Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU35493

Large Language Models for Task Planning in Embodied AI: A Survey

Zhen Zhang

Faculty of Engineering, University of New South Wales, Sydney, Australia
bmozeefolks@gmail.com

Abstract. Large language models (LLMs) have recently emerged as promising components for task planning in embodied artificial intelligence (AI), where agents must decompose high-level natural language instructions into executable action sequences under dynamic environments and physical constraints. Unlike purely text-based planning, embodied task planning requires grounding in object affordances, partial observability, and the gap between symbolic reasoning and low-level control execution. Classical planning methods, such as STRIPS, PDDL, and HTN, provide formal and interpretable frameworks, yet they struggle with unstructured real-world settings and open-ended instructions. This paper surveys LLM-based approaches to embodied task planning. We present a structured taxonomy that organizes existing work into three complementary paradigms: (1) hierarchical planning, where LLMs serve as high-level planners that decompose goals into subgoals; (2) closed-loop planning, where execution feedback and environmental state monitoring support replanning; and (3) end-to-end embodied planning frameworks, where multimodal LLMs and vision-language-action models integrate perception, language understanding, and action prediction within learned policies. vcThese categories are not strictly mutually exclusive, but rather represent dominant architectural tendencies in current LLM-based embodied task planning research.. We compare these paradigms along dimension of accuracy, robustness, scalability, efficiency, and sim-to-real transfer. The comparison suggests that while LLMs are effective for commonsense-driven task decomposition and feedback-based replanning, they remain limited in physical reasoning, real-time efficiency, and reliable low-level execution. Open challenges are further discussed, including granularity mismatch, physical commonsense deficits, safe replanning, and benchmark standardization, and outline future directions toward more reliable and physically grounded embodied planning.

Keywords: Embodied artificial intelligence, Large language models, Task planning

1. Introduction

Embodied task planning requires an agent to transform high-level natural language instructions into executable action sequences in dynamic physical environments [1]. Unlike text-only planning, embodied task planning must address partial observability, object affordances, physical constraints, and long-horizon reasoning. Classical planning methods, such as STRIPS, PDDL, and hierarchical task networks, provide formal and interpretable frameworks, but they often rely on manually

DOI: 10.54254/2755-2721/2026.GU35493

specified domain models and struggle with open-ended instructions and unstructured real-world scenarios [1]. Recent advances in large language models (LLMs) offer new opportunities for embodied task planning through natural language understanding, commonsense reasoning, and task decomposition [2]. However, LLM-based methods still face important limitations, including weak physical grounding, unreliable action planning, inefficient replanning, and the mismatch or lack of grounding between symbolic reasoning and low level control.

This paper surveys LLM-based embodied task planning, organizing existing work into three paradigms: hierarchical planning, closed-loop planning with replanning, and end-to-end planning as policy learning [3]. These paradigms are not strictly mutually exclusive, but instead represent dominant architectural tendencies in current embodied task planning research. It then compares these approaches and discusses key challenges and future directions.

2. Foundations of embodied task planning

2.1. Definition and key challenges

Embodied task planning refers to the process of transforming high-level natural language instructions into executable action sequences in physical or simulated environments. Unlike text-only planning, it requires agents to reason about partial observability, object affordances, physical constraints, and long-horizon execution. For example, the instruction "put the cup on the shelf" requires the agent to identify the relevant objects, reason about spatial relations, select feasible manipulation actions, and execute them safely.

In embodied robotics and autonomous agents, task planning is usually described as the process of understanding user requirements, decomposing high-level goals into subtasks, and generating executable action sequences— a core component of autonomous robotic systems [1]. However, real-world environments are dynamic and uncertain: object positions may change, visual observations may be incomplete, and clutter or occlusion may increase perception difficulty [1].

2.2. Limitations of classical planning methods

Classical planning methods, such as STRIPS, PDDL, and hierarchical task networks, provide formal and interpretable frameworks for generating action plans. STRIPS was originally proposed as a problem solver that searches through "world models" to find a state where a goal is achieved [4]. PDDL later became a standard planning language for describing planning domains and problems, including possible states, actions, initial states, and goals. HTN planning represents tasks hierarchically and decomposes high-level tasks into lower-level actions, but it also depends heavily on manually designed task hierarchies and domain-specific knowledge. These properties make classical planners well-suited to structured and well-specified domains.

However, embodied AI often involves vague user instructions, diverse and previously unseen objects, noisy sensor observations, and changing environments. As a result, manually defining complete domain models, action preconditions, and effects becomes difficult, and classical methods struggle with natural language understanding, commonsense inference, and online adaptation.

2.3. Core capabilities of large language models

Large language models provide new opportunities for embodied task planning because they can interpret natural language, infer implicit goals, perform commonsense reasoning, and decompose

DOI: 10.54254/2755-2721/2026.GU35493

complex tasks into ordered subgoals. In robotics, SayCan shows that language models can provide high-level semantic knowledge for long-horizon instructions, while affordance models or skill-value models help evaluate whether candidate actions are feasible for a specific robot [5]. ProgPrompt further demonstrates that program-like prompts can guide LLMs to generate situated robot task plans using available actions, objects, and executable examples [6]. These works suggest that LLMs are useful as high-level planners or task decomposers. However, LLMs are not standalone embodied planners. The raw outputs of LLMs may include actions that are impossible for the robot or inconsistent with the current environment, so reliable embodied planning requires integration with perception, feedback, external tools, affordance models, and low-level control [6].

Based on these opportunities and limitations, recent studies have placed LLMs within embodied planning systems in different ways. Some systems use LLMs for high-level task decomposition, while others rely on them to revise plans after execution feedback or updated environmental observations. A further line of work integrates language models with perception and action learning, making the model closer to a policy rather than a separate planner. Section 3 therefore classifies existing methods according to the main role played by the LLM in the planning architecture.

3. Taxonomy of LLM-based planning methods

LLM-based task planning in embodied AI can be understood in terms of the role that the language model plays in the planning system. In some methods, the LLM mainly works as a high-level planner: it interprets a user instruction, decomposes the goal into smaller subtasks, and leaves low-level execution to external modules. In other methods, the LLM is placed inside a feedback loop, where it revises the plan according to execution results or updated environmental observations. More recent approaches go further by treating planning as part of a learned policy, especially in vision-language-action models that connect perception, language, and action more directly. Therefore, this section organizes existing methods into three main paradigms: hierarchical planning, closed-loop planning with replanning, and end-to-end planning as a policy. These categories are not completely isolated, since many practical systems combine decomposition, tool use, and feedback. However, this taxonomy clarifies the main architectural choices behind current LLM-based embodied planning methods.

3.1. Hierarchical planning with LLM as high-level planner

Hierarchical planning is one of the most common ways to use LLMs in embodied task planning. In this paradigm, the LLM does not directly control the robot at every low-level step. Instead, it works as a high-level planner that interprets the user's instruction, decomposes the overall goal into smaller subtasks, and decides a reasonable order for execution. The actual execution is then handled by lower-level modules, such as perception systems, motion planners, affordance models, or robot control APIs. The division makes sense because natural language instructions are usually abstract, while robot actions must be physically precise. For example, the instruction "clean the table" may be decomposed into "find cleaning cloth," "wipe the table," and "put the cloth away," but each of these subtasks still requires perception, grasping, navigation, and motion control. Existing works such as SayCan and ProgPrompt follow this general idea: the LLM provides high-level semantic reasoning, while additional mechanisms constrain the plan according to available skills, objects, or executable action formats [5, 6]. Therefore, hierarchical planningbridges the flexible reasoning ability of LLMs and the physical execution requirements of embodied agents.

DOI: 10.54254/2755-2721/2026.GU35493

3.1.1. Prompt-based task decomposition

Prompt-based task decomposition is usually the starting point of hierarchical LLM planning. In this type of method, the LLM receives a task description together with prompts, examples, or step-by-step guidance, and then produces a rough sequence of subtasks. For instance, "clean the table" may be broken down into finding a cloth, wiping the surface, and putting the cloth away. The advantage of this approach is that it is simple and easy to adapt to different tasks, since the planner can be changed mainly through the prompt rather than through model retraining. However, this also means that the plan is still mainly language-based. The model may overlook missing objects, unavailable robot skills, or physical constraints in the current scene. Therefore, prompt-based decomposition is useful for generating an initial plan, but it often needs additional grounding through tools, perception modules, or feedback before the plan can be executed reliably.

3.1.2. Tool-augmented and symbolic planning

Tool-augmented and symbolic planning connects LLMs with external tools, formal planners, code interpreters, or robot skill libraries. The main idea is that the LLM should not handle the whole planning pipeline alone. Instead, it can interpret the user's instruction and generate a structured intermediate plan, while external modules verify feasibility, solve symbolic planning problems, or execute low-level robot actions. This matters because LLMs are strong at semantic reasoning but cannot reliably guarantee that their plans are physically executable.

One common approach combines LLMs with symbolic planners such as PDDL-based systems. For example, LLM+P uses an LLM to translate natural language tasks into PDDL representations, and then relies on a classical planner to generate the final action sequence [7]. Another approach is to let the LLM generate executable code or call predefined robot skills. Code as Policies shows that language models can produce robot programs that invoke perception and control functions [8]. SayCan further combines LLM reasoning with affordance scores, selecting actions that are both semantically appropriate and physically feasible for the robot [5]. Overall, tool-augmented planning makes LLM-based planning more practical, but its performance still depends heavily on the quality of external tools, predefined skills, and perception modules.

3.2. Closed-loop planning with replanning

Closed-loop planning addresses a key weakness of one-shot planning: real-world execution rarely follows the original plan perfectly. In embodied environments, objects may be moved, perception may be incomplete, and robot actions may fail because of reachability, grasping errors, or unexpected obstacles. Instead of assuming a plan will run from start to finish, closed-loop methods let the agent observe execution and revise the plan when needed. In this setting, the LLM is not only a planner at the beginning of the task, but also a replanning module that can update its reasoning based on feedback. There are two common types of feedback.

First, execution feedback tells the model whether an action succeeded or failed. For example, if a robot fails to pick up a cup, the system can send this failure information back to the LLM, and the LLM may suggest moving closer, trying a different grasp, or selecting another object. ReAct follows a similar idea by interleaving reasoning and action, allowing the language model to update plans through interaction rather than only producing a static answer [9]. Inner Monologue also shows that language feedback, such as success detection, scene descriptions, and human responses, can help LLMs reason more effectively in robotic control scenarios [10].

DOI: 10.54254/2755-2721/2026.GU35493

The second type is environmental state feedback, where the planner receives updated information from perception systems, such as cameras, visual-language models, object detectors, or scene graphs. This is especially important because LLMs do not directly perceive the physical world. ReplanVLM, for instance, uses visual-language models to support task replanning and error correction when execution fails, improving robustness in both simulation and real-robot settings [11]. Overall, closed-loop planning is more adaptive than one-shot planning, but it also introduces extra costs. Replanning increases inference time, token usage, and system complexity. In physical environments, repeated trial-and-error can be unsafe or expensive. Future systems thus need more efficient and safer ways to use feedback.

3.3. End-to-end planning as a policy

End-to-end planning treats embodied planning less as a separate reasoning module and more as part of a learned policy. Instead of first generating a high-level plan and then passing it to external tools or controllers, the model directly maps observations and instructions to actions, or action-like outputs. In this setting, planning, perception, and action prediction become more tightly connected. This makes the approach attractive for embodied AI: real robot tasks often require balancing what the agent sees, what the user asks, and what the robot can physically do. However, end-to-end planning is also harder to interpret and usually requires large amounts of robot interaction data.

3.3.1. Direct action generation

In direct action generation, the model outputs actions more directly instead of just producing abstract subgoals. These actions may be discrete commands, such as "move forward," "pick up object," or "open drawer," or they may be represented as action tokens that can be decoded into robot behavior. Compared with hierarchical planning, this approach reduces the number of hand-designed intermediate steps. The model does not need to explicitly call a symbolic planner or manually written task tree; instead, it learns the connection between language instructions, observations, and actions from data.

The main advantage of direct action generation is simplicity at the system level. When the model works well, it avoids some errors from translating between high-level language plans and low-level controllers. However, this approach also has obvious risks. A language model may generate an action that sounds reasonable but is physically impossible for the robot. It may also struggle when fine-grained timing, force control, or collision avoidance is required. Therefore, even direct action generation often still depends on low-level controllers, safety filters, or embodied training data.

3.3.2. Vision-language-action models

Vision-language-action models(VLAs) are currently one of the most important directions in end-to-end embodied planning. These modelsaim to connect visual observations, language instructions, and robot actions in a single unified model. PaLM-E is an early example of an embodied multimodal language model that incorporates visual and state information into a language model and is trained for tasks such as sequential robotic manipulation planning and visual question answering [12]. RT-2 goes further by expressing robot actions as text tokens and co-training vision-language models on both web-scale vision-language data and robot trajectory data, allowing the model to transfer web knowledge into robotic control [13]. More recently, OpenVLA provides an open-source VLA model

DOI: 10.54254/2755-2721/2026.GU35493

trained on a large collection of real-world robot demonstrations, showing the growing interest in generalist robot policies [14].

Compared with modular planning, VLA models reduce the gap between perception and action because they learn directly from multimodal and embodied data. They are promising for building more general embodied agents. At the same time, they still face major limitations. They require large-scale training data, are less transparent than hierarchical systems, and may not generalize well across different robot bodies, environments, or task types. For this reason, end-to-end planning shouldbe seen not as a complete replacement for hierarchical or closed-loop methods, but as a complementary direction with strong long-term potential.

4. Comparative analysis

The three paradigms differ not just in architecture, but also in the kinds of embodied tasks they are best suited for. So a fair comparison needs to consider both planning performance and deployment constraints. Table 1 summarizes the main trade-offs across accuracy and robustness, scalability and efficiency, sim-to-real transfer, and their relationship with classical planning.

Table 1. Comparison of LLM-based embodied planning paradigms

Dimension	Hierarchical Planning	Closed-loop Planning	End-to-End Planning
Accuracy & Robustness	Strong task decomposition, but sensitive to perception and environment errors	More robust through feedback and replanning, but replanning may add instability	Direct perception-action mapping, but failures are less interpretable
Scalability & Efficiency	Efficient for long-horizon tasks, but relies on external modules and skills	Adaptive but computationally costly due to repeated feedback cycles	Lower inference overhead, but requires massive embodied training data
Sim-to-Real Gap	Uses explicit skills and constraints to reduce infeasible actions	Can adapt online to real-world failures, though slowly and sometimes unsafely	Learns directly from embodied data, but transfer generalization remains limited
Relation to Classical Planning	Enhances classical planning with language-based decomposition.	Adds feedback-driven replanning under uncertainty	Reduces symbolic planning, but still needs safety constraints

Overall, hierarchical planning remains the most interpretable and modular paradigm, especially when high-level plans can be grounded by symbolic planners, affordance models, or robot skill libraries. Closed-loop planning offers stronger robustness in dynamic environments, but this comes at the cost of additional inference and system complexity.

End-to-end planning as a policy is promising because it reduces the separation between perception, language, and action. However, its data requirements and limited interpretability mean that it is better viewed as a complementary direction rather than a complete replacement for hierarchical or feedback-based LLM-based embodied task planning.

5. Challenges and future directions

LLM-based embodied planning still has several key challenges. First, there is a granularity gap between high-level language plans and low-level robot control. An instruction like "pick up the cup" still needs precise grasping, motion planning, and collision avoidance. Second, LLMs often lack reliable physical commonsense and thus may generate plans that sound reasonable but are not physically executable. Third, replanning in the real world is costly because repeated failures can

DOI: 10.54254/2755-2721/2026.GU35493

waste time or even damage objects. Finally, current benchmarks are not standardized, which makes it difficult to compare different methods fairly. Future work should combine LLM reasoning with perception, affordance models, world models, symbolic verification, and safer low-level controllers. Rather than replacing classical planners or robot controllers, LLMs are more likely to serve as high-level reasoning modules in grounded embodied systems.

6. Conclusion

This survey has examined how large language models can support task planning in embodied AI, where agents must turn high-level natural language instructions into executable action sequences in dynamic and physically constrained environments. Compared with classical planning methods such as STRIPS, PDDL, and HTN, LLMs offer stronger flexibility in language understanding, commonsense reasoning, and task decomposition. These abilities enable them to interpret vague user goals and generate structured plans. At the same time, LLMs still lack reliable physical grounding and cannot guarantee the real-world executability of a generated plan.

The reviewed methods can be grouped into three main paradigms. Hierarchical planning uses LLMs as high-level planners that decompose tasks and rely on tools, symbolic planners, or robot skill libraries for execution. Closed-loop planning improves robustness by using execution feedback and environmental observations to revise plans during task execution. End-to-end planning, especially through vision-language-action models, attempts to connect perception, language, and action prediction within a unified policy. Each paradigm has clear strengths along with important limitations in accuracy, efficiency, scalability, and sim-to-real transfer.

Overall, LLMs should not be viewed as complete replacements for classical planners or low-level robot controllers. A more realistic direction uses them as high-level reasoning modules within grounded, feedback-aware, and physically constrained embodied systems. Future research should focus on bridging the gap between symbolic plans and low-level control, improving physical commonsense, enabling safer replanning, and developing standardized benchmarks for fair comparison.

References

[1] Bian, S., Zhang, Y., Tian, G., et al. (2026). Large language model-based task planning for service robots: A review. *Biomimetic Intelligence and Robotics, 6*(1), 100274. https://doi.org/10.1016/j.birob.2026.100274

[2] Wang, H., Karnik, S., Lim, B., & Bansal, S. (2026). Using language models as closed-loop high-level planners for robotics applications: A brief overview and benchmarks. arXiv:2511.07410v2.

[3] Liang, W., Zhou, R., Ma, Y., et al. (2025). Large model empowered embodied AI: A survey on decision-making and embodied learning. arXiv:2508.10399.

[4] Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence, 2*(3–4), 189–208. https://doi.org/10.1016/0004-3702(71)90010-5

[5] Ahn, M., Brohan, A., Brown, N., et al. (2022). Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691.

[6] Singh, I., Blukis, V., Mousavian, A., et al. (2023). ProgPrompt: Program generation for situated robot task planning using large language models. *Autonomous Robots, 47*, 999–1012. https://doi.org/10.1007/s10514-023-10135-3

[7] Liu, B., Jiang, Y., Zhang, X., et al. (2023). LLM+P: Empowering large language models with optimal planning proficiency. arXiv:2304.11477.

[8] Liang, J., Huang, W., Xia, F., et al. (2023). Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. https://doi.org/10.1109/ICRA48891.2023.10160591

[9] Yao, S., Zhao, J., Yu, D., et al. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

DOI: 10.54254/2755-2721/2026.GU35493

[10] Huang, W., Xia, F., Xiao, T., et al. (2023). Inner monologue: Embodied reasoning through planning with language models. In K. E. Bekris, K. Hauser, S. L. Herbert, & J. Yu (Eds.), *Proceedings of the 7th Conference on Robot Learning*, 1769–1782.
[11] Mei, A., Zhu, G.-N., Zhang, H., & Gan, Z. (2024). ReplanVLM: Replanning robotic tasks with visual language models. arXiv:2407.21762.
[12] Driess, D., Xia, F., Sajjadi, M. S. M., et al. (2023). PaLM-E: An embodied multimodal language model. arXiv:2303.03378.
[13] Brohan, A., Brown, N., Carbajal, J., et al. (2023). RT-2: Vision-language-action models transfer web knowledge to robotic control. arXiv:2307.15818.
[14] Kim, M. J., Pertsch, K., Karamcheti, S., et al. (2025). OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, 2679–2713.