the problem-solving abilities of LLMs.
- It reviews and encapsulates how LLMs and VLMs have been employed to augment robot intelligence across five topics as shown in Figure 1: (1) reward design for reinforcement learning, (2) low-level control, (3) high-level planning, (4) manipulation, and (5) scene understanding.

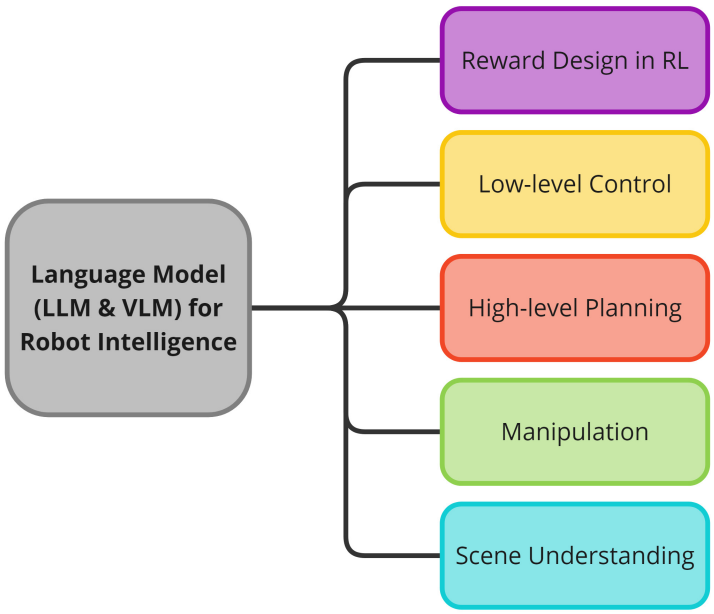

Figure 1. Five categories for robot intelligence with large language models in this study.

The **reward design in RL** category represents a research field in which an LLM develops and enhances reward functions employed in reinforcement learning via code-based descriptions and natural language input. This enables robots to learn optimal policies for specific tasks through reinforcement learning, even in complex environments. The **low-level control** category includes a research area in which LLMs and VLMs generate command sequences that directly control the robot's actuators through natural language and visual input. The **high-level planning** category is a research area where the LLM identifies the present circumstances and objective of the tasks, subsequently developing an explainable plan based on the reasoning required for problem-solving. In this research area, the LLM is also tasked with developing the optimal robot behavior plan, which entails evaluating the feasibility of the established plan. In the **manipulation** category, the LLM interprets high-level instructions and the VLM (and LLM) analyzes various conditions based on their understanding of the surroundings to assist robot arms in performing the specific tasks. While this category can be broadly included in the high-level planning category, there are numerous studies that are specifically related to manipulation with a robot arm, which is why the manipulation category was separated. The **scene understanding** category represents a research area that seeks to combine LLMs and VLMs with the objective of assisting robots in comprehending their surrounding environment. This is accomplished by identifying objects based on natural language instructions and visual information, as well as by evaluating the relationships between them. This research area is also closely related to the field of autonomous visual navigation. From a boarder perspective, there is an overlap between the scene understanding category and the perception-related components of the high-level planning category. However, in this review, the scene understanding category was considered a distinct category due to its prevalence as an application of VLM models.

Table 1 lists resources that aid in understanding robot intelligence based on language models. The review [5] examined recent advancements in LLMs with a particular emphasis on four key areas: pre-training, adaptation tuning, utilization, and capacity evaluation. Furthermore, it provided a summary of the resources currently available for the development of LLMs and discussed potential future directions for research in this field. The survey [12] conducted a comprehensive and systematic review of VLMs for visual recognition tasks. It addressed the evolution of the visual recognition paradigm, the principal architectures and datasets, and the fundamental principles of VLMs. Moreover, the paper provided an overview of the pre-training, transfer learning, and knowledge distillation methods employed in the context of VLMs. The review [3] examined the potential for leveraging existing natural language processing and computer vision foundation models in robotics. In addition, it explored the possibility of developing a robot-specific foundation model. The review [13] presented an analysis of recent studies on language-based approaches to robotic manipulation. It comprised an analysis of learning paradigms integrated with foundation models related to manipulation tasks, including semantic information extraction, environment and evaluation, auxiliary tasks, task representation, safety issues, and other pertinent considerations. The survey paper [14] presented an analysis of recent research articles that employed foundation models to address robotics challenges. It investigated the extent to which foundation models enhanced robot performance in perception, decision-making, and control. In addition, it examined the obstacles impeding the implementation of foundation models in robot autonomy and proposed avenues for future advancements. The review paper [15] presented a comprehensive review of the research in the field of vision and language navigation (VLN), encompassing tasks, evaluation metrics, and methodologies related to autonomous navigation.

Table 1. Useful Review Papers.

Title	Keywords	Ref.
Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis	Foundation Models	[3]
A Survey of Large Language Models	LLM	[5]
Vision-Language Models for Vision Tasks: A Survey	VLM	[12]
Language-conditioned Learning for Robotic Manipulation: A Survey	LLM, VLM, Manipulation	[13]
Foundation Models in Robotics: Applications, Challenges, and the Future	Foundation Models	[14]
Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions	VLN	[15]

2. Review Protocol

This survey covered four databases: Web of Science, ScienceDirect, IEEE Xplore, and arXiv. In fact, many of the articles surveyed had not been peer-reviewed and published at the time of our search because the subject matter was relatively recent. Therefore, a considerable number of articles reviewed in this survey were sourced from the arXiv database.

The selection process of this study primarily relied on two iterations:

- The titles and abstracts of the articles were reviewed to eliminate duplicates and irrelevant articles.
- The full texts of the selected articles from the first iteration were thoroughly examined and categorized.
- Article searching began on 18 September 2023.

Regarding the search queries,

- the publication years were those after 2020,
- the keywords of Robotics and LLM, which were ((“Robotic” OR “Robotics”) AND (“LLM” OR “LM” OR “Large Language Model” OR “Language Model”)), and relevant journal and conference articles written in English were considered.

From these search criteria, recent studies utilizing language models in robotics research were expected to be collected. Our aim is to provide a robust understanding of how language models and their variants have been utilized to enhance robot intelligence in the literature.

All articles that met the above criteria were included in this review. Following an intensive survey of the abstracts of the selected articles, we categorized the research topics into five groups: reward design for reinforcement learning, low-level control, high-level planning, manipulation, and scene understanding. Figure 1 illustrates these five categories. Our categorization was based on a thorough review of the sources in the literature. Subsequently, duplicate articles were removed, and those not meeting the specified eligibility criteria were excluded. The exclusion criteria included: (1) articles in languages other than English and (2) articles discussing general concepts that do not focus on deep reinforcement learning-based manipulation.

3. Related Works

3.1. Language Model

Zhao’s LLM review paper [5] categorizes the evolution of Language Models (LMs) into four phases. The initial stage, the statistical language model (SLM) [16–19], utilizes methods based on statistical learning techniques and the Markov assumption to construct word prediction models. A notable method from this phase is the n-gram language model, which predicts words based on a fixed context length of n. Although SLMs have enhanced performance in various domains such as information retrieval (IR) [16,20] and natural

language processing (NLP) [21–23], higher-order language models have encountered limitations due to the curse of dimensionality, which necessitates estimating exponentially increasing transition probabilities.

The subsequent phase of LMs, termed neural language models (NLMs), leveraged neural networks such as multi-layer perceptron (MLP) and recurrent neural networks (RNNs) to model the probability of word sequences [24–26]. A key element of this stage is the development of word vectors, also known as word embeddings, which form word prediction models based on vectors that use a distributed representation of words [24,27]. Word2vec, a simplified shallow neural network approach, was introduced to learn these distributed word representations [28,29]. It proved highly effective across various NLP tasks by calculating meaningful similarities between word vectors. NLMs progressed from basic word sequence modeling to sophisticated techniques for representing language through word2vec.

Following the NLM phase, the field advanced to pre-trained language models (PLMs), which encompass models such as ELMO [30] and BERT [31]. PLMs, utilizing large-scale text data, learn text patterns, structures, and meanings to develop pre-trained context-sensitive word representations. They have successfully executed a variety of language understanding and generation tasks using this acquired knowledge. ELMo [30] introduced a pre-training method employing bidirectional LSTM (biLSTM) networks for modeling deep contextualized word representations, optimizing performance through specific fine-tuning of the trained biLSTM network for downstream tasks. ELMo is also characterized as a bidirectional language model for its dual-directional use of language models.

Another PLM model, BERT [31], leverages the transformer architecture [32], exhibiting remarkable effectiveness with self-attention mechanisms and parallel processing. BERT, a pre-trained bidirectional language model, utilizes extensive unlabeled text data. The method of unsupervised learning-based pre-training in BERT comprises two primary tasks: masked language models and next sentence prediction. PLMs that provide pre-trained context-aware word representations are profoundly effective in general-purpose semantic feature extraction, facilitating enhancements in NLP task performance. Owing to these characteristics, numerous subsequent studies employing pre-training and fine-tuning have been introduced, featuring varied structures [33,34] (e.g., BART [33] and GPT-2 [35]) and enhancing pre-training strategies [36–38].

Based on subsequent studies, it has been found that increasing the model size or data size of PLMs typically enhances the performance of LM models [39]. This has prompted research into training large-scale PLMs, such as GPT-3 with 175B parameters and PaLM with 540B parameters. The focus of this research, grounded in scaling laws, primarily centers on augmenting model sizes and exploring the capabilities of larger models. These capabilities, known as the emergent abilities of LLMs, have sparked significant interest. For example, GPT-3 can address problems it has not been trained on with minimal examples through in-context learning, a feat GPT-2 finds challenging. Due to these characteristics, the academic community commonly designates these large PLMs as LLMs [40–43]. Consequently, research in this area is highly active. Notably, since the introduction of OpenAI's ChatGPT, there has been a surge in the number of arXiv papers on LLMs. Following Microsoft's announcement [2] about integrating ChatGPT into robotics, a variety of studies have explored the application of LLMs across different areas of robotics research. The available LLM models are presented in chronological order in Table 2. Additionally, Table 3 includes the VLM models.

Table 2. Chronicle of LLM models.

Release Date	Model Name	Developer	Ref.	Release Date	Model Name	Developer	Ref.
2018-06	GPT-1	OpenAI	[44]	2024-02	OLMo	Allen Institute for AI	[45]
2019-02	GPT-2	OpenAI	[35]	2024-02	StarCoder2	Hugging Face	[46]
2019-10	T5	Google	[47]	2024-03	Claude 3	Anthropic	[48]
2020-05	GPT-3	OpenAI	[49]	2024-03	InternLM2	Shanghai AI Lab	[50]
2021-07	Codex	OpenAI	[51]	2024-03	Jamba	AI21Labs	[52]
2021-09	FLAN	Google	[53]	2024-04	Stabe Code	Stability AI	[54]
2021-10	T0	Hugging Face	[37]	2024-04	HyperCLOVA	Naver	[55]
2021-12	Gopher	DeepMind	[56]	2024-04	Grok-1.5	xAI	[57]
2022-03	InstructGPT	OpenAI	[58]	2024-04	Llama3	Meta AI Research	[59]
2022-04	PaLM	Google	[60]	2024-04	Phi-3	Microsoft	[61]
2022-05	OPT	Meta AI Research	[62]	2024-05	GPT-4o	OpenAI	[1]
2023-02	LLaMA	Meta AI Research	[63]	2024-06	Claude 3.5	Anthropic	[64]
2023-03	Alpaca	Stanford Univ.	[65]	2024-07	GPT-4o mini	OpenAI	[66]
2023-03	GPT-4	OpenAI	[50]	2024-07	Falcon2-11B	TII	[67]
2023-05	StarCoder	Hugging Face	[68]	2024-07	Llama 3.1 405B	Meta AI Research	[69]
2023-07	LLaMA2	Meta AI Research	[70]	2024-07	Large2	Mistral AI	[71]
2023-09	Baichuan2	Baidu	[72]	2024-07	Gemma2	Gemma Team, Google DeepMind	[73]
2023-10	Mistrial	Mistral AI	[74]	2024-08	EXAONE 3	LG AI Research	[75]
2024-01	DeepSeek-Coder	DeepSeek-AI	[76]	2024-08	Grok-2 and Grok-2 mini	xAI	[77]

Table 3. Chronicle of VLM models.

Release Date	Model Name	Developer	Ref.	Release Date	Model Name	Developer	Ref.
2020-05	DETR	Facebook AI	[78]	2023-04	LLaVA	UW–Madison	[79]
2020-12	DeiT	Facebook AI	[80]	2023-04	MiniGPT-4	KAUST	[81]
2021-02	DALL-E	OpenAI	[82]	2023-09	GPT-4V	OpenAI	[83]
2021-02	CLIP	OpenAI	[7]	2023-11	Florence-2	Microsoft	[84]
2021-03	Swin Transformer	Microsoft	[85]	2024-01	Lumiere	Google Research	[86]
2021-05	SegFormer	Univ. of Hong Kong	[87]	2024-01	Fuyu	Adept	[88]
2021-06	Vision Transformer	Google Research, Brain	[89]	2024-03	Gemini 1.5	Gemini Team, Google	[90]
2021-06	BEiT	HIT, Microsoft Research	[91]	2024-04	InternLMXComposer2	Shanghai AI Lab.	[92]
2021-11	ViTMAE	Facebook AI	[93]	2024-04	IDEFICS 2	Hugging Face	[94]
2021-12	Stable Diffusion	LMU Munich, IWR	[95]	2024-05	Idefics2	Hugging Face	[96]
2022-03	R3M	Meta AI, Stanford Univ.	[97]	2024-05	Chameleon	Meta AI Research	[98]
2022-04	Flamingo	DeepMind	[99]	2024-07	InternLM-XComposer-2.5	Shanghai AI Lab.	[98]
2023-01	BLIP-2	Salesforce Research	[100]	2024-07	PaliGemma	Univ. of Hong Kong	[101]
2023-04	SAM	Meta AI Research	[102]	2024-08	SAM 2	Meta AI	[103]
2023-04	DINOv2	Meta AI Research	[104]	2024-08	Qwen-VL2	Alibaba Group	[105]

3.2. LLM Architectures and Tunings

The architecture of LLMs fundamentally utilizes the transformer architecture, with three representative types based on different transformer configurations as shown in Figure 2. Firstly, the prevalent encoder–decoder structure of transformers employs the encoder to process the input sequence and generate a latent representation through multi-head self-attention layers; the decoder then uses cross-attention on this representation to autoregressively produce the target sequence. Notable encoder–decoder PLMs include T5 [47] and BART [33], with Flan-T5 [106] being an encoder–decoder-based LLM. Secondly, the causal decoder employs a unidirectional attention mask to restrict each input token to attend only to past and present tokens, processing input and output tokens similarly through the decoder. This method underpins the development of the GPT series. Lastly, the prefix decoder, resembling the causal decoder's masking mechanism, allows bidirectional attention on prefix tokens [107] and unidirectional attention on generated tokens. Similar to the encoder–decoder, the prefix decoder bidirectionally encodes the prefix sequence and sequentially predicts output tokens individually. Examples of prefix decoder-based LLMs include GLM-130B [108] and U-PaLM [109]. Additionally, various architectures have been proposed to address efficiency challenges during training or inference with long inputs, due to the quadratic computational complexity of the traditional transformer architecture. For instance, the Mixture-of-Experts (MoE) scaling method [34] sparsely activates a subset of the neural network for each input.

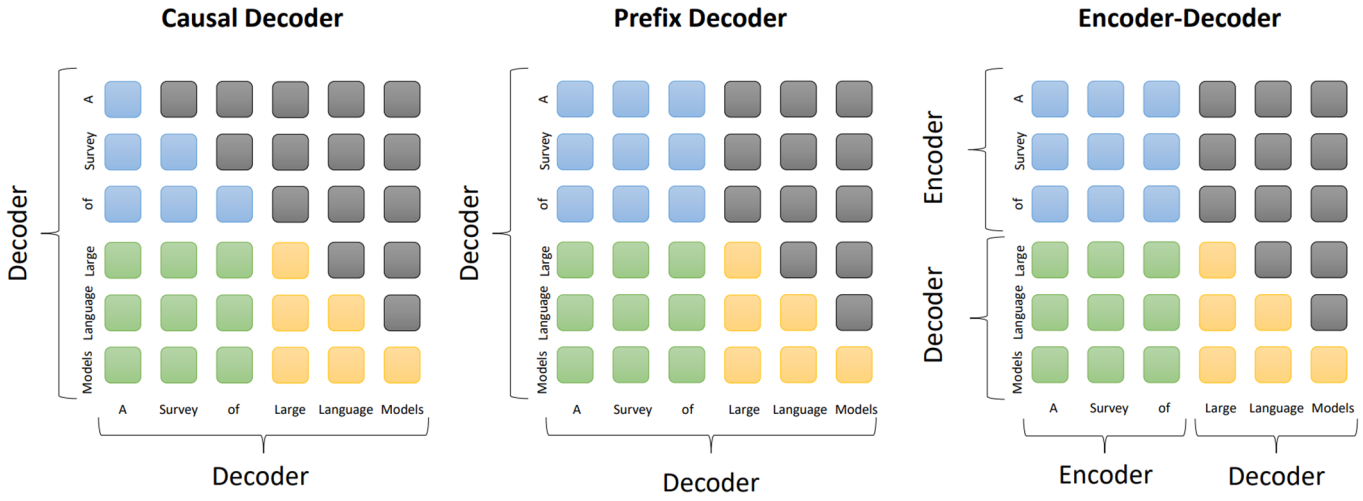

Figure 2. Attention patterns in three mainstream architectures: Causal Decoder (**left**), Prefix Decoder (**middle**), and Encoder–Decoder (**right**). The blue, green, yellow, and grey rounded rectangles represent attention between prefix tokens, attention between prefix and target tokens, attention between target tokens, and masked attention [5].

In terms of the tuning of LLMs, these models are essentially pre-trained on massive datasets and require fine-tuning for different application domains. However, the considerable model size and number of parameters pose challenges for fine-tuning on standard computers and GPUs. The subsequent sections will discuss methods to address these challenges.

LLM tuning is broadly divided into two categories based on the training objective. Instruction tuning is a form of supervised learning where the training data typically include descriptions of tasks, inputs, and corresponding outputs. This type of tuning is designed (1) to enhance the functional capabilities of LLMs, (2) to specialize them by training with discipline-specific information, and (3) to improve task generalization and consistency through a better understanding of natural language commands. Conversely, alignment tuning (or preference alignment) seeks to align the behavior of LLMs with human values and preferences. Prominent methods include reinforcement learning from human feedback

(RLHF) [110], which involves fine-tuning LLMs using human feedback to better reflect human values, and direct preference optimization (DPO) [111], focusing on training with pairs of human preferences that usually include an input prompt and the preferred and non-preferred responses.

For both instruction tuning and alignment tuning, which involve training LLMs with extensively large model parameters, substantial GPU memory and computational resources are required, with high costs typically incurred when utilizing cloud-based resources. Under these conditions, parameter-efficient fine-tuning (PEFT) offers a method designed to efficiently conduct fine-tuning of such LLMs [112].

Among the methods of PEFT, there are four major approaches as shown in Figure 3: adapter tuning, prompt tuning, prefix tuning, and low-rank adaptation (LoRA). Adapter tuning [113,114] involves integrating small neural network modules, known as adapters, into the core components of a transformer model, specifically into the attention and feed-forward layers. These adapters are inserted serially following these layers, allowing fine-tuning of only the adapter modules according to specific task goals, while the parameters of the original language model remain unchanged. Consequently, adapter tuning effectively reduces the number of trainable parameters. Additionally, prompt tuning [115,116] diverges from adapter tuning by adding trainable prompt vectors to the input layer. Prefix tuning [117] entails appending a sequence of prefixes to each transformer layer of the language model, which consists of trainable continuous vectors. During fine-tuning, the model focuses on identifying the optimal prefix vectors, which are retained for use in LLM model inference.

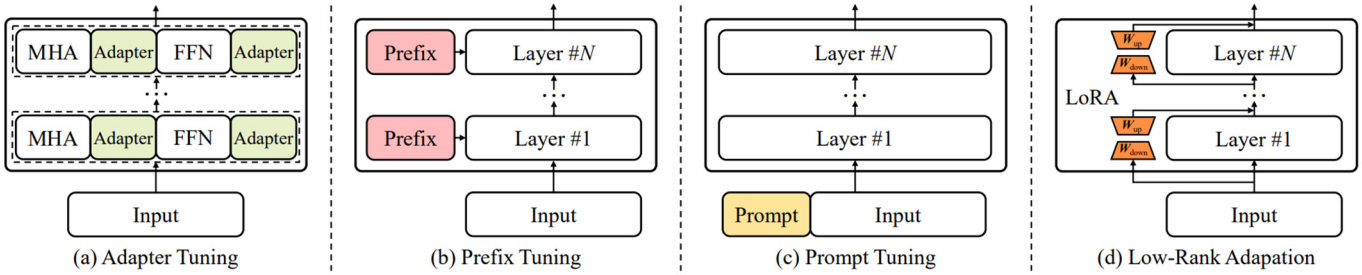

Figure 3. An overview of four strategies for parameter-efficient fine-tuning: (**a**) Adapter Tuning, (**b**) Prefix Tuning, (**c**) Prompt Tuning, and (**d**) Low-Rank Adaptation [5].

In practice, a commonly employed method for LLM fine-tuning, LoRA [118], uses a low-rank constraint on transformer layers to approximate the update matrices through training. This method keeps the original LLM parameter matrices fixed and approximates the parameter updates using low-rank decomposition matrices. The primary benefit of LoRA is a substantial reduction in the memory and storage requirements for fine-tuning, such as VRAM. Additionally, quantization methods, which directly minimize the memory size required for parameter representation, are frequently utilized in LLM fine-tuning. Specifically, the practice of merging LoRA with quantization is known as QLoRA [119].

3.3. Prompt Techniques for Increasing LLM Performance

To enhance the performance of LLMs, the most straightforward approach involves training with additional data via fine-tuning techniques, which mirrors supervised learning in conventional machine learning. Another method for improving performance involves the use of in-context learning, which capitalizes on prompts for zero-shot learning, a capability first observed in LLMs with the advent of GPT-3. The adaptation of these prompts for specific tasks is known as prompt engineering. Fundamentally, prompt engineering (or prompting) entails supplying inputs to the model to perform a distinct task, designing the input format to encapsulate the task's purpose and context, and generating the desired output. The four components of prompt engineering can be analyzed as follows: within the prompt, "*Instructions*" delineate the specific tasks or directives for the model and

"*Context*" provides external or additional contextual information that can tune the model. Furthermore, "*Input data*" refers to the type of input or questions seeking answers, and "*Output data*" defines the output type or format within the prompt, thereby optimizing the LLM's performance for particular tasks. Various methodologies for creating prompts have been introduced, as described below.

Zero-shot prompting [53] is a technique that allows the model to take on new tasks with no prior examples. The model relies solely on the task description or instructions without additional training. Likewise, **few-shot prompting** [49] introduces a small number of examples to aid the model in learning new tasks. This approach does not require extensive datasets and can improve the model's performance through a limited set of examples.

Chain-of-thought (CoT) [41] is a technique that explicitly describes intermediate reasoning steps, enabling the model to perform step-by-step reasoning. This approach allows the model to incrementally solve complex tasks. For instance, when asked, "If someone's age will be 30 in 5 years, how old are they now?", the model uses the information "age in 5 years is 30" to perform the intermediate reasoning step of "$30 - 5 = 25$" to derive the final answer. **Self-consistency** [120] involves the model generating various independent reasoning paths through few-Shot CoT, ultimately selecting the most consistent answer among the outputs. This method enhances the performance of CoT prompts in both arithmetic and commonsense reasoning tasks. **Multimodal CoT** [121] is a two-stage framework that integrates text and visual modalities. Initially, intermediate reasoning steps are generated through rationale generation based on multimodal data. Subsequently, the answer inferences are intertwined, and the informative rationales are utilized to derive the final answer.

Generally, CoT relies on human-generated annotations, which may not always provide the optimal solution for problem-solving. To overcome this limitation, **active prompt** [122] has been proposed. Active prompt enhances model performance by intensively training the model on questions with higher uncertainty levels. It evaluates the uncertainty of answers by posing questions to the model, with or without CoT examples. Questions with high uncertainty are selected for human annotation, and newly annotated examples are used to reason through each question. **Program-aided language models** (PAL) [123] is a technique that employs the model to understand natural language problems and generate programs as intermediate reasoning steps. Unlike CoT, PAL solves problems stepwise using a program runtime such as Python rather than free-form text.

Tree of thoughts (ToT) [124] is a method whereby the model breaks down a problem into smaller units called thoughts, which it then assesses through a reasoning process to gauge its progress toward a solution. The ability of the model to generate and evaluate these thoughts is integrated with search algorithms such as breadth-first and depth-first search, facilitating systematic thought exploration with lookahead and backtracking capabilities. In contrast to the CoT method, which addresses problems sequentially, ToT concurrently examines multiple pathways to find a solution. **Prompt chaining** [125] is a strategy where the model divides a task into sub-tasks, uses the outputs of each sub-task as subsequent inputs, and links prompts in input–output pairs. This approach improves the precision and consistency of the outputs at each stage and simplifies the handling of complex tasks by subdividing them into manageable sub-tasks.

Generated knowledge prompting [126] is a technique in which the model incorporates knowledge and information pertinent to the question and provides it alongside the question to generate more accurate answers. This method not only enhances the commonsense reasoning capabilities but also retains the flexibility of existing models. **Retrieval augmented generation** (RAG) [127] merges external information retrieval with natural language generation. RAG can be fine-tuned for knowledge-intensive downstream tasks and enables straightforward modifications or additions of knowledge within the framework. This facilitates an increase in the model's factual consistency, enhances the reliability of generated responses, and helps alleviate issues with hallucination. **Automatic reasoning and tool-use** (ART) [128] is a framework that utilizes external tools to autonomously gen-

erate intermediate reasoning steps. It chooses relevant tasks from a library that includes demonstrations and calls on external tools as necessary to integrate their outputs into the reasoning process. The model generalizes from demonstrations using tools to decompose new tasks and learns to use tools effectively. Enhancing ART's performance is possible by modifying the task library or incorporating new tools. **Automatic prompt engineer** (APE) [129] is a framework designed for the automatic generation and selection of commands. The model generates command candidates for a problem and selects the most suitable one based on a scoring function, such as execution accuracy or log probability.

Directional stimulus prompting [130] is a technique that directs the model to consider and generate responses in a particular direction. By deploying a tunable policy LM (e.g., T5 [47]), it creates directional stimulus prompts for each input and uses these as cues to steer the model toward producing the desired outcomes [131]. **ReAct** combines reasoning with action within the model. It enables the model to perform reasoning in generating answers, take actions based on external sources (e.g., documents, articles, and news), and refine reasoning based on observations of these actions. This process facilitates the creation, maintenance, and modification of action plans while incorporating additional information from interactions with external sources. **Reflexion** [132] augments language-based agents with language feedback. Reflexion involves three models: the actor, the evaluator, and self-reflection. The actor initiates actions within a specific environment to generate task steps, the evaluator assesses these steps, and self-reflection provides linguistic feedback, which the actor uses to formulate new steps and achieve the task's objective. The introduced prompt techniques are summarized in Table 4.

Table 4. Prompt Techniques.

Name	Explanation	Ref.
Zero-Shot Prompting	Enabling the model to perform new tasks without any examples	[53]
Few-Shot Prompting	Providing a few examples to enable performing new tasks	[49]
Chain-of-Thought	Explicitly generating intermediate reasoning steps to perform step-by-step inference	[41]
Self-Consistency	Generating various reasoning paths independently through Few-Shot CoT, with each path going through a prompt generation process to select the most consistent answer	[120]
Generated Knowledge Prompting	Integrating knowledge and information relevant to a question, and then providing it along with the question to generate accurate answers	[126]
Prompt Chaining	Dividing a task into sub-tasks and connecting prompts for each sub-task as input–output pairs	[125]
Tree of Thoughts	Dividing a problem into subproblems with intermediate steps that serve as "thoughts" towards solving the problem, where each thought undergoes an inference process and self-evaluates its progress towards solving the problem	[124]
Retrieval Augmented Generation	Combining external information retrieval with natural language generation	[127]
Automatic Reasoning and Tool-use	Using external tools to automatically generate intermediate reasoning steps	[128]
Automatic Prompt Engineer	Automatically generating and selecting commands	[129]
Active Prompt	Addressing the issue that the effectiveness may be limited by human annotations	[122]
Directional Stimulus Prompting	Guiding the model to think and generate responses in a specific direction	[130]
Program-Aided Language Models	Using models to understand natural language problems and generate programs as intermediate reasoning steps	[123]
ReAct	Combining reasoning and actions within a mode	[131]
Reflexion	Enhancing language-based agents through language feedback	[132]
Multimodal CoT	A two-stage framework that integrates text and vision modalities	[121]

4. Language Models for Robotic Intelligence

4.1. Reward Design in Reinforcement Learning

Research in reinforcement learning, closely associated with the field of robotics, has actively incorporated studies using LLM models. Specifically, Nvidia has developed a GPU-based multi-environment reinforcement learning platform. Utilizing its Omniverse 3D virtual environment platform, Nvidia created Isaac Sim, which is dedicated to robot simulation. Isaac Sim published research findings on Isaac Gym (Preview), which achieved significant reductions in reinforcement learning training times through GPU-based multi-environment approaches. Subsequently, Isaac Gym (Preview)'s features were integrated into Isaac Sim and released as Omni Isaac Gym. Later, Nvidia introduced Orbit [133], facilitating the simulation of PhysX 5.1-based cloth, soft-body, fluid, and rigid-body dynamics, along with RGBD, LiDAR, and contact sensor simulation. Orbit also incorporates various robot platforms into the simulation environment. Recently, Orbit was updated to Isaac Lab and integrated into Isaac Sim 4.0. Nvidia has continuously advanced dynamic simulation environment technologies for reinforcement learning using GPU parallel computation. Leveraging this GPU reinforcement learning, they launched Eureka [11], which automates the design of reward functions for reinforcement learning using LLMs. Following this, Nvidia introduced DrEureka [134], an automated platform addressing the Sim2Real problem [135] in reinforcement learning based on Eureka.

Eureka (Evolution-driven Universal REward Kit for Agent) [11], shown in Figure 4, automatically generates reward functions for various tasks using different robots, eliminating the need for specific templates or tailored reward functions for the robot's form or explanations for reinforcement learning tasks. Eureka consists of three main components: *environment-as-context*, *evolutionary search*, and *reward reflection*. Environment-as-context generates executable reward functions in a zero-shot manner by utilizing virtual environment source (Python) code as context. Evolutionary search iteratively generates reward function candidates and proposes enhanced functions based on previously generated and best-performing ones, while also creating new functions through mutation. Reward reflection offers a text summary of reward function quality based on training statistics recorded during reinforcement learning, which assists in generating subsequent reward functions as feedback for the performance of previous functions. The reward functions generated outperformed expert-generated functions in 83% of benchmark tests. Moreover, Eureka solved the pen spinning problem where a robot hand must spin a pen as much as possible according to predefined rotations, a task previously considered unsolvable through manual reward engineering. Eureka introduces a universal reward function design algorithm based on a code LLM and in-context evolutionary search, facilitating human-level reward generation for various robots and tasks without the need for prompt engineering or human intervention.

Following Eureka, DrEureka [134], shown in Figure 5, was developed to address the sim-to-real problem by automatically configuring appropriate reward functions and domain randomization for physical environments. DrEureka's reward-aware physics priors mechanism defines the lower and upper bounds of physical environment parameters based on policies trained through initial reinforcement learning, facilitating reinforcement learning across various physical environment domains. This randomization enables the trained model to excel in actual environments. Consequently, DrEureka achieved benchmark success in real-world quadruped locomotion with walking globe and cube-rotation manipulation using real robots, all without human supervision.

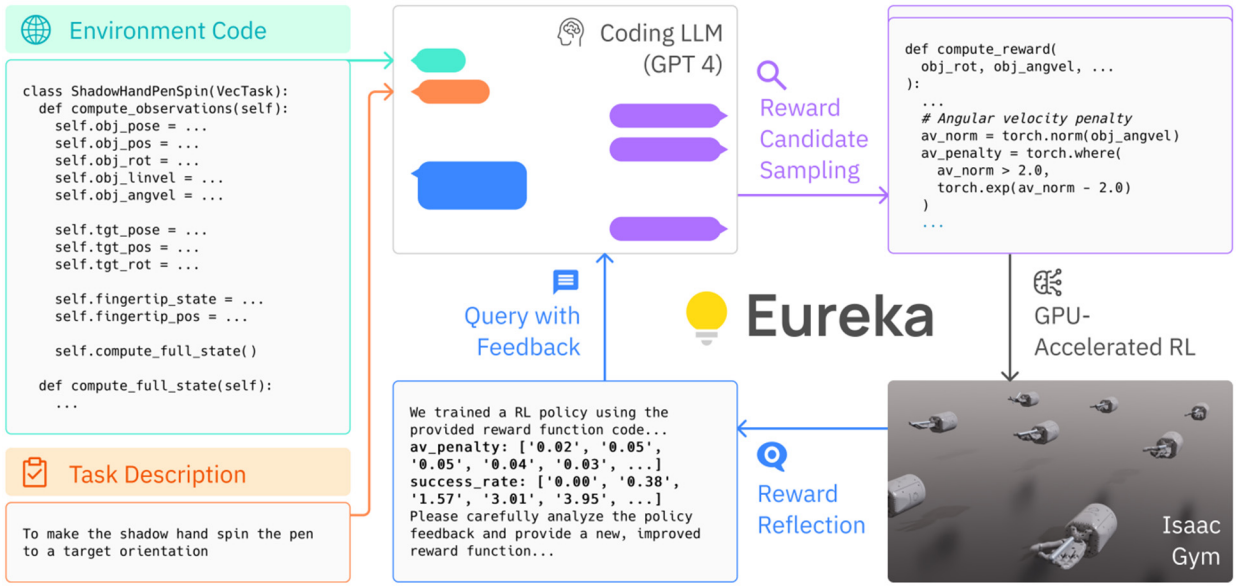

Figure 4. Eureka leverages LLM to generate reward functions for robotic tasks and surpasses expert-designed functions through iterative improvements [11].

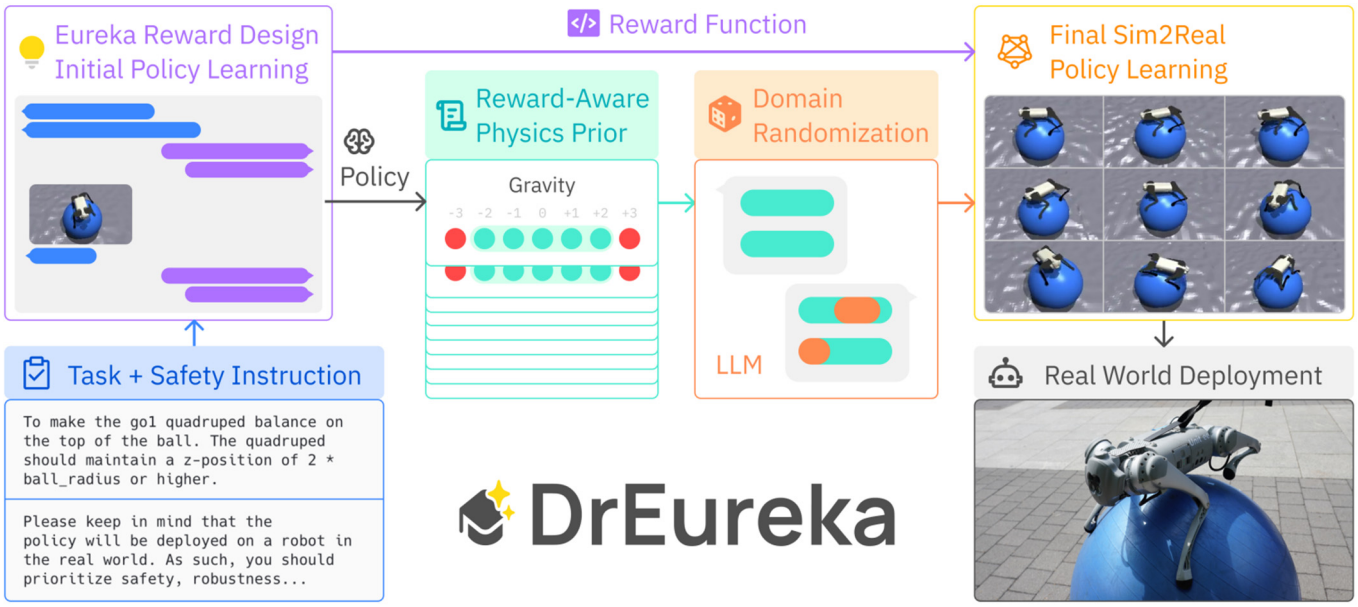

Figure 5. DrEureka leverages LLM to design reward functions and solves the sim-to-real problem through its Reward-Aware Physics Priors mechanism and domain randomization [134].

Xie [136] introduced *Text2Reward*, a framework that automatically generated dense reward functions for reinforcement learning using LLMs. Provided with a goal expressed in natural language, Text2Reward produced executable dense reward functions derived from a compact representation of the environment. This framework generated free-form dense reward codes and delivered performance comparable to or surpassing that of policies trained with expert-designed codes across a variety of tasks, including 17 manipulator-related tasks and six novel locomotion behaviors. Additionally, Text2Reward incorporated user feedback to iteratively enhance the generated reward functions, thereby increasing the success rate of the learned policies.

Di Palo [137] explored the use of LLMs and VLMs to improve reinforcement learning agents' understanding of human intentions. They developed a framework that utilized language as a primary inference tool, investigating how it could address key challenges

in reinforcement learning, such as efficient exploration, data reuse in experience, skill scheduling, and observational learning. This framework employed LLMs and VLMs to address these reinforcement learning challenges by (1) efficiently exploring environments with sparse rewards, (2) reusing collected data to sequentially bootstrap the learning of new tasks, (3) scheduling learned skills for novel tasks, and (4) acquiring knowledge from observing expert agents.

Du [138] developed success detectors that identified whether actions or tasks were successfully completed, utilizing the large multimodal language model Flamingo and human reward annotations. The study on success detection spanned three distinct domains: (1) interactive language-conditioned agents in simulated households, (2) real-world robotic manipulation tasks (inserting and removing small, medium, and large gears), and (3) "in-the-wild" human egocentric videos. These success detectors adapted to new language instructions and visual changes using VLMs such as Flamingo, which were trained on a broad range of language and visual data. Furthermore, success detection was reframed as a VQA problem, enabling the tracking of task progress through multiple frames to ascertain whether tasks had been successfully completed. The proposed method proved to be more accurate in detecting success compared to custom reward models in the first two domains, even with new language instructions or visual changes. However, success detection in unseen real-world videos in the third domain posed a more challenging generalization task, underscoring the need for additional research.

Du [139] introduced the *ELLM* (exploring with LLMs) framework, which provided guidelines for pre-training reinforcement learning using LLMs. ELLM utilized the natural language processing capabilities of LLMs to define goals and furnish reward functions for reinforcement learning agents. This strategy enabled agents to undertake meaningful exploration and learning within their environments. The paper assessed ELLM's performance in two settings: Crafter, a 2D version of Minecraft, and Housekeep, involving the task of rearranging household objects. Experimental results demonstrated that ELLM surpassed other methods in both settings. In the Crafter setting, ELLM attained high performance through goal-oriented learning, proving especially effective in scenarios with sparse reward signals. In the Housekeep setting, the agent conducted sensible exploration by adhering to goals set by the LLM, achieving a high success rate. While the accuracy of goal setting by the LLM varied with the objects and locations, it generally showed high performance. These experimental findings suggested that ELLM was successful in enhancing reinforcement learning performance across diverse environments, highlighting the vital role of providing reward signals based on human commonsense.

4.2. Low-Level Control

Research is also being conducted on generating commands that directly control a robot's actuators (i.e., enabling low-level control) through various applications of LLM models. Among these projects, the Google research team developed RT-1 [8], which consists of film-conditioned EfficientNet-B3, TokenLearner, and Transformer. RT-1 is a model that receives images and natural language instructions at a rate of 3Hz and outputs discretized robot actions. RT-1 was trained on a vast demo dataset with over 130k episodes from more than 700 tests collected over 17 months using 13 robots.

A notable feature of RT-1 is its ability to enhance performance by learning from data gathered from heterogeneous robots or simulations. In the study [8], the authors evaluated the performance of a model trained exclusively on data from the EveryDay Robot (EDR) against a model trained using data from both EDR and Kuka IIWA robots. They recorded a 12% improvement in the bin-picking test. Another experiment compared model performance using data from real environments and simulations for items not encountered in actual settings. The findings indicated that incorporating simulation data in RT-1 training enhances performance over using purely real environment data, suggesting that RT-1 can substantially improve model performance by integrating diverse data from robots of varied morphologies or simulations while sustaining existing task capabilities.

RT-2 [9] is defined as a vision-language-action (VLA) model that facilitates fine-grained control of robots through vision and language commands. RT-2 is fine-tuned with robotic trajectory data based on VLM models such as PaLM-E [140], which has 12 billion parameters and is trained on VQA data, alongside PaLI-X [141], which has parameter sizes ranging from 5 billion to 55 billion. The RT-2 system operates as an integrated closed-loop robotic system that combines low-level and high-level control policies. Despite not explicitly learning certain capabilities during pre-training, RT-2 exhibits improved task performance via real-world generalization involving diverse objects, visual scenes, and instructional contexts. The paper [9] quantitatively assesses RT-2's emergent capabilities in areas such as reasoning, symbol understanding, and human recognition. Furthermore, applying chain-of-thought prompting techniques to RT-2 has proven effective in solving more complex semantic inference tasks, such as using a rock as an improvised hammer or offering an energy drink instead of a carbonated beverage to a thirsty person. In comparison with the earlier study on RT-1, RT-2 demonstrates enhanced performance in both familiar and novel tasks.

AutoRT [10] is a follow-up study based on the research results of RT-1 and RT-2, establishing an orchestration of large-scale robotic agents for data collection in real-world scenarios. AutoRT employed 53 robots to gather 77,000 real robot episodes over seven months through both teleoperation and autonomous robot policies. At the heart of AutoRT is a robust foundation model that generates 'task proposals' based on given visual observations. Notably, AutoRT introduces a 'Robot Constitution' using constitutional prompting to ensure actions during the task proposal process do not compromise the safety of the robot or nearby individuals. This Robot Constitution, inspired by Asimov's three laws [142], comprises basic rules, safety rules that identify unsafe or unwanted tasks, and embodiment rules that clarify the robot's operational boundaries.

AutoRT enhances data collection by initially scanning the surroundings to identify interesting scenes or tasks (exploration). It interprets the given context through a VLM and proposes potential tasks via an LLM (task generation). Subsequently, tasks suggested by the LLM are screened (affordance) to assess their feasibility and the need for human intervention, employing the Robot Constitution. During this procedure, viable tasks are chosen and performed, while pertinent data are gathered (data collection). The collected data are then assessed for (diversity scoring) the visual diversity of the robot trajectories and the linguistic diversity of the language instructions generated by AutoRT (LLM). The aim of this diversity evaluation is to confirm that, unlike simulations, real-world data collection by robots is labor-intensive, making it essential to gather data across a broad spectrum of tasks. Experimental outcomes illustrate that AutoRT achieves higher visual and linguistic diversity compared to RT-1 or BC-Z [143].

Other researchers include Tang [144], who developed an approach that connects natural language user commands with a locomotion controller using foot contact patterns as an interface for low-level commands. This innovative interface translates human commands into the robot's foot contact patterns, allowing the robot to move at a specified speed with precise timing for each foot's contact with the ground. To achieve this, the robot used a cyclic sliding window to extract foot contact flags from a pattern template, thus generating the required foot contact patterns. During training, a random pattern generator created foot contact patterns, and during testing, an LLM translated human commands into these patterns. The robot then adjusted its movements based on the foot contact patterns it learned through deep reinforcement learning, closely adhering to the intended foot contact patterns and speed commands. This approach demonstrated a 50% higher success rate in task evaluation (across 30 tasks, including standing still) compared to two baselines (which employed discrete gaits and sinusoidal functions as interfaces), successfully solving 10 more tasks than the baselines.

Mandi [145] introduced a novel method for multi-robot collaboration that utilizes LLMs for both high-level communication and low-level path planning. In this method, the robots employ the LLM to discuss and reason about task strategies. They generate sub-task

plans and task space waypoint paths, which a multi-arm motion planner then uses to expedite trajectory planning. Additionally, environmental feedback, such as collision detection, prompts the LLM agent to refine plans and waypoints contextually. This method achieved a high success rate across all tasks in the RoCoBench (including duties such as sweeping the floor), effectively adapting to variations in task semantics. In real-world experiments, specifically the block-sorting task, RoCo demonstrated its ability to communicate and collaborate with other robot agents to successfully complete the tasks.

Wang [146] proposed a novel paradigm for utilizing few-shot prompts in physical environments. This method involved gathering observation and action pairs from existing model-based or learning-based controllers to form the initial text prompts. Data included sensor readings, such as IMU and joint encoders, coupled with target joint positions. These data formed the starting input for LLM inference. As the robot interacted with its environment and collected new observational data, these initial data were updated with outputs from the LLM. In the subsequent prompt engineering phase, observation and action pairs, along with explanatory prompts, were crafted to enable the LLM to function as a feedback policy. The explanatory prompts provided clear descriptions of the robot walking task and control design details, while the observation and action prompts delineated the format and significance of each observation and action. This method allowed the LLM to directly output low-level target joint positions for robot walking. The approach was tested using the ANYmal robot in MuJoCo and Isaac Gym simulators for robot walking, indicating that the LLM could act as a low-level feedback controller for dynamic motion control within sophisticated robot systems.

Liang [147] introduced a new framework named *Code as Policies* (CaP) that directly constructs robot policies from executable code generated by a code LLM. This framework enabled the interpretation and execution of natural language instructions through an LLM, supporting the creation of high-level policies for robots and accommodating a variety of robotic tasks. Specifically, CaP interpreted natural language instructions through descriptions and formulated an action plan for the robot. Moreover, it utilized VLMs such as ViLD and MDETR to identify objects and ascertain their locations. Based on this information, the framework controlled the robot's movements to carry out specified tasks. The paper demonstrated the CaP framework across diverse domains, including whiteboard drawing, tabletop manipulation, and mobile robot navigation and manipulation. Experimental results showed that CaP achieved similar or better success rates than existing systems such as CLIPort, displaying notably strong generalization capabilities for new tasks. These findings underscored the flexibility and efficacy of the CaP framework, establishing its effectiveness across various robotic systems.

Mirchandani [148], shown in Figure 6, suggested that pre-trained LLMs could autoregressively complete complex token sequences and function as general sequence modelers through in-context learning without needing additional training. Expanding on this concept, the study evaluated LLMs' ability to operate as pattern machines in three domains: sequence transformation, sequence completion, and sequence improvement. In sequence transformation, the research demonstrated that LLMs could generalize specific sequence transformations using benchmarks such as ARC (abstraction and reasoning corpus) and PCFG (probabilistic context-free grammar), thereby proving their utility in spatial reasoning tasks for robotics. In sequence completion, the study examined whether LLMs could finish patterns in elementary functions (e.g., sinusoids), illustrating their utility in robotic tasks such as extending a wiping motion from kinesthetic demonstrations or creating drawings on a whiteboard. Finally, in sequence improvement, the research revealed that by utilizing reward-labeled trajectories as context and incorporating online interaction, LLM-based agents could explore small grids and refine simple trajectories using human-in-the-loop methods, such as optimizing a CartPole controller.

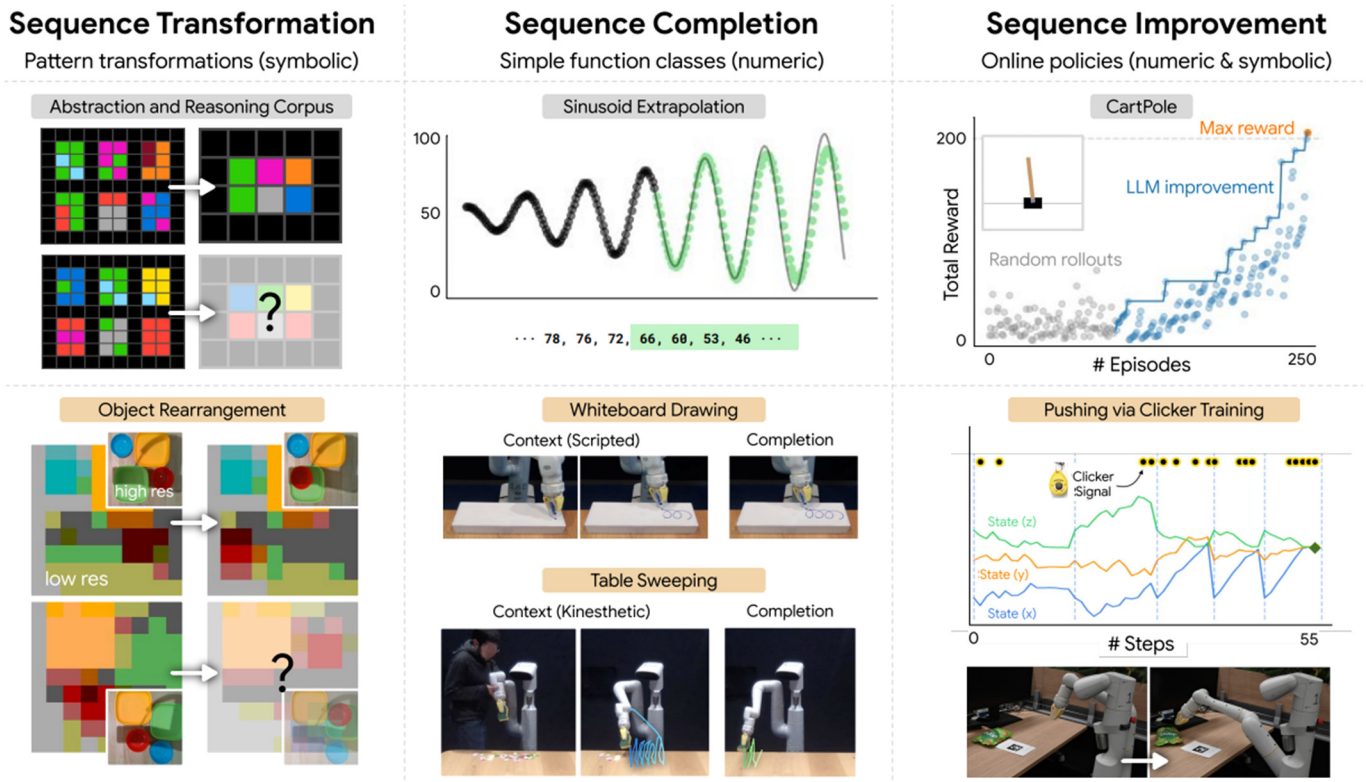

Figure 6. Pre-trained LLMs can act as general sequence modelers, and their abilities were assessed in sequence transformation, completion, and improvement [148].

4.3. High-Level Planning (Including Decision-Making and Reasoning)

The abstraction and generalization capabilities of LLMs offer effective methodologies for high-level planning tasks in robotic systems. Leveraging these capabilities, various research outcomes have been realized in the fields of planning, decision-making, reasoning, and behavior trees within robotics.

Yoneda [149] introduced Statler, a framework designed to provide LLMs with an explicit world state representation through a continuously maintained 'memory'. The core of Statler consisted of two components: the world model reader and the world model writer. These components interacted with and sustained the world state. The world model reader interpreted user commands and generated executable code based on the current state representation, while the world model writer updated the system's state according to execution outcomes. By facilitating access to the world state 'memory', Statler improved LLMs' ability to reason about planning tasks with extended time horizons, overcoming limitations imposed by context length.

Mu [150], shown in Figure 7, introduced EmbodiedGPT, a model specifically designed for Embodied AI, which leverages LLMs. This framework processes visual observations and natural language to establish long-term plans and execute tasks in real-time. EmbodiedGPT utilizes pre-trained vision transformers and the LLaMA language model to encode visual features and map them to the language modality. The generated plan was subsequently converted into specific task commands using general visual tokens, encoded by the vision model. The framework's functionality comprises (1) encoding current visual features, (2) mapping visual features to the language modality via attention-based interactions between visual tokens and text queries or learnable embedded queries, (3) generating plans with the LLaMA language model and translating them into specific task commands, and (4) querying the encoded visual tokens from the vision model and translating them into low-level control commands through a downstream policy network for task execution. Experimental results, utilizing the MS-COCO dataset, revealed that EmbodiedGPT excels in object recognition and understanding spatial relationships. Notably, implement-

ing a closed-loop design and a "chain-of-thought" training mode significantly enhanced EmbodiedGPT's performance. These results demonstrate that EmbodiedGPT effectively handles various autonomous tasks, exhibiting superior capability in object recognition, understanding spatial relationships, and generating logical, executable plans.

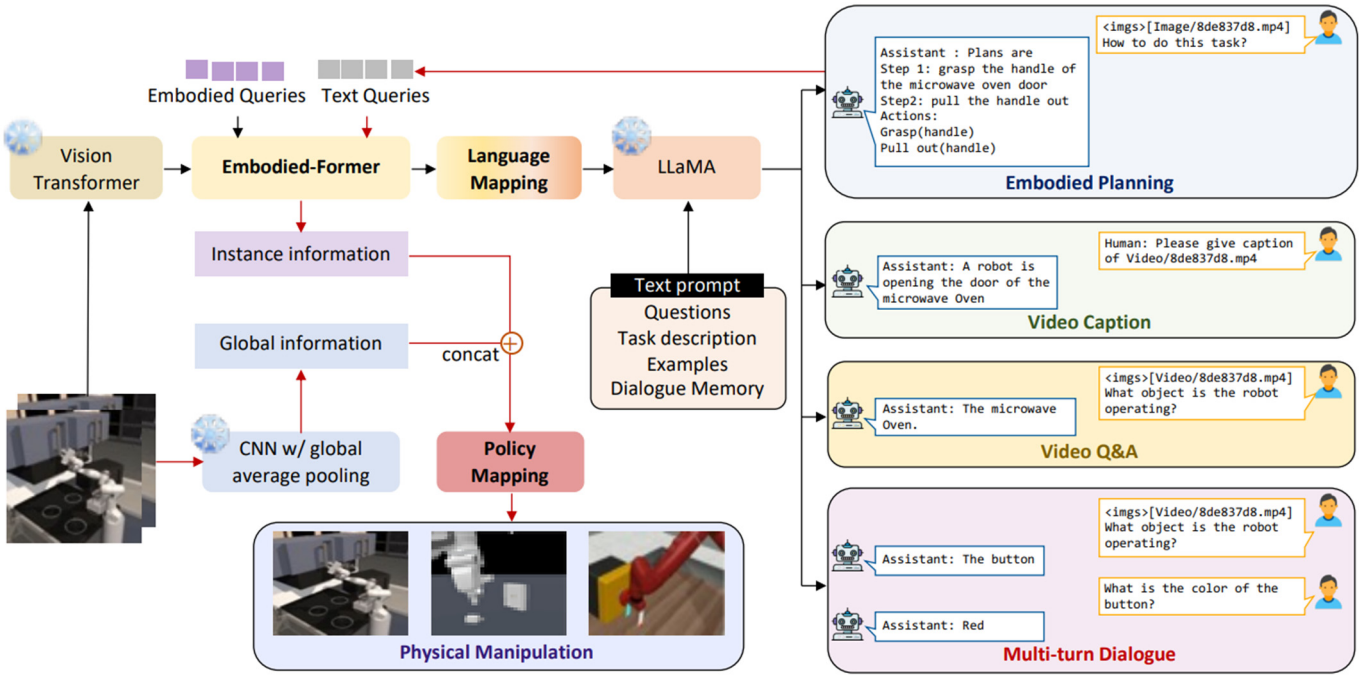

Figure 7. After encoding visual features, they are mapped using visual tokens and text queries. A plan is then created with the LLaMA model and turned into task commands. The visual tokens are queried and converted into low-level control commands to perform the task [150].

Chen [151] introduced the language-model-based commonsense reasoning (LMCR) framework to assist robots in comprehending incomplete natural language instructions. This framework enabled robots to receive instructions in natural language from humans, observe their surroundings, and employ a commonsense reasoning method to autonomously infer missing information. LMCR utilized a model of commonsense reasoning learned from web-based text materials, allowing robots to understand incomplete instructions and autonomously execute tasks. The framework comprised three main functions: language understanding, commonsense reasoning, and action planning. In language understanding, LMCR translated human natural language instructions into a form interpretable by robots, parsing them into verb frames to convert them into executable structures. During the commonsense reasoning phase, the robot analyzed surrounding objects and employed a language model trained on large-scale unstructured text materials to fill in the missing details from the instructions. This model identified the most suitable verb frame to complete the gaps. Subsequently, based on the completed verb frame, the robot formulated its actions using predefined action plans for each verb to guide the movements of the robot arm and execute the assignment. Experimental results showed that LMCR demonstrated superior generalization performance for novel concepts not presented in the training set and surpassed GCNGrasp, which depends on a predefined graph structure for all concepts and their relationships. This indicated that LMCR was an effective tool, combining the semantic reasoning capabilities of language models with planning that adapted to the robot's specific environment and context, effectively managing complex and prolonged tasks.

Huang [152] introduced a methodology named grounded decoding (GD), which offers a method for generating LLM-based robot action plans. These plans enable robots to execute long-term tasks across diverse physical environments. The methodology encompasses two primary elements: linking the text generated by the language model to actionable task

commands in the physical world via GD and adjusting the tokens generated by the LLM to real-world conditions to formulate feasible commands. This approach synergizes the high-level semantic reasoning of LLMs with plans that are aligned with the robot's physical environment and capabilities, thus facilitating the execution of complex and long-term tasks. The method addresses several limitations robots face in performing complex, long-term tasks, such as a lack of physical world experience, an inability to process non-verbal cues, and a disregard for necessary robotic constraints such as safety and rewards. The paper details experiments in a simulated tabletop rearrangement, a mini-grid 2D maze, and real-world kitchen mobile manipulation settings to evaluate long-horizon reasoning performance. Comparative experiments with SayCan revealed that while SayCan limits the range of robot actions, GD can represent a wider array of actions. In contrast to CLIPort, which executes high-level language instructions directly, GD achieves enhanced performance through detailed, step-by-step planning.

Huang [153], as shown in Figure 8, proposed the inner monologue method, which allowed LLMs to plan and adjust based on feedback from the environment. This approach enabled robots to formulate plans in dynamic environments, retry upon facing failure, or seek human feedback to refine their strategies. The author clarified that this method emerged from integrating the LLM's high-level planning capabilities with perceptual feedback and low-level control, thereby facilitating more adaptable and intelligent interactions. Inner monologue integrated various feedback sources into the language model to assist the robot in executing given instructions, including text-based indicators of the robot's action success or failure, object recognition and descriptions within the scene, the robot's ability to ask questions to gather additional information, breaking down instructions into multiple steps to establish an execution plan, and enabling the robot to interact with humans to execute and refine the instructions. The inner monologue method was evaluated in both simulated and real-world environments, such as tabletop rearrangement tasks and manipulation tasks in a real kitchen. The results showed that inner monologue was an effective framework, enabling robots to act intelligently in complex interactive settings by effectively integrating environmental feedback to plan and execute tasks.

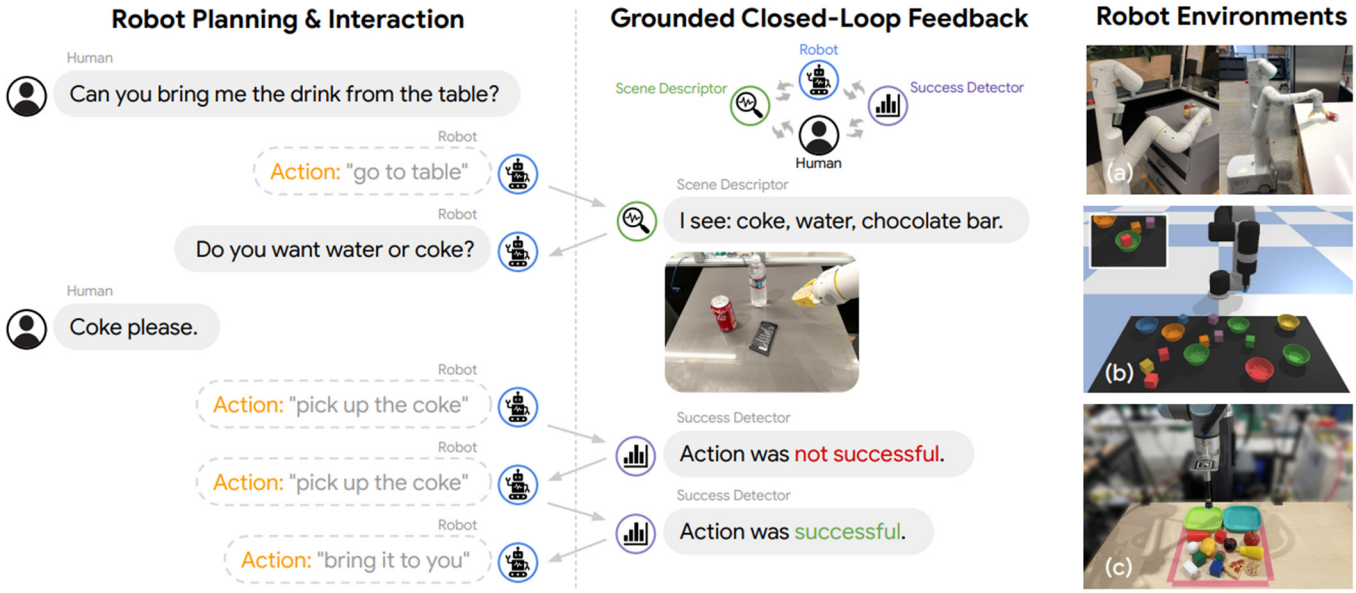

Figure 8. Inner Monologue integrates various feedback sources into the language model to enable robots to carry out instructions: (**a**) mobile manipulation and (**b**,**c**) tabletop manipulation, in both simulated and real-world environments [153].

Lykov [154] introduced a novel approach to autonomous robot control named LLM-BRAIn, which facilitated the command-based generation of robot behaviors. LLM-BRAIn, a transformer-based LLM, fine-tuned the Stanford Alpaca 7B model to generate robot behavior trees (BTs) from textual descriptions. The developed model was compact enough to operate on a robot's onboard microcomputer, while adept at constructing complex robot behaviors. It provided structurally and logically correct BTs and demonstrated the ability to handle instructions that were not included in the training set.

Song [155], as shown in Figure 9, proposed LLM-Planner, a system designed for few-shot planning in embodied agents. LLM-Planner processed natural language instructions to generate high-level plans, selected subgoals from these plans, and identified actions via a low-level planner. It continuously updated environmental information as new objects were detected during action implementation and revisited the LLM to adjust the plan if subgoals failed or were delayed based on updated observations. This iterative process was repeated until the subgoal was achieved, after which the system moved to the next goal. Compared to traditional models such as HLSM and FILM, LLM-Planner demonstrated competitive performance with significantly reduced training data and proved its ability to generalize in various tasks (e.g., ALFRED) with minimal examples.

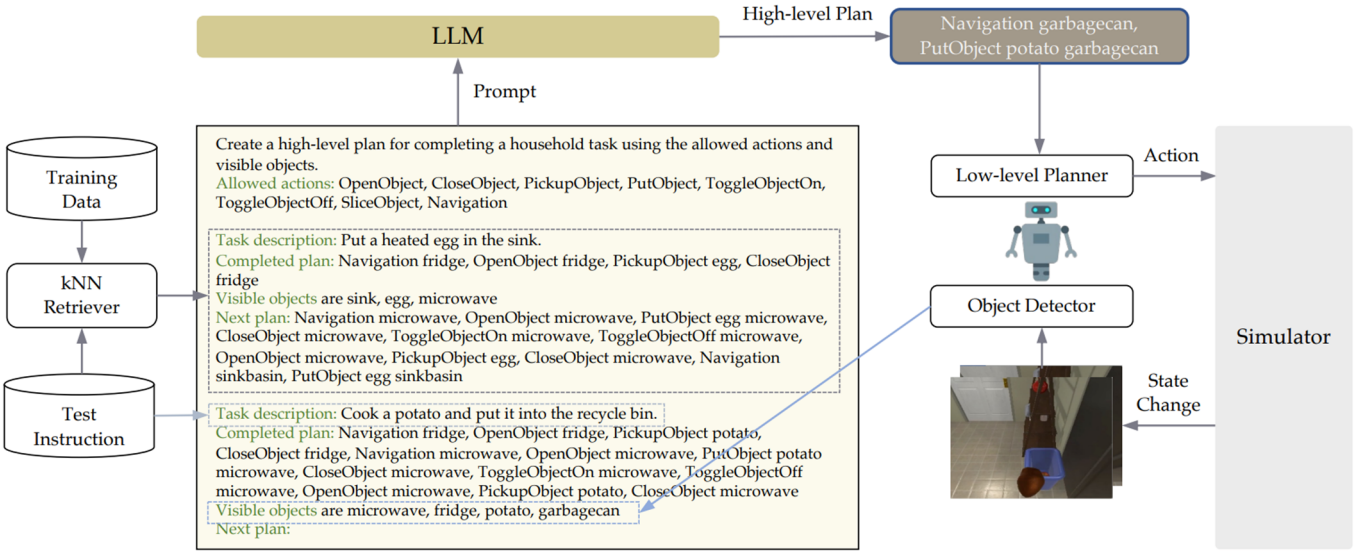

Figure 9. LLM-Planner is a system that creates high-level plans based on natural language commands, sets subgoals to determine actions, and continuously updates the plan to reflect environmental changes [155].

Singh [156], as shown in Figure 10, introduced ProgPrompt, a programmatic LLM prompt structure designed for generating plans across diverse situated environments, robot capabilities, and tasks. ProgPrompt functioned as a robot task-planning system that leveraged LLMs and included a Python programming structure to facilitate information about the environment and executable actions. It featured a feedback mechanism, using executable program plan examples and assertion statements to mitigate errors, enhancing task success rates. Additionally, ProgPrompt verified the current state through environmental feedback during plan execution and revised the plan accordingly. The results indicated that the integration of programming language features substantially improved task performance in contexts such as VirtualHome and real-world manipulation tasks in terms of success rate, goal conditions recall, and executability.

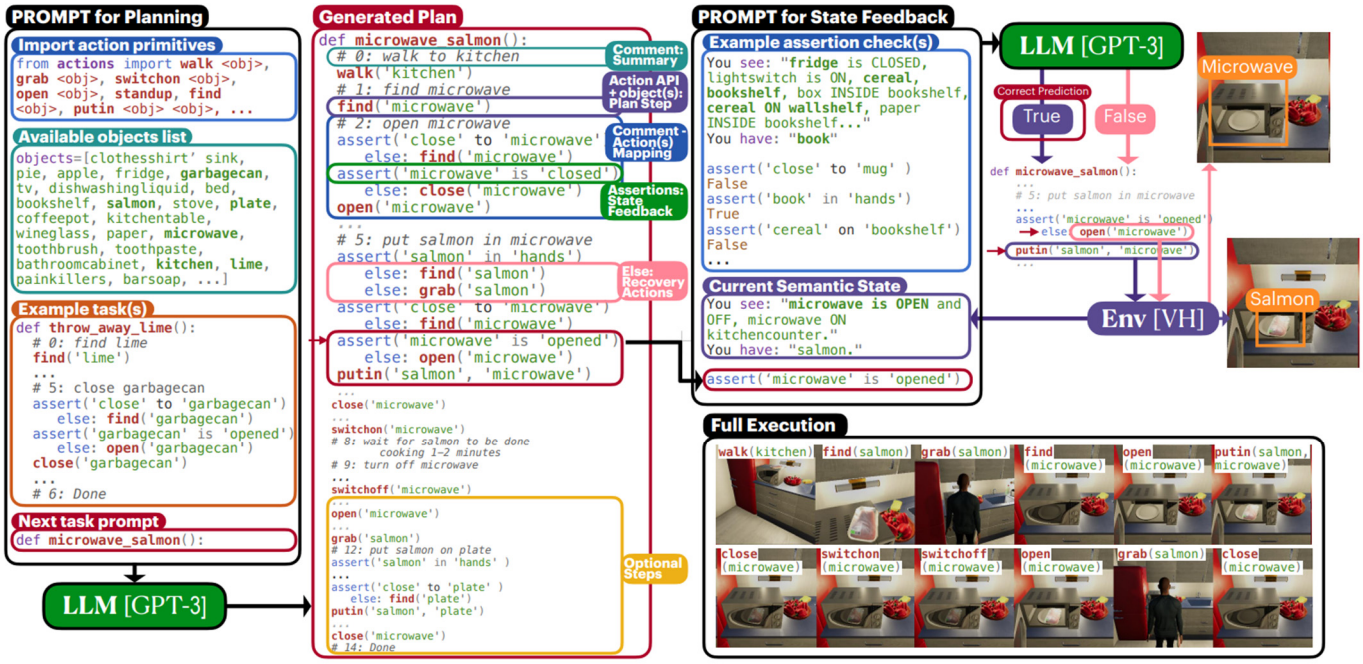

Figure 10. ProgPrompt is a system that uses Python programming structures to provide environmental information and actions, enhancing the success rate of robot task planning through an error recovery feedback mechanism and environmental state feedback [156].

Rana [157] introduced SayPlan, a scalable method for large-scale task planning using LLMs and based on a 3D Scene Graph (3DSG) representation. SayPlan involved the LLM searching a collapsed 3D scene graph and task instructions to identify all relevant items and then locating the subgraph that contained the necessary items to complete the task. The identified subgraph was subsequently used by the LLM to generate a high-level plan that addressed the navigational aspect of the task. This plan was formatted as a JSON 3D scene graph and subjected to a repetitive replanning process through feedback from the scene graph simulator and a set of API calls for manipulation and operation until an executable plan was determined. SayPlan was tested in two large-scale environments, featuring up to three floors, 36 rooms, and 140 assets and objects, proving its capability to ground large-scale and long-horizon task plans from abstract and natural language instructions, thereby enabling a mobile manipulator robot to execute these tasks.

Zeng [158], as shown in Figure 11, proposed the Socratic model (SM), a modular framework that synergistically utilizes various forms of knowledge and employs multiple pre-trained models to exchange information and leverage new multimodal capabilities. SM operates without fine-tuning by integrating diverse pre-trained models and functions in a zero-shot approach (e.g., using multimodal prompts), which enables it to harness new multimodal capabilities. SM demonstrated state-of-the-art performance in zero-shot image captioning and video-to-text retrieval, and it effectively answered free-form questions about egocentric video. Additionally, it supported interactions with external APIs and databases (e.g., web search) for multimodal assistive dialogue, robot perception, and planning, among other novel applications.

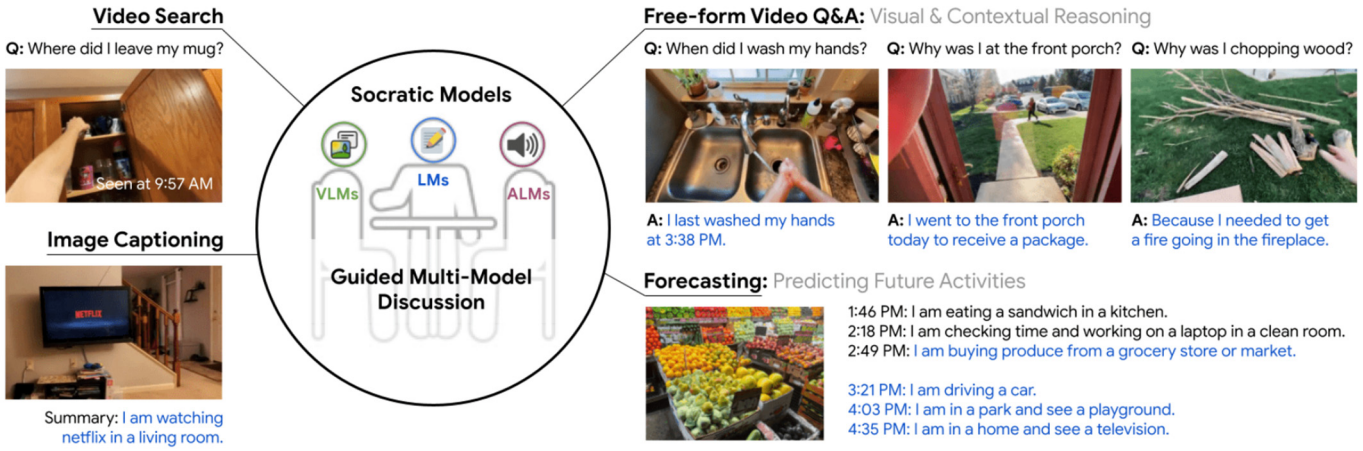

Figure 11. SM integrates various types of knowledge by using multiple pre-trained models and provides meaningful results even in complex computer vision tasks such as image captioning, context inference, and activity prediction [158].

Lin [159] introduced Text2Motion, a language-based framework designed to handle sequential manipulation tasks that require long-horizon reasoning. Text2Motion interpreted natural language instructions to formulate task plans and generated multiple candidate skill sequences, evaluating the geometric feasibility of each sequence. By employing a greedy search strategy, it selected the optimal skill sequence to verify and execute the final plan. This method enabled Text2Motion to perform complex sequential manipulation tasks with a higher success rate compared to existing language-based planning methods, such as Saycan-gs and Innermono-gs, and provided semantically generalized characteristics among skills with geometric relationships.

Wu [160] investigated personalization in-home cleaning robots that organize and tidy spaces, using an LLM to convert user-provided object placement locations into generalized rules. By using a camera to identify objects and CLIP to categorize them, TidyBot efficiently relocated objects according to these rules. This method attained an impressive accuracy of 91.2% for unseen objects in a benchmark dataset, which encompassed a variety of objects, receptacles, and example placements of both "seen" and "unseen" objects across 96 scenarios. Additionally, it achieved an 85% success rate in removing objects during real-world tests.

4.4. Manipulation by LLMs

In robotics research, the manipulation domain, which includes robotic arms and end effectors, encompasses various areas that benefit from foundation models such as LLMs for language-based interactions and VLMs for object handling. Among the studies integrating manipulation with foundation models, Stone [161] introduced an approach called manipulation of open-world objects (MOO). This approach determined whether a robot could follow instructions involving unseen object categories by linking pre-trained models to robotic policies. MOO utilized pre-trained vision-language models to derive object information from language commands and images, guiding the robot's actions based on the current image, command, and identified object data. Experimental use of real mobile manipulation robots showed that MOO could adapt to new object types and environments in a zero-shot fashion. Moreover, MOO responded to non-verbal cues such as pointing at specific objects, extending its scope to open-world exploration and manipulation.

Existing VLMs often lack a comprehensive understanding of physical concepts such as material and fragility, which limits their effectiveness in robotic manipulation tasks. To address this issue, Gao [162] introduced PhysObjects, an object-centric dataset featuring 39.6K crowd-sourced annotations and 417K automated annotations of physical concepts. The automated annotations involved assigning specific concept values to predefined object categories or continuous concepts such as material and fragility. Fine-tuning a VLM on PhysObjects enhanced comprehension of physical concepts by capturing human biases related to the visual appearance of objects. Integrating this physically grounded VLM with an LLM-based robotic planner framework improved performance in tasks requiring reasoning about physical concepts.

The traditional pre-training and fine-tuning pipeline often suffers from decreased learning efficiency and challenges in generalizing to unseen objects and tasks due to its reliance on domain-specific action information and domain-general visual information. To address these limitations, Wang [163] proposed a modular approach named ProgramPort, which utilizes the syntactic and semantic structure of language instructions. Wang's framework incorporated a semantic parser to reconstruct executable programs, composed of functional modules based on vision and action across multiple modalities. Each functional module combined deterministic computation with learnable neural networks. Program execution involved generating parameters for general manipulation primitives used by the robot's end effector. The entire module network was trainable with an end-to-end imitation learning objective. Experimental results demonstrated that the model effectively separated action and perception, achieving enhanced zero-shot and compositional generalization across various manipulation tasks, specifically 16 tasks related to robot manipulation.

Ha [164] proposed a framework aimed at robot skill acquisition. This framework provided a comprehensive solution by utilizing language guidance, without necessitating expert demonstrations or reward specification/engineering. It consisted of two main components. The first component, scaling up language-guided data generation, employed LLMs to break down tasks into subtasks and generate a hierarchical plan or task tree. This plan was materialized into various robot trajectories using 6-DoF exploration primitives. These trajectories were subsequently verified and retries were performed as needed until success was achieved. This approach enhanced the success rate of data collection and more effectively mitigated the low-level understanding gap in LLMs by incorporating retry processes as part of the robot's experiences. The second component, distilling down to language-conditioned visuomotor policy, transformed robot experiences into a policy that deduced control sequences from visual observations and natural language task descriptions. By extending diffusion policies, this component handled language-based conditioning for multi-task learning. To assess long-horizon behavior, commonsense reasoning, tool use, and intuitive physics, a new multi-task benchmark comprising 18 tasks related to robot manipulation across five domains (mailbox, transport, drawer, catapult, and bus balance) was developed. This benchmark effectively supported the learning of retry behaviors in the data collection process and enhanced success rates.

Huang [165], as shown in Figure 12, aimed to synthesize dense robot trajectories, including 6-DoF end-effector waypoints, for various manipulation tasks using an open set of instructions and objects. Huang noted that LLMs were skilled at deriving affordances and constraints from free-form language instructions. Further, by harnessing code generation capabilities, Huang developed 3D value maps for the agent's observation space through interactions with VLMs. These 3D value maps were integrated into a model-based planning framework to generate closed-loop robot trajectories robust to dynamic perturbations in a zero-shot approach. The proposed framework demonstrated efficient learning of the dynamics model for scenes with contact-rich interactions and provided advantages in these complex scenarios.

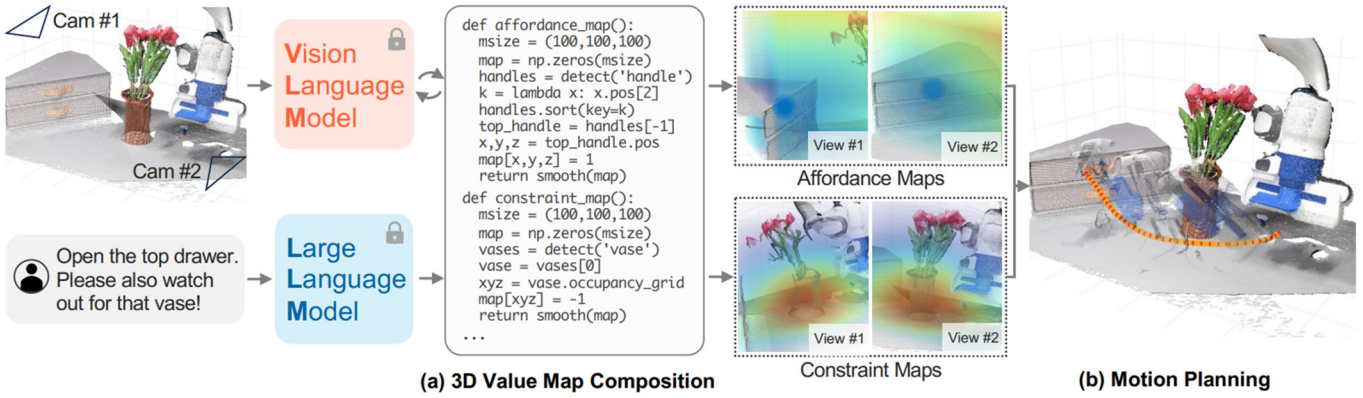

Figure 12. Based on language instructions and RGB-D data, the LLM interacts with the VLM to generate 3D affordance and constraint maps and design robot trajectories without additional training [165].

Ahn [166] introduced a framework named SayCan, which integrates LLMs with reinforcement learning value functions, enabling robots to follow high-level text instructions. SayCan comprises two primary components: Say, which uses an LLM for task-based decision-making, and Can, which evaluates the feasibility of these decisions via reinforcement learning. Say leverages task-based knowledge from the LLM and reinforcement learning functionality to assess the feasibility of task execution by robots in real-world scenarios. The LLM determines the actions necessary to achieve high-level goals and evaluates the effectiveness of each action in fulfilling the instructions. Learned through reinforcement learning, the affordance function estimates each action's success probability in the current state, confirming the executability of actions proposed by the LLM. This process allows the LLM to assess the robot's current state and capabilities, ultimately generating an interpretable action plan. SayCan was evaluated across 101 robot tasks, achieving an 84% plan success rate and a 74% execution success rate in a simulated kitchen environment. In a real kitchen setting, the plan success rate decreased slightly to 81% and the execution success rate fell to 60%, demonstrating that the policy and value functions generalize well to real-world settings.

Huang [167] introduced the Instruct2Act framework, which employs LLMs to sequentially map multi-modality instructions to robot actions. The previous method, CaP, generated robot policy program code directly from in-context examples based on language instructions. However, this approach was constrained by the capabilities of the generated code and encountered difficulties with longer, more complex commands due to the required high precision of code. To overcome these limitations, Instruct2Act introduced a novel strategy that used multi-modality models and LLMs to simultaneously address recognition, task planning, and low-level control modules. Instruct2Act utilized the segment anything model for identifying potential objects in input images for multi-modality recognition and the CLIP model for object classification. As a result, Instruct2Act developed an integrated search system capable of managing various input modalities and instruction types, including both pure language instructions and combined language-visual instructions, facilitating the integration of diverse instruction types into a unified architecture. Moreover, for pointer-language instructions, the framework supported task segmentation based on the user's clicks.

4.5. Scene Understanding in LLMs and VLMs

To address the VQA problem, robotics research increasingly uses pre-trained VLMs to derive high-level information from visual data. This method is advantageous for scene understanding as it helps determine affordances that describe the relationship between the current state and the next action based on images from cameras. Related studies focus on aspects of scene understanding.

Chen [168] explored methods to integrate commonsense into scene understanding using LLMs and introduced three paradigms for classifying room types within indoor environments based on included objects. The zero-shot approach utilized a pre-trained language model to identify the objects in a room and estimate their types. The feed-forward classifier approach involved inputting sentences that listed a room's objects into the language model to generate embedding vectors, which were subsequently input into a pre-trained shallow multilayer perceptron to predict each room type. Lastly, the classifier approach embedded images of rooms alongside textual descriptions to identify the best-matching description, thereby determining the room type. These paradigms demonstrated the capacity to generalize to objects not presented in the training set and to make inferences within a space larger than that defined by the trained object labels.

Yang [169] introduced the innovative zero-shot, open-vocabulary, LLM-based 3D visual grounding pipeline called LLM-Grounder. This method breaks down complex natural language queries into semantic components and uses visual grounding tools such as OpenScene or LERF to locate objects within 3D scenes. Subsequently, the LLM evaluates spatial and commonsense relationships among these objects to achieve the final grounding. Remarkably, LLM-Grounder operates without labeled training data and has proven its capacity to adapt to new 3D scenes and diverse text queries, enhancing grounding capabilities for complex language queries and establishing itself as an effective solution.

Chen [170] developed NLMap, an open-vocabulary, queryable scene representation system. Designed to accumulate and incorporate contextual data within a scene representation for natural language queries, this system allows an LLM planner to visualize and query objects, thereby generating contextual plans. Initially, a VLM sets up a scene representation for natural language queries; then, an LLM-based object suggestion module reviews instructions, suggests relevant objects, and queries the scene for object availability and location. Using this information, the LLM planner devises plans uniquely tailored to the scene's context. NLMap equips robots with the ability to function without a predefined catalog of objects or actions, overcoming the constraints of earlier methods and enabling more adaptable operations in environments with novel or absent objects.

Elhafsi [171] introduced a monitoring framework that employed an LLM with superior contextual understanding and reasoning capabilities to detect edge cases and anomalies within vision-based policies. This framework monitored the robot's perception stream through an LLM-based module, designed to detect semantic anomalies that might occur during operations. By converting the robot's visual observations into textual descriptions at regular intervals and integrating these into LLM prompts, it could pinpoint factors leading to policy errors, unsafe behavior, or task confusion. The conversion of visual information into natural language descriptions used various techniques, without restriction to any specific method. This flexibility enabled both fully end-to-end policies and classical autonomy stacks using learned perception to align more closely with human intuition. The findings indicated that semantic anomalies did not always correspond to semantically explainable failures, and end-to-end policies could sometimes behave unpredictably.

Hon [172] introduced a new model family named 3D-LLM, which incorporated 3D world information into LLMs. The 3D-LLM model utilized 3D point clouds and their features as input, enabling it to handle a variety of spatially aware 3D tasks. These tasks included 3D captioning, dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialogue, and navigation. The model used a 3D feature extractor to align 3D features from multi-view images with language features, facilitating more precise text generation and question answering based on spatial understanding. To train 3D-LLM, a pre-trained 2D VLM formed the backbone, enhanced by the addition of 3D positional embeddings to better capture 3D spatial information. The model generated location tokens through linguistic descriptions of specific objects and was trained using 3D features as input. Experimental results showed that 3D-LLM excelled in various 3D-related tasks, achieving approximately a 9% higher BLEU-1 score compared to previous models on the ScanQA dataset. It demonstrated superior performance in 3D captioning, task composition, and

3D-assisted dialogue, outperforming 2D VLMs and displaying an improved understanding of object locations, shapes, and interactions.

In the extension of scene understanding using VLMs, the keyword VLN (vision-and-language navigation) is widely used in navigation-related research, where language foundation models are increasingly utilized.

Shah [173], as shown in Figure 13, introduced a robotic navigation system named LM-Nav, which capitalized on the advantages of training with large, unlabeled trajectory datasets while providing a high-level interface for users. LM-Nav utilized three large-scale pre-trained models: ViNG, CLIP, and GPT-3. Initially, the LLM translated natural language instructions into a sequence of textual landmarks. The VLM integrated these textual landmarks with images to identify the relevant images through probabilistic distribution. Subsequently, the VNM utilized these landmarks to plan and execute robot trajectories within the environment. During this process, the robot utilized a graph search algorithm to determine optimal trajectories and to navigate along these paths in the real world. This method demonstrated LM-Nav's ability to perform long-horizon navigation in complex outdoor environments using natural language instructions.

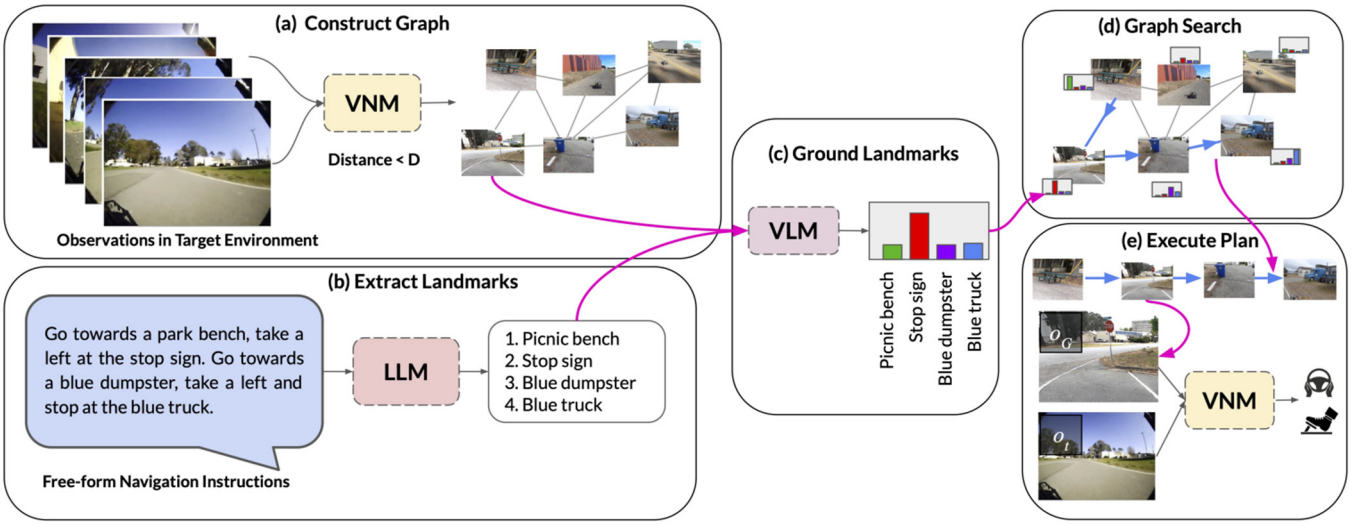

Figure 13. LM-Nav uses three pre-trained models: (**a**) VNM builds a topological graph from observations, (**b**) LLM converts instructions into landmarks, (**c**) VLM matches landmarks to images, (**d**) A graph search algorithm then finds the best robot trajectory, and (**e**) the robot executes the planned path [173].

Zhou [174] introduced NavGPT, an LLM-based navigation agent designed to follow instructions. NavGPT is a vision-language navigation system that employs an LLM to translate visual inputs from a visual foundation model (VFM) into natural language. The LLM then interprets the current state and makes informed decisions to reach the intended goal, based on these converted visuals, navigation history, and potential future routes. NavGPT conducts various functions, including high-level planning, decomposing instructions into sub-goals, identifying landmarks in observed scenes, monitoring navigation progress, and modifying plans as necessary. Although NavGPT's performance on zero-shot tasks from the R2R dataset has not yet matched that of trained models, it underscored the potential of utilizing multi-modality inputs with LLMs for visual navigation and tapping into the explicit reasoning capabilities of LLMs to enhance learned models.

Huang [175] introduced VLMaps, a spatial map representation that integrates pre-trained vision-language features with a 3D reconstruction of the physical world. VLMaps, when combined with an LLM, translate spatially organized sequences of open-vocabulary navigation goals (e.g., "between the sofa and the TV") into natural language commands. These commands can be directly localized on a map and generate new obstacle maps in real-

time, facilitated by sharing among various robot types. Extensive experiments conducted in both simulated environments (using the Habitat simulator with the Matterport3D dataset and the AI2THOR simulator) and real-world settings (with the HSR mobile robot for indoor navigation) demonstrated that VLMs can navigate based on more complex language instructions than previous methods. The reviewed papers in this study are summarized in Table 5.

Table 5. Summary of the reviewed papers in this study.

Name	Explanation	Ref.
Reward Design in RL	• Eureka automatically generates and improves reward functions based on the virtual environment source code. • DrEureka builds reward-aware physics priors using Eureka and supports effective operation in the real world through domain randomization. • LLMs design and refine reward functions based on natural language input. • LLMs and VLMs integrate multimodal data to generate reward functions.	[11,134,136–139,176–180]
Low-level Control	• Generating commands to control actuators capable of low-level control. • RT-1 and RT-2 enable robots to perform complex tasks based on language-vision data. • AutoRT establishes a system where robots can autonomously collect and utilize data.	[8–10,144–148,181–183]
High-level Planning	• LLMs provide an effective methodology for tasks related to high-level planning within robotic systems. • By using natural language, LLMs can formulate plans to solve tasks that require long-horizon reasoning. • LLMs assess the feasibility of actions to determine and execute the optimal robotic behavior. • LLMs generate behavior trees to structure complex robotic actions accurately.	[149–160,184–207]
Manipulation	• Using LLMs and VLMs to integrate language and vision data allows various manipulations. • LLMs interpret high-level instructions to generate the necessary robot actions and assess their feasibility. • VLMs extract object information from images to assist in performing manipulations.	[161–167,208–215]
Scene Understanding	• To solve VQA problems, use VLMs to extract high-level information from vision data. • For scene understanding, estimate and identify objects and evaluate relationships between objects. • For navigation, convert natural language instructions and combine them with vision data to identify the image through probability distributions.	[168–175,216–223]

5. Discussion and Future Directions

The review revealed two potentials of foundation models: (1) commonsense reasoning for planning and (2) the ability to generate code.

The first finding from this review study is the potential to enhance robot intelligence through foundation models. Beyond the studies mentioned here, numerous recent studies have shown that pre-trained models such as LLMs and VLMs can enhance various aspects of robot intelligence, such as situational awareness, high-level task planning, and human interaction. LLMs allow communication with humans in natural languages, object

utilization based on extensive information, and high-level planning using that information. VLMs can describe tasks in text and understand visual information. Furthermore, the information from VLMs can be supplemented by connecting to knowledge databases via LLMs. These capabilities are crucial for enhancing robot intelligence, broadening the scope of robot applications, and maximizing robot utility.

The second finding is the code generation capability of LLMs, which has the potential to automate the robot development process traditionally performed by humans. Additionally, robots that can autonomously update their own algorithms are no longer just science fiction. Although limitations exist for robots to self-update, frameworks such as Eureka and DrEureka, which automatically enhanced reinforcement learning performance for robot motion control, demonstrate the potential for future advancements. This suggests that LLMs may not only enhance human interactions but could also pave the way for self-improvement without human intervention.

While foundation models offer considerable potential for advancing robotics intelligence, several limitations and future considerations remain. These include (1) the speed of inference required for real-time applications, (2) the computational efficiency necessary for embedded systems, (3) the ability to handle multi-modality information, and (4) the necessity of addressing safety and ethical considerations.

First of all, LLMs and VLMs hold considerable potential for enhancing robot intelligence. Nonetheless, several critical issues remain to be addressed. Foundation models, characterized as large-scale models pre-trained on extensive datasets, face challenges related to real-time requirements and limited computational resources in robotic applications. Moreover, concerns such as personal information protection, privacy, and security from external attacks need resolution to enable cloud-based LLMs for robotics.

Secondly, to enhance the computational efficiency and usability of LMs, there is ongoing research into small language models (SLMs). Despite having fewer parameters, SLMs can achieve performance comparable to LLMs in specific applications. Several SLMs have been introduced, including DistilBERT [224], which is a compact version of Google's BERT; Phi-3 [61], another SLM; Florence-2 [84], a small VLM model from Microsoft; MobileBERT [225], which is optimized for mobile platforms; and compact open-source versions of OpenAI's GPT models such as GPT-Neo [226] and GPT-J [227]. Generally, SLMs are streamlined models with fewer parameters compared to LLMs, which can number in the billions. SLMs utilize smaller, domain-specific datasets and require shorter training periods, typically just a few weeks, unlike LLMs, which demand vast datasets for broad learning and multiple months of training. Developing SLMs to excel within specific domains for robotic systems and ensuring real-time performance with minimal computational resources are essential research directions for advancing robot intelligence with SLMs.

The third implication is that LLMs, based on text-centered natural language processing, are limited as single-modality models when applied to real-world robotic systems where information often blends in diverse ways. Research on LLMs is transitioning from single-modality to multimodality models, as evidenced by VLMs and OpenAI Sora [228], with increasing demand for such models. Currently, to address the limitations of LLMs' single-modality, robotic systems are being developed with multimodality models that integrate vision, such as VLMs. However, relying solely on text and images falls short of the diverse information range required in the real world, including images, sounds, videos, and proprioceptive sensory information (such as the position, orientation, balance, movement degree, and direction of various parts of the robot). Proprioceptive sensory information related to actions and movements is particularly vital for enhancing dynamic human interaction, information processing, and manipulation and planning skills based on dynamic movements. For instance, the integrated VLA model, which facilitates low-level control based on LLMs and VLMs as shown by Google's RT-2 model, highlights the necessity for models capable of integrating information from a broader range of modalities to enhance robot intelligence.

Finally, the fourth area to consider is how to address safety and ethical issues when LLM is applied to robotic intelligence systems. Studies were conducted to address the issue of discriminatory and unsafe behaviors that may be generated by robot applications powered by LLMs [229]. The outputs of LLMs have the potential to generate content that is biased based on personal characteristics (such as race, nationality, religion, gender, disability, and so forth). In addition, they can also be used to instruct robotic systems to engage in violent or illegal behaviors such as misstatements, sexual predation, etc. Notable examples include discriminatory behaviors such as inadequate recognition of children or individuals with specific skin tones in human detection systems, and the exclusion of individuals with disabilities from task assignments. It is imperative to consider the potential social biases of LLM when integrating with robotic systems. Although this kind of consideration was secondary in traditional robotic systems because of the limitation of their language capability, it is a necessary consideration for LLMs to be able to generate human-like language. To address this issue, previous studies have attempted to resolve it in various ways, such as AutoRT's constitutional rules [10], DrEureka's safety instructions [134], and NeMo's guardrails [230]. The guideline-based output control of LLMs can represent an accessible method to ensure safety.

As an extension of this point, safety issues can be identified when integrating LLMs and VLMs into robotic intelligence systems [231]. Typically, in robotic intelligence systems, LLM models generate high-level action plans in various forms, such as programming codes and behavior trees based on natural language or vector prompts. At this point, a prompt attack has the potential to disrupt the inference of the LLMs, thereby threatening the reliability and safety of the robotic system. Prompt injection is one of the prompt attacks, whereby the inference of LLMs is subtly altered through specific inputs. Jailbreak, another prompt attack, bypasses safety rules and causes LLMs to generate abnormal behaviors to be performed by the robotic system. Consequently, even minor disturbances in the input prompts have the potential to cause the entire robotic system to malfunction. To defend against this critical threat to the reliability and safety of robotic systems, various techniques have been proposed, such as input validation, which filters the model's input, and context locking, which restricts access based on the history and content of the prompt. Furthermore, strict guardrails that restrict harmful or unsafe outputs from models can be an alternative to improve the reliability of robotic systems. However, it is essential to recognize that the security techniques may potentially lead to a decline in the performance of the robot system. Consequently, the trade-off between performance and safety must be carefully considered.

Since the emergence of ChatGPT and Microsoft's implementation of robot systems using ChatGPT [2], artificial intelligence components have been applied more widely and intensively in robotics research. Despite existing challenges, it is expected that research involving foundation models to improve robot intelligence will persist across various domains and methods, which will likely enhance the usability and market potential of robot systems influenced by these advancements.

6. Conclusions

In this paper, we have explored the potential impact and applicability of LLMs on robotics research fields by summarizing studies that applied LLMs and VLMs to robots. Fundamentally, LLMs can enhance the capability of robots in natural language processing to interact with humans and to improve the robots' autonomy in various task scenarios. In particular, the ability of LLMs to understand and generate natural language plays a crucial role in enabling robots to comprehend and execute complex commands. This survey confirmed that the scope of utilizing LLMs in robotics was not limited to simple natural language processing but also extended to broader research areas. This study explored extensive LLM applications in the robotics literature, such as planning, manipulation, and scene understanding, as well as reinforcement learning automation frameworks such as Eureka, and included robot actions in language models such as AutoRT. Moreover, the research direction of current generative AI models is transitioning towards multimodal

language models, moving beyond information acquisition and cognition aspects such as text, images, and videos to include actuator actions within large models in the robotics field.

While the surveyed studies indicated that LLMs play a promising role in the future of robotics, certain limitations were also identified. First, the increased computational resources and energy consumption associated with embedding LLMs into robotic systems must be addressed. Second, biases in language models and ethical considerations are significant issues that need to be tackled in robotics. Therefore, continual efforts will be necessary in future research to resolve these challenges.

Overall, LLMs are valuable tools that can significantly advance robotics. This review has revealed that innovative robot applications are possible through the integration of LLMs and VLMs. Moreover, these foundation models are expected to serve as critical elements for future robot research and practical applications in the real world.

Author Contributions: Conceptualization, S.S. and C.K.; methodology, S.S.; formal analysis, H.J., H.L. and S.S.; investigation, H.J., H.L. and S.S.; resources, H.J., H.L., S.S. and C.K.; writing—original draft preparation, H.J., H.L., S.S. and C.K.; writing—review and editing, H.J., H.L., S.S. and C.K.; visualization, H.J. and H.L.; supervision, S.S. and C.K.; project administration, S.S.; funding acquisition, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Technology Innovation Program (RS-2024-00423702, A Meta-Humanoid with Hypermodal Cognitivity and Role Dexterity: Adroid4X) funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea) and Regional Innovation Strategy (RIS) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2023RIS-007).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hello GPT-4o. Available online: https://openai.com/index/hello-gpt-4o/ (accessed on 13 August 2024).
2. Vemprala, S.H.; Bonatti, R.; Bucker, A.; Kapoor, A. ChatGPT for Robotics: Design Principles and Model Abilities. *IEEE Access* **2024**, *12*, 55682–55696. [CrossRef]
3. Hu, Y.; Xie, Q.; Jain, V.; Francis, J.; Patrikar, J.; Keetha, N.; Kim, S.; Xie, Y.; Zhang, T.; Zhao, S.; et al. Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis. *arXiv* **2023**, arXiv:2312.08782.
4. Xiao, X.; Liu, J.; Wang, Z.; Zhou, Y.; Qi, Y.; Cheng, Q.; He, B.; Jiang, S. Robot Learning in the Era of Foundation Models: A Survey. *arXiv* **2023**, arXiv:2311.14379.
5. Mao, Y.; Ge, Y.; Fan, Y.; Xu, W.; Mi, Y.; Hu, Z.; Gao, Y. A Survey on LoRA of Large Language Models. *arXiv* **2024**, arXiv:2407.11046.
6. Hunt, W.; Ramchurn, S.D.; Soorati, M.D. A Survey of Language-Based Communication in Robotics. *arXiv* **2024**, arXiv:2406.04086.
7. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. *Proc. Mach. Learn. Res.* **2021**, *139*, 8748–8763.
8. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; et al. RT-1: Robotics Transformer for Real-World Control at Scale. In Proceedings of the Robotics: Science and Systems 2023, Daegu, Republic of Korea, 10–14 July 2023. [CrossRef]
9. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C.; et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv* **2023**, arXiv:2307.15818.
10. Ahn, M.; Dwibedi, D.; Finn, C.; Arenas, M.G.; Gopalakrishnan, K.; Hausman, K.; Ichter, B.; Irpan, A.; Joshi, N.; Julian, R.; et al. AutoRT: Embodied Foundation Models for Large Scale Orchestration of Robotic Agents. *arXiv* **2024**, arXiv:2401.12963.
11. Ma, Y.J.; Liang, W.; Wang, G.; Huang, D.-A.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; Anandkumar, A. Eureka: Human-Level Reward Design via Coding Large Language Models. *arXiv* **2023**, arXiv:2310.12931.
12. Ma, Y.; Song, Z.; Zhuang, Y.; Hao, J.; King, I. A Survey on Vision-Language-Action Models for Embodied AI. *arXiv* **2024**, arXiv:2405.14093.
13. Zhou, H.; Yao, X.; Meng, Y.; Sun, S.; Bing, Z.; Huang, K.; Knoll, A. Language-Conditioned Learning for Robotic Manipulation: A Survey. *arXiv* **2023**, arXiv:2312.10807.