Enhancing LLM Planning for Robotics Manipulation through Hierarchical Procedural Knowledge Graphs

Jiacong Zhou[1*] **Jiaxu Miao**[2*] **Xianyun Wang**[2,3] **Jun Yu**[2,3†]
[1]School of Computer Science, Hangzhou Dianzi University
[2]The School of Intelligence Science and Engineering, Harbin Institute of Technology, Shenzhen
[3]Pengcheng Laboratory
jczhou@hdu.edu.cn yujun@hit.edu.cn

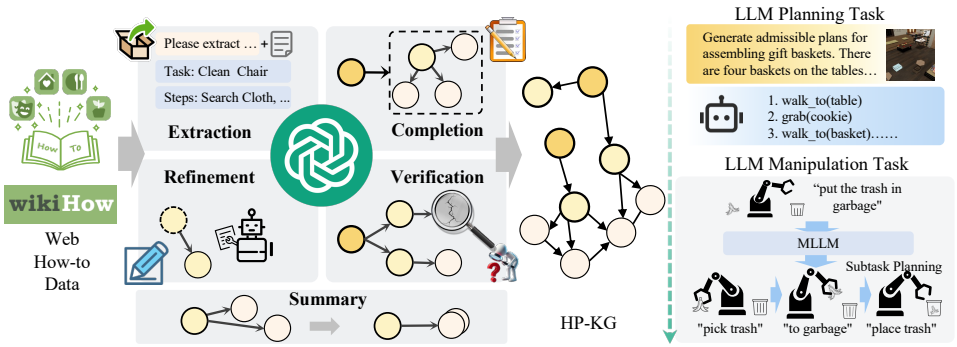

Figure 1: Overview of our work. We propose a large-scale Procedural Knowledge Graph constructed from web how-to data, which can significantly enhance the planning capabilities of LLMs in robot manipulation tasks.

Abstract

Large Language Models (LLMs) have shown the promising planning capabilities for robotic manipulation, which advances the development of embodied intelligence significantly. However, existing LLM-driven robotic manipulation approaches excel at simple pick-and-place tasks but are insufficient for complex manipulation tasks due to inaccurate procedural knowledge. Besides, for embodied intelligence, equipping a large scale LLM is energy-consuming and inefficient, which affects its real-world application. To address the above problems, we propose Hierarchical Procedural Knowledge Graphs (**HP-KG**) to enhance LLMs for complex robotic planning while significantly reducing the demand for LLM scale in robotic manipulation. Considering that the complex real-world tasks require multiple steps, and each step is composed of robotic-understandable atomic actions, we design a hierarchical knowledge graph structure to model the relationships between tasks, steps, and actions. This design bridges the gap between human instructions and robotic manipulation actions. To construct HP-KG, we develop an automatic knowledge graph construction framework powered by LLM-based multi-agents, which eliminates costly manual efforts while maintaining high-quality graph structures. The resulting HP-KG encompasses over 40k activity steps across more than 6k household tasks, spanning diverse everyday scenarios. Extensive experiments demonstrate that small scale LLMs (7B) enhanced by our HP-KG significantly improve the planning capabilities, which are stronger than 72B LLMs only. En-

*Equal contribution.
†Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

couragingly, our approach remains effective on the most powerful GPT-4o model. Our code and data will be publicly available[3].

1 Introduction

Embodied AI [1, 2] refers to AI that is integrated into physical systems, such as robots, enabling them to interact with the physical world with the abilities of perception, reasoning, planning, and execution. Recent achievements of Large Language Models (LLMs) [3–8] have shown the remarkable capabilities for robotic task planning [9, 10], greatly advancing the development of embodied AI. These models usually serve as high-level planners to decompose human instructions into executable sub-goals [11, 9, 12, 10, 13], while relying on pre-defined skills [14–16] for execution.

Recent studies [17, 18] show that such LLM-driven planning methods usually generate unrealistic or logically inconsistent planning steps due to lacking procedural commonsense, especially in complex manipulation tasks. For instance, LLMs may ignore the agent's current physical state, thus failing to include necessary prerequisite actions (e.g., a standup action before any movement) [19], or overlook physical constraints such as the need to open a closed container before fetching items from inside [20]. Moreover, robots are often limited by a finite energy supply, while LLM-driven planners typically require large-scale models (e.g., PaLM-E [21] with 562B parameters) to possess sufficient planning capabilities in complex, long-horizon scenarios. For embodied intelligence, equipping a large scale LLM is energy-consuming and inefficient, which affects its real-world application.

To address the above problems, we propose to construct an effective Procedural Knowledge Graph to enhance LLM-driven planners. Procedural knowledge refers to the understanding of how to perform specific tasks, typically expressed as sequences of steps required to achieve a specific goal [22, 19]. Since the Procedural Knowledge Graph provides the commonsense needed for planning, injecting the correct Procedural Knowledge into LLMs can effectively enhance their reasoning accuracy. Therefore, a small-scale LLM equipped with the Procedural Knowledge Graph has sufficient capability to perform planning, alleviating the computational cost demands of embodied AI.

However, developing an effective representation of procedural knowledge to enhance robotic planning capabilities remains a challenge. Existing methods usually adopt coarse-grained task decomposition and simply represent procedural knowledge as reference documents [23, 24, 19], without specific structure design. The main challenge in robot planning lies in translating high-level goals into feasible steps, due to the domain gap between language comprehension and robotic execution. A natural observation is that procedures of robot manipulation can be divided into a finite set of atomic actions that the robot can understand and execute. These atomic actions can be combined into a series of steps, which can further be combined into tasks that are highly abstract as human instruction in the real world. Motivated by this, we design a novel and effective Hierarchical Procedural Knowledge Graph (**HP-KG**) to organize procedures into three distinct layers: tasks, steps, and actions, as shown in Figure 2. Each procedure is further enriched with textual attributes (description, name, and tips).

Furthermore, to eliminate the need for manual knowledge engineering and reduce human effort, we introduce a framework to automatically construct the procedural knowledge graph through multi-agents calibration [25, 5, 26]. Specifically, we focus on household activities in this work as they are frequently performed by everyone and represent a promising area for robotic assistance to facilitate daily living [27]. We systematically filter household-related tasks from the WikiHow corpus [28] and combine them with the BEHAVIOR dataset [27] as knowledge source. We then prompt an LLM to extract steps from each household task and complete them by generating corresponding actions and their textual attributes. Subsequently, two LLM agents are employed to iteratively verify and refine the generated procedures based on designed rules. To enrich information while reducing redundancy, we perform semantic similarity clustering and LLM-based knowledge merging.

Finally, we propose a retrieval approach to leverage the constructed HP-KG. Given a language instruction, a refined query is generated to retrieve relevant knowledge nodes based on semantic similarity. Through K-hop breadth-first search and re-ranking, it identifies the most pertinent nodes and converts their sub-graphs into textual descriptions for contextual planning. Extensive experiments on ActPlan-1K [18] and RLBench [29] demonstrate that our HP-KG enables smaller models (7B) to achieve stronger capabilities than larger models (72B) only. Encouragingly, our approach remains effective on the most powerful GPT-4o model. Overall, the contributions of our work are as follows:

[3] https://anonymous.4open.science/r/HP-KG-68EE/

• We design a novel hierarchical procedural knowledge graph structure that effectively formalizes complex household tasks through a structure of tasks, steps, and actions.

• We introduce a novel automated framework that leverages LLM-based multi-agents to construct hierarchical procedural knowledge graphs, eliminating the need for manual knowledge engineering.

• We present HP-KG, a large-scale procedural knowledge graph, containing 42,000+ activity steps across 6,000+ household tasks in daily scenarios.

• Our method significantly improves the planning capability (+17.64% on 7B LLMs) and reduces the scale demand for LLMs.

2 Related Works

Knowledge Graph Construction. Traditional Knowledge Graph Construction methods typically involve multiple tasks, including entity extraction [30, 31] and relation classification [32, 33], which incur substantial human effort and cost [34]. With the advancement of pre-trained language models like BERT [35], GPT-3 [36], end-to-end triplet extraction emerges as a promising paradigm [34, 37]. Contemporary graph construction approaches [38–40] leverage large language models (LLMs) [3, 5, 8] for entity extraction via prompting or fine-tuning.

LLMs with Knowledge Graph. Knowledge graphs (KGs), which provide structured factual representations, have emerged as a powerful tool to enhance LLM performance [41–43]. Recent advances in Knowledge Graph Augmented Generation [44–46] demonstrate that KGs can serve as external knowledge bases to provide accurate factual information, effectively enhancing the factual correctness in LLM-generated responses [47–50]. Furthermore, several studies have shown that leveraging the structural relationships inherent in KGs can enhance the reasoning capabilities of LLMs [51–53].

Foundation Models for Robotics Manipulation. Recent achievements of vision-language foundation models [54–56] have significantly influenced the field of robotics manipulation. These models demonstrate the potential for controlling robots to perform complex tasks. Recent studies can be broadly categorized into two paradigms: One stream adopts vision-language-action models [57, 58] such as RT-2 [59], RT-X [60], and OpenVLA [61] to directly map visual inputs and language instructions to robotic actions. The other paradigm leverages vision-language models (VLMs) to decompose high-level instructions into sub-goals [11, 9, 62–64], which are then solved through pre-defined skills [14, 65, 15, 66]. For example, Yang et al. [67] leverages Vision Language Models to generate subgoals, then employs dynamics-based planning to solve the subgoal. Our method is orthogonal to these approaches. Benefiting from the in-context understanding capabilities of vision language models, our HP-KG can integrate seamlessly with various methods' subgoal generation processes.

3 Approach

In this section, we first present our structure design of the Hierarchical Procedural Knowledge Graph (HP-KG) for household activities. Second, we propose an automated procedural knowledge graph construction framework for building our HP-KG. Third, we introduce a retrieval method that retrieves relevant procedural knowledge as contextual input to enhance LLM-based planning.

3.1 Hierarchical Structure Design for Procedural Knowledge Graph

One main challenge of robotics planning tasks is how to bridge high-level human instructions and atomic actions that are understandable by robots. Representing and organizing procedural knowledge is central to its ability to assist robotics planning effectively. A well-structured procedural knowledge graph should be easily applicable to diverse downstream applications and guide reliable execution by both human operators and robotic systems.

To this end, our procedure structure design should satisfy the following principles. First, the nodes of this knowledge graph should include comprehensive tasks and atomic actions, as well as how to effectively connect them. Second, the knowledge graph should include safety-critical guidelines to prevent potential accidents, which is essential for ensuring correct procedure execution [68]. We observe that complex procedures in household activities are composed of a set of atomic procedures (e.g., open container). Some atomic actions combine to form a simple step. A series of steps, in turn,

combine to create complex tasks that approach high-level human instructions (e.g., clean the room), naturally forming a hierarchical structure of procedural knowledge.

Based on these considerations, as Figure 2 shows, we design a simple yet effective Hierarchical Structure that encompasses multiple characteristics. To represent the hierarchical nature of procedural knowledge, we categorize procedures into three levels: tasks, steps, and actions, which correspond to complex composite procedures, simple composite procedures, and atomic procedures, respectively. A task node links to step nodes via *HasStep* edges, and step nodes link to action nodes through *HasAction* edges. The temporal sequence is captured by *NextStep* edges between steps and *NextAction* edges between actions. Additionally, each procedure contains two key attributes: *HasDescription* specifies what needs to be performed, *HasTips* provides tips and warnings.

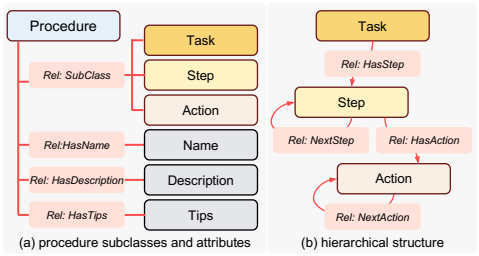

Figure 2: Overview of procedural graph structure. **(a)** Procedure subclass and their attribute relationships. **(b)** The hierarchical structure between different procedures.

3.2 Automatic Graph Construction Framework

To construct our HP-KG, we propose an LLM-based multi-agent framework for automatic Procedural Knowledge Graph construction. As Figure 3 shows, the framework comprises four key stages: (1) data source cleaning and processing, (2) procedures generation and completion, (3) rules-guided iterative verification and refinement, and (4) hierarchical clustering and summarization.

Data Source Cleaning and Processing. In our work, we utilize two main data sources. The first is the WikiHow corpus [28], one of the largest online databases containing comprehensive how-to articles across multiple domains. Each article in WikiHow typically consists of a title, a main goal, and several methods to achieve this goal, where each method contains multiple steps with detailed descriptions. The second is the BEHAVIOR-1K dataset [27], which comprises approximately 1,000 most common household activities. Each activity is defined through a BDDL (Behavior Domain Definition Language) file that specifies the activity goals and available objects in the environment.

The WikiHow corpus contains numerous articles across various domains, while only household activities are needed in this paper. Moreover, documents in the corpus may contain semantically similar activities, which could lead to redundancy in the extracted procedures. Therefore, we first filter documents by categories, retaining only household-related categories. Subsequently, we employ Deepseek-V3 [4] to determine whether each document describes a household activity based on its title and descriptions (detailed prompt template is provided in Appendix E). Finally, we cluster documents based on semantic similarity. We obtain approximately 5.8K unique household activities from the WikiHow corpus. Furthermore, we standardize input formats for BDDL-defined activities in the BEHAVIOR-1K dataset by using LLMs to generate WikiHow-style content.

Procedures Generation and Completion. Based on collected documents, we prompt the LLM agent as the **Procedure Generator** to extract and complete procedures. During the extraction phase, the generator extracts each activity's tasks, steps, and corresponding textual attributes (e.g., task content, task tips) from each document. According to our hierarchical graph structure, we build the interconnection between tasks and steps through the *hasStep* relation. After extraction, we design an action completion prompt to generate atomic action procedures with minimal redundancy. Using this prompt, the generator produces corresponding atomic actions and their attributes for each step procedure. These actions along with their attributes constitute action procedures, which are then linked to their respective step procedures through the *hasAction* relation. These hierarchical procedures interconnect through defined relations, forming a preliminary procedures graph.

Rules-Guided Iterative Verification and Refinement. Due to the inherent limitations in Large Language Models (LLMs), the generated preliminary procedures inadequately adhere to procedural principles (e.g., they contain redundant information and tend to be overly specific). To address these challenges, we propose an Iterative Verification and Refinement method to improve the quality of procedures. A **Rules-Guided Verifier** invokes multiple rule-checking models to assess the given procedure's compliance with established rules, subsequently aggregating these results to generate a

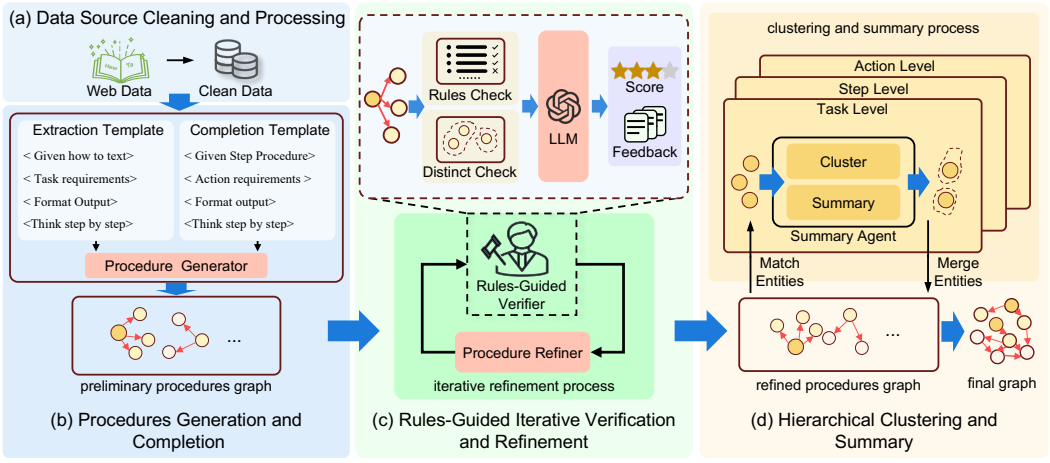

Figure 3: Overview of our automatic construction pipeline: (a) Data source cleaning and processing. (b) A **Generator** extracts steps with attributes from the data source and performs actions completion. (c) A **Verifier** and A **Refiner** iteratively validate procedures against designed rules while progressively refining them. (d) A **Summary Agent** hierarchically clusters and merges redundant procedures.

validation score and feedback. Additionally, we prompt an LLM as a **Procedure Refiner** to iteratively modify the provided procedures until the score surpasses the predefined threshold.

Specifically, based on fundamental principles of our Hierarchical Structure Design, we construct a comprehensive set of validation rules (e.g., check if content accurately describes a procedure, check if action is atomic, etc.), denoted as $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, where n is the total number of rules. Given a generated procedure p_t and its corresponding sub-procedures $\mathbf{p_c} = (p_c^1, ..., p_c^k)$, we concatenate them into $\mathbf{p} = [p_t; p_c^1; ...; p_c^k]$ to serve as input to the Verifier, where sub-procedures represent nodes that are linked by p_t via *hasStep* or *hasAction* relations. The agent then invokes a rule-checking language model M to evaluate whether generated procedures $\mathbf{p}$ comply with rules in $\mathcal{R}$. For each rule $r_i \in \mathcal{R}$, the model outputs a binary decision $M(\mathbf{p}, r_i) \in \{0, 1\}$, where 1 indicates compliance and 0 indicates violation of the respective rule. Therefore, we obtain rule compliance scores, denoted as:

$$\mathbf{s}(\mathbf{p}, \mathcal{R}; M) = \Big(M(\mathbf{p}, r_i), M(\mathbf{p}, r_2), ..., M(\mathbf{p}, r_n)\Big). \tag{1}$$

To identify redundant elements across sub-procedures $\mathbf{p_c}$, the agent also employs an embedding model[4] to encode both the name and description of each procedure. It then groups sub-procedures whose pairwise embedding cosine similarity exceeds a threshold, resulting in a list of clusters $\mathbf{C} = (C_1, C_2, ...C_t)$, where t is the number of groups. Furthermore, the Verifier converts rule validation results $\mathbf{s}$ and aggregation results C into textual descriptions separately. It then uses these results and procedures $\mathbf{p}$ to prompt a powerful LLM (e.g., GPT-4o) to generate a score and feedback.

After the verification, we use a Procedure Refiner to modify given procedures $\mathbf{p}$. The Refiner utilizes feedbacks, textual result descriptions, and given procedures $\mathbf{p}$ as inputs to generate modified procedures $\mathbf{p}'$. These refined procedures are then iteratively verified until their scores surpass a predefined threshold. We provide the complete prompts and detailed framework rules in Appendix E.

Hierarchical Cluster and Summary. Although our iterative verification and refinement process enhances the quality of hierarchical procedures (e.g., tasks and their sub-steps, steps and their sub-actions), redundancy persists among procedures at the same level. To address this problem, we cluster and merge procedures of the same level via a **Summary Agent**.

We first match all procedures at a given level (e.g., task procedures) as the input of the Summary Agent. This agent uses the same embedding model to encode each procedure's name and content and computes the cos similarity between each pair of procedures. Procedures with similar semantics will be aggregated into a group U_i, together forming the group list $\mathbf{U} = (U_1, U_2, \ldots)$.

For each group U_i that contains multiple procedures, the Summary Agent employs an LLM to generate comprehensive summaries based on designed prompt "Given multiple semantically similar procedures,

[4]https://huggingface.co/NovaSearch/stella_en_400M_v5

analyze and synthesize them into a single and actionable summary...". After the summarization process, we merge these newly generated procedures into the existing graph. We iteratively repeat the summarization process from task level to action level, ultimately deriving the final procedural graph.

3.3 Graph Retrieval-Augmented Planning

In this section, we propose a graph retrieval-augmented reasoning approach that utilizes HP-KG to enhance LLM planning and robotic manipulation. We aim to retrieve the top-K relevant nodes at specified target levels and transform their sub-graphs into textual descriptions to facilitate contextual planning. Our retrieval-augmented method consists of (1) Node Indexing, (2) Query Retrieval, (3) Entity Expansion and Re-Ranking, (4) Procedural Graph Augmented Planning.

Node Indexing. For each node n with textual attributes x_n, we apply a pre-trained embedding encoder (e.g., Zhang et al. [69]) to obtain its representation $z_n \in \mathbb{R}^d$:

$$z_n = \text{Encoder}(x_n). \tag{2}$$

Query Retrieval. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ denote our HP-KG, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges respectively. Additionally, $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ represents the node feature matrix, where $|\mathcal{V}|$ denotes the number of nodes and d is the feature dimension.

We begin with a general instruction and a retrieval target level L_{target}. First, we prompt LLMs to extract the task objectives as query x_q and apply same embdding model to encode x_q as follows:

$$z_q = \text{Encoder}(x_q) \in \mathbb{R}^d. \tag{3}$$

We then retrieve the top-k_1 most semantically similar procedure nodes V_{k_1}.

$$V_{k_1} = \text{TopK}_{n \in \mathcal{V}} \, \text{sim}(z_q, z_n), \tag{4}$$

where z_n denotes the embeddings of node n and $\text{sim}(z_q, z_n)$ denotes the cosine similarity between z_q and z_n. The TopK returns the top-k nodes with highest similarity.

Entity Expansion and Re-Ranking. Since neighboring nodes often encapsulate complementary or contextually relevant semantic information for the initially retrieved nodes, we aim to expand our retrieved nodes to discover additional relevant procedures. Specifically, we expand the retrieved nodes V_{k_1} by performing a K-hop breadth-first traversal from each retrieved node as follows:

$$V_{k_1}^{'} = \bigcup_{n \in V_{k_1}} \{n' \mid \text{dist}(n', n) \leq K, n' \in \mathcal{V}\}, \tag{5}$$

where $\text{dist}(\cdot)$ denotes the path distance between nodes in HP-KG. Furthermore, we filter out nodes that do not match the target level L_{target} from $V_{k_1}^{'}$, obtaining a refined set:

$$V_{\text{target}} = \{n \in V_{k_1}^{'} \mid \text{Level}(n) = L_{\text{target}}\}, \tag{6}$$

Where $\text{Level}(\cdot)$ denotes the level of node. Then, we re-rank the candidate nodes V_{target} based on query x_q, and select the top-k_2 nodes V_{k_2} as follows:

$$V_{k_2} = \text{TopK}_{n \in V_{\text{target}}} \, \text{sim}(z_q, z_n). \tag{7}$$

Procedural Graph Augmented Planning. After obtaining the retrieved nodes V_{k_2}, we aim to convert these nodes into textual context for LLM planning. To this end, we require not only the retrieved nodes but also the associated sub-procedures for each procedure node. These sub-procedures detail the sequential steps of each task and the specific actions required within each step.

Therefore, we extract associated lower-level nodes based on predefined relations (e.g., HasStep, HasAction, etc.), constructing hierarchical subgraphs where each retrieved node serves as the root. Furthermore, we convert each hierarchical subgraph into a textual description (detailed templates are in Appendix E). To enable the procedural graph augmented planning, we feed the converted textual descriptions into the LLM as additional context. The detailed algorithm is presented in Appendix B.

4 Experiments

In this section, we conduct comprehensive experiments on LLM planning and robotic manipulation benchmarks to evaluate the effectiveness and efficiency of our HP-KG. Extensive ablation studies are also conducted to show the effectiveness of our hierarchical structure and the construction framework.

Table 1: Comparison of task success rates (%) for zero-shot robotic manipulation on RLBench. Our proposed HP-KG improves baseline success rates when retrieving the Top-K (K=3,5) most relevant procedures (numbers in parentheses indicating absolute gains).

Method	HP-KG	Top-K	Open_wine	Take_scale	Take_umbrella	Slide_block	Play_jenga	Average
Voxposer [9]	-	-	10%	15%	60%	50%	0%	27%
	✓	Top-3	40%	25%	60%	45%	0%	34%(**+7%**)
	✓	Top-5	40%	20%	65%	60%	0%	37%(**+10%**)
MA [11]	-	-	20%	0%	35%	10%	20%	17%
	✓	Top-3	20%	0%	50%	30%	40%	28%(**+11%**)
	✓	Top-5	10%	0%	40%	30%	35%	23%(**+6%**)

Table 2: We compare the effectiveness of our HP-KG against Chain of Thought (COT) prompting for action planning. In both cases, we employ RVT as the action executor.

Method	Top-K	Open_wine	Take_lid	Take_scale	Take_umbrella	Slide_block	Average
RVT [57]	-	65%	5%	0%	30%	15%	23%
RVT w/ COT	-	70%	10%	15%	35%	5%	27%(+4%)
RVT w/ HP-KG	Top-3	75%	10%	10%	55%	20%	34%(**+11%**)
RVT w/ HP-KG	Top-5	85%	5%	15%	50%	10%	33%(**+10%**)

4.1 Experimental Settings

Benchmarks. We mainly evaluate our HP-KG on **RLBench** [29] and **Blocks Arrange** [70] tasks for robotics manipulation. Furthermore, we also evaluate our approach on **ActPlan-1K** [18] for LLM planning. Detailed environment setup is provided in the Appendix A.

- **RLBench.** This benchmark comprises 100 distinct hand-crafted manipulation tasks with varying complexity, ranging from basic target reaching and door opening to complex multi-stage operations. Each task is specified by a natural language instruction that defines the manipulation goal. In our experiments, we select a set of 6 complex tasks, each requiring multiple actions.
- **Blocks Arrange.** This task requires robots to arrange randomly located colored boxes according to given textual instructions. In our experiments, we selected 5 complex tasks with varying difficulty levels, ranging from relatively simple one like `put_blocks_in_corners` to more challenging ones such as `stack_blocks_into_three_towers`.
- **ActPlan-1K.** This dataset contains multi-modal household activity planning instances. The task in ActPlan-1K requires a MLLM to generate procedural plans P given a task description T and a sequence of environment images $\{I_1, I_2, ...\}$ as input. Each instance is annotated with two reference procedural plans P_1^*, P_2^*, where each plan consists of multi-step action sequences.

Baselines. In RLBench, we compare different zero-shot manipulation approaches that leverage large language models (e.g., VoxPoser [9], MA [11]). We enhance their sub-task planning process by incorporating procedural knowledge and compare the results against their baseline performance. In the ActPlan-1K benchmark, we evaluate various multi-modal large language models with different scales (e.g., GPT-4o [54], Gemini [55], Qwen2-VL [72]) both with and without HP-KG augmentation. We also leverage LLMs to generate subgoals and employ a pretrained policy [57] for execution on RLBench. In this context, we compare our HP-KG against Chain of Thought (COT)[73] method. Additionally, we compare with SayCan[71] on Blocks Arrange task.

Metrics. We employ Success Rate (SR) for RLBench and Blocks Arrange, while using the Longest Common Subsequence (LCS) metric for the ActPlan-1K benchmark. Detailed metrics are as follows:

- **Success Rate.** Similar to VoxPoser, we conduct 20 trials per task and calculate the success rate.
- **LCS.** On the ActPlan-1K benchmark, we use the longest common subsequence (LCS) metric to evaluate the generated plans against annotated reference plans (using the first one as default). Following [18], we encode each step with a language embedding model[5] and consider those with similarity above 0.8 as matching steps in the LCS calculation.
- **Mix LCS.** Given the diversity in planning approaches and individual preferences, we propose a Mixed Longest Common Subsequence (MIX LCS) metric. In our LCS similarity comparison, we consider a generated plan step to be similar if it matches any step in either reference plan.

Implementation details. In our Iterative Verification and Refinement process, the maximum iterations is set to 3. In our Retrieval process, we set k_1 to 100 and set k_2 to 3 or 5 depending on the experimental

[5] sentence-transformers/all-MiniLM-L6-v2

Table 3: Comparison of Different Planning method on the Blocks Arrange task.

Method	Top-K	Average SR
SayCan [71]	-	26%
GPT-4o w/ COT	-	33%
GPT-4o w/ HP-KG	Top-3	**45%**
	Top-5	42%

Table 4: Comparison of multimodal large models of different scales on Actplan-1K.

Model	HP-KG	Top-K	LCS ↑	Mix LCS ↑	Average ↑
GPT-4o-2024-11-20 [54]	-	-	11.7848	12.2700	12.0274
	✓	Top-3	11.8987	12.5063	12.2025(**+1.45%**)
	✓	Top-5	12.2616	12.7552	12.5084(**+3.99%**)
Gemini-2.0-Flash [55]	-	-	10.4261	11.0717	10.7486
	✓	Top-3	11.0210	11.5316	11.2763(**+4.90%**)
	✓	Top-5	10.9746	11.4767	11.2256(**+4.43%**)
Qwen2-VL-7B-Instruct [56]	-	-	7.2909	7.8045	7.5477
	✓	Top-3	8.2067	9.5527	8.8797(**+17.64%**)
	✓	Top-5	8.4092	9.0084	8.7088(**+15.38%**)
Qwen2-VL-72B-Instruct [56]	-	-	8.5189	9.0464	8.7826
	✓	Top-3	9.5527	10.0759	9.8143(**+11.74%**)
	✓	Top-5	9.2784	9.7637	9.5210(**+8.41%**)

Table 5: Comparison of Different Graph Structures on the Actplan-1K. All methods use Top-K=3.

Graph Structure	LCS	Mix LCS	Average
Qwen2-VL-72B-Instruct [56]			
-	8.5189	9.0464	8.7826
Unstructured Docs	8.9118	9.4345	9.1771
Coarse Graph Structure	8.9113	9.2953	9.1033
Hierarchical Graph (ours)	9.5527	10.0759	**9.8143**

Table 6: Effectiveness analysis of each stage in our automated graph construction framework on Actplan-1K.

Procedures Generator	Iterative Verification	Summary Agent	Average LCS
×	×	×	8.7826
✓	×	×	9.4809
✓	✓	×	9.5968
✓	✓	✓	**9.8143**

conditions. For zero-shot manipulation approaches, we utilize GPT-4o as the primary planner unless explicitly stated in experiments. Appendix A provides comprehensive implementation details.

4.2 Main Results

Results on RLBench. Table 1 presents a comprehensive performance comparison of whether the method is enhanced with our proposed HP-KG. Compared to Voxposer and MA, our HP-KG significantly improves their success rates by up to 10% and 11%, respectively. For all baselines, we achieve improvements on more than half of the tasks, while the remaining tasks are not completed due to the limitations of the baselines' manipulation capabilities. For instance, we find that MA struggles with tasks requiring precise operations due to its lack of precision in identifying target positions, which prevents it from accurately grasping the lid handle during lid-removal tasks. Nevertheless, our HP-KG still shows promising improvements in handling these challenging scenarios. We also compare our HP-KG with COT method, and results are presented in Table 2.

Results on Blocks Arrange. Table 3 compares the task success rates of different methods, with GPT-4o serving as the base LLM for all approaches. The results demonstrate that LLMs enhanced with our HP-KG achieve higher success rates compared to both SayCan and COT methods. This performance gain stems from our approach's ability to inject explicit procedural knowledge into LLMs, whereas SayCan and COT methods merely leverage the implicit knowledge embedded in LLMs.

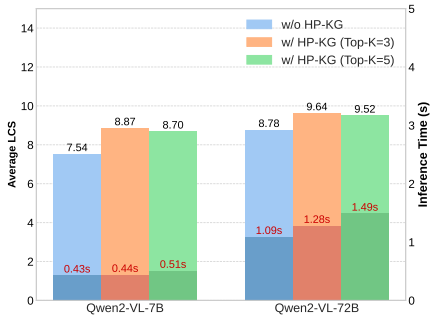

Figure 4: Performance and inference time comparison across different model sizes.

Results on ActPlan-1K. Table 4 presents a comprehensive comparison of planning abilities across multi-modal large models with/without HP-KG enhancement. Our HP-KG improves the planning abilities of models across all scales. Specifically, our approach significantly boosts the average LCS of the small-scale model (Qwen2-VL-7B) by 17.64% compared to the 7B baseline, even surpassing the performance of the 72B model baseline. Furthermore, for large-scale models with high capability, such as GPT-4o, Gemini-2.0-Flash, and Qwen2-VL-72B, our approach achieves performance improvements of up to 3.99%, 4.90%, 11.74%, respectively. Comparing results of different retrieval number (top-3 v.s. top-5), we find that the performance of smaller models decreases as the Top-K increases, due to their limited long context comprehension ability. In contrast, for larger models like GPT-4o, the additional procedural knowledge further enhances their performance.

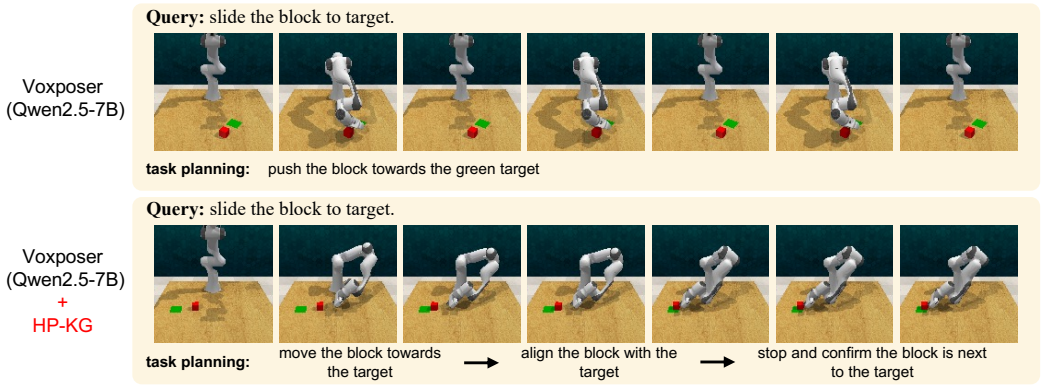

Figure 5: Qualitative Results: Voxposer with small LLM (Qwen2.5-7B) correctly decomposes tasks and executes them successfully when enhanced with our HP-KG framework.

4.3 Ablation Studies

Ablation Study on Hierarchical Structure Design. To assess the effectiveness of our hierarchical structure, we transform retrieved nodes into various alternative structures and evaluate each variant on the Actplan-1K. Specifically, we introduce two variants: (1) Unstructured Docs, where we directly convert the retrieved nodes into text without sub-graphs; and (2) Coarse Graph Structure, where we extract sub-steps for each retrieved task node and only transform these tasks and their steps into textual descriptions without actions. Table 5 demonstrates that our Hierarchical Graph Structure shows promising improvements over two variants. We find that providing procedural knowledge enhances LLMs' planning capabilities by supplementing limited procedural commonsense. Our hierarchical structure is more effective at organizing procedural knowledge than alternative variants.

Ablation Study on Graph Construction. We perform ablation experiments across the three stages in our Automatic Graph Construction Framework. We evaluate Qwen2-VL-72B-Instruct enhanced with HP-KGs from each stage on Actplan-1K, and report Average LCS metric. Table 6 shows that our preliminary procedures graph provides modest improvement over the baseline. Both our Iterative Verification and Refinement process and Summary Agent further enhance the quality of the procedural graph, leading to additional improvements in overall performance.

Efficiency Analysis. We also compare the inference time across MLLMs of various scales to evaluate the efficiency of our HP-KG. We deploy Qwen2-VL-7B-Instruct and Qwen2-VL-72B-Instruct on the same hardware platform and evaluate them on the Actplan-1K benchmark, measuring both average inference time and Average LCS metric. Results are presented in Figure 4. As the retrieval number increases (from top-3 to top-5), inference time of LLMs increases slightly due to longer context length. Notably, Qwen2-VL-7B enhanced with our HP-KG achieved performance comparable to the Qwen2-VL-72B baseline while saving 50% of inference time, demonstrating that our method can effectively reduce reliance on large-scale LLMs. We also compare different LLMs within Voxposer for robotics manipulation tasks and evaluate their computational efficiency in Appendix C.

4.4 Case Studies

Figure 5 presents a qualitative analysis by comparing scenarios with and without HP-KG enhancement. We utilize Qwen2.5-7B as the planner in Voxposer to decompose tasks into different subtasks and execute them sequentially. The Qwen2.5-7B model alone struggles with proper task decomposition, hindering the robotic arm's ability to complete assigned tasks. In contrast, with HP-KG enhancement, the model generates more logical subtasks, enabling successful execution of the overall task objective. We also conduct experiments in a long-horizon robot task planning benchmark using the Kitchen-World environment [67]. Due to space constraints, detailed results are presented in Appendix D.

5 Limitations and Conclusion

In this work, we propose Hierarchical Procedural Knowledge Graphs (HP-KG) to advance long-horizon planning capabilities of LLMs. We introduce a hierarchical graph structure that captures

complex real-world task relationships. We also develop an automatic knowledge graph construction framework powered by LLM-based multi-agents. Furthermore, we propose a retrieval approach to leverage the constructed HP-KG enhancing LLMs for robotic manipulation. Extensive experiments on multiple benchmarks demonstrate our improvements compared to existing approaches. However, our knowledge graph is limited to household activities, constraining its applicability in general scenarios. We plan to develop a general procedural graph and apply it to wider field in the future.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62125201, U24B20174, No. 62306273. The authors declare no competing interests.

References

[1] Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, et al. Embodiedscan: A holistic multi-modal 3d perception suite towards embodied AI. In *CVPR*, 2024.

[2] Alessandro Suglia, Claudio Greco, Katie Baker, Jose L. Part, et al. Alanavlm: A multimodal embodied AI foundation model for egocentric video understanding. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *EMNLP*, 2024.

[3] OpenAI. Gpt-4v(ision) system card. *https://api.semanticscholar.org/CorpusID:263218031*, 2023.

[4] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[5] DeepSeek-AI, Daya Guo, Dejian Yang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[6] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[7] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Wu, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

[8] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[9] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *CoRL*, 2023.

[10] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[11] Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. In *CoRL*, 2024.

[12] Siyuan Huang, Haonan Chang, Yuhan Liu, Yimeng Zhu, Hao Dong, Peng Gao, Abdeslam Boularias, and Hongsheng Li. A3vlm: Actionable articulation-aware vision language model. *arXiv preprint arXiv:2406.07549*, 2024.

[13] Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *CVPR*, 2024.

[14] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[15] Yecheng Jason Ma, William Liang, Guanzhi Wang, et al. Eureka: Human-level reward design via coding large language models. In *ICLR*, 2024.

[16] Ajay Mandlekar, Yuke Zhu, Animesh Garg, et al. ROBOTURK: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018.

[17] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. In *Neurips*, 2024.

[18] Ying Su, Zhan Ling, Haochen Shi, Cheng Jiayang, Yauwai Yim, and Yangqiu Song. Actplan-1k: Benchmarking the procedural planning ability of visual language models in household activities. In *EMNLP*, 2024.

[19] Valentina Anita Carriero, Antonia Azzini, Ilaria Baroni, Mario Scrocca, and Irene Celino. Human evaluation of procedural knowledge graph extraction from text with large language models. In *EKAW*, 2024.

[20] Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

[21] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[22] Anisa Rula, Gloria Re Calegari, Antonia Azzini, Davide Bucci, Ilaria Baroni, and Irene Celino. Eliciting and curating procedural knowledge in industry: Challenges and opportunities. In *Qurator*, 2022.

[23] Kaichen Zhao, Yaoxian Song, Haiquan Zhao, Haoyu Liu, Tiefeng Li, and Zhixu Li. Towards coarse-grained visual language navigation task planning enhanced by event knowledge graph. In *CIKM*, 2024.

[24] Michael Beetz, Philipp Cimiano, Michaela Kümpel, Enrico Motta, Ilaria Tiddi, and Jan-Philipp Töberg. Transforming web knowledge into actionable knowledge graphs for robot manipulation tasks. In *ESWC*, 2024.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[26] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

[27] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, et al. BEHAVIOR-1K: A benchmark for embodied AI with 1, 000 everyday activities and realistic simulation. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski, editors, *CORL*, 2022.

[28] Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about goals, steps, and temporal ordering with wikihow. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *EMNLP*, 2020.

[29] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics Autom. Lett.*, 2020.

[30] Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. Joint learning of named entity recognition and entity linking. In *ACL*, 2019.

[31] Aniruddha Mahapatra, Sharmila Reddy Nangi, Aparna Garimella, and Anandhavelu N. Entity extraction in low resource domains with selective pre-training of large language models. In *EMNLP*, December 2022.

[32] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, 2015.

[33] Zexuan Zhong and Danqi Chen. A frustratingly easy approach for entity and relation extraction. In *ACL*, 2021.

[34] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. *ACM Comput. Surv.*, 2023.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[36] Tom B. Brown, Benjamin Mann, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[37] Lishan Qiao, Limei Zhang, Songcan Chen, and Dinggang Shen. Data-driven graph construction and graph learning: A review. *Neurocomputing*, 2018.

[38] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. Codekgc: Code language model for generative knowledge graph construction. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 2024.

[39] Yansong Ning and Hao Liu. Urbankgent: A unified large language model agent framework for urban knowledge graph construction. In *NeurIPS*, 2024.

[40] Bowen Zhang and Harold Soh. Extract, define, canonicalize: An llm-based framework for knowledge graph construction. In *EMNLP*, 2024.

[41] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*, 2025.

[42] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *NeurIPS*, 2024.

[43] Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *EMNLP*, 2023.

[44] Bohan Chen and Andrea L. Bertozzi. Autokg: Efficient automated knowledge graph generation for language models. In *2023 IEEE International Conference on Big Data (BigData)*, 2023.

[45] Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. Knowledge graph-guided retrieval augmented generation. In *NAACL*, 2025.

[46] Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, and Christos Faloutsos. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *arXiv preprint arXiv:2412.16311*, 2024.

[47] Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *WWW*, 2025.

[48] Lei Liang, Mengshu Sun, Zhengke Gui, et al. KAG: boosting llms in professional domains via knowledge augmented generation. *arXiv preprint arXiv:2409.13731*, 2024.

[49] Junjie Wang, Mingyang Chen, Binbin Hu, et al. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *EMNLP*, 2024.

[50] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *ICLR*, 2025.

[51] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. In *ICLR*, 2024.

[52] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. *arXiv preprint arXiv:2407.10805*, 2024.

[53] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*, 2024.

[54] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[55] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[56] Peng Wang, Shuai Bai, Sinan Tan, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[57] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: robotic view transformer for 3d object manipulation. In *CoRL*, 2023.

[58] Zhutian Yang, Caelan Reed Garrett, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Sequence-based plan feasibility prediction for efficient task and motion planning. In *Robotics: Science and Systems XIX*, 2023.

[59] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, et al. RT-2: vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

[60] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, et al. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration. In *ICRA*, 2024.

[61] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[62] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *CORL*, 2024.

[63] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of GPT-4V in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

[64] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: from natural language instructions to feasible plans. *Auton. Robots*, 2023.

[65] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2022.

[66] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024. URL `https://openreview.net/forum?id=ehfRiF0R3a`.

[67] Zhutian Yang, Caelan Reed Garrett, Dieter Fox, et al. Guiding long-horizon task and motion planning with vision language models. *arXiv preprint arXiv:2410.02193*, 2024.

[68] Arda Uzunoglu, Gözde Gül Sahin, and Abdulfattah Safa. PARADISE: evaluating implicit planning skills of language models with procedural warnings and tips dataset. In *ACL*, 2024.

[69] Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. Jasper and stella: distillation of sota embedding models. *arXiv preprint arXiv:2412.19048*, 2025.

[70] Yongchao Chen, Yilun Hao, Yang Zhang, and Chuchu Fan. Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation. *arXiv preprint arXiv:2503.01700*, 2025.

[71] Brian Ichter, Anthony Brohan, Yevgen Chebotar, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *CoRL*, 2022.

[72] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[73] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[74] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

[75] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[76] Qwen, :, An Yang, Baosong Yang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

[77] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *ICCV*, 2025.

[78] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: yes.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: we discuss the limitations in our paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [No]

   Justification: we do not provide a theoretical conclusion

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: we provide via detailed instructions for how to replicate the results in our paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: we promise to make code public and help the community reproduce our results when our work is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: we provide via detailed instructions for how to replicate the results in our paper and promise to make code public and help the community reproduce our results when our work is accepted.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: No need to calculate error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: **[TODO]**

   Justification: we provide detailed instructions about computer resources in our paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: our research conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: In our paper, we propose mainly an effective method to enhance the planning capacity of LLM, which is not subject to high risk of abuse.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

Justification: In our paper, we propose mainly an effective method to enhance the planning capacity of LLM, which is not subject to high risk of abuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: we have credited all asset creators and respected the associated licenses and terms of use.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: we promise to make code public and help the community reproduce our results when our work is accepted.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [No]

Justification: we did not use crowdsourcing or conducted research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: we did not use crowdsourcing or conducted research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: No.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Overall Experimental Details

Software and Hardware Configurations. We conduct our experiments on servers equipped with NVIDIA A6000 GPUs (48GB VRAM), with NVIDIA CUDA Toolkit version 11.8. For inference time comparison, we deploy different-sized models (7B and 72B) using the vLLM inference framework and use the AWQ quantized version for the 72B model to fit within the available GPU memory.

Experimental Parameter Settings. In our Iterative Verification and Refinement process, the maximum iterations is set to 3. For all clustering operations, we employ cosine similarity with a threshold of 0.85. In our Procedural Graph Retrieval process, we set k_1 to 100 and set k_2 to 3 or 5 depending on the experimental conditions. For zero-shot manipulation approaches, we utilize GPT-4o as the primary planner unless explicitly stated in experiments and we set the target retrieval level L_{target} to the step procedure. For experiments on Actplan-1K, we set the target retrieval level L_{target} to the Task procedure due to its more complex task objectives.

Details of the Constructed Graph. Our HP-KG encompasses a rich variety of actions and objects, enabling comprehensive coverage of diverse tasks in common daily scenarios. Additionally, we present the task distribution of our HP-KG, demonstrating its potential for assisting robots in complex task completion. The specific details are shown below:

Table 7: Statistics of nodes and their corresponding average number of child nodes.

Node Type	Node Count	Average Sub-Nodes
Task Nodes	6380	6.58
Step Nodes	40659	4.11
Action Nodes	136943	-

- **Graph Statistics.** Our HP-KG comprises 6K Task Nodes, 40K Step Nodes, and 136K Action Nodes in total, with comprehensive statistics detailed in Table 7. Furthermore, HP-KG incorporates over 1,300 distinct verbs and approximately 26,275 objects, whose distributions are illustrated through word cloud visualizations in Figure 6.
- **Task Length Distribution.** We present the distribution of total task lengths in HP-KG in Figure 7. Our tasks span from simple procedures (approximately 3-7 actions) to complex operations (over 30 actions). This diversity highlights HP-KG's capability to support robotic systems in executing both straightforward and complex task sequences.

Figure 6: Statistical analysis of verbs and objects in HP-KG actions. **(Left)** Word cloud visualization of verbs. **(Right)** Word cloud visualization of objects.

B Procedural Graph Retrieval-Augmented Planning Algorithm

The complete formulation of the retrieval algorithm is detailed in Algorithm 1.

C Additional Experiments Results

In this section, we present additional experimental results on both ActPlan-1K and RLBench benchmarks.

Additional comparison on ActPlan-1K. Table 8 presents a broader comparison of diverse large vision-language models (e.g., InternVL-2.5-26B [74], Llama-3.2-90B-Vision-Instruct [75]) on the

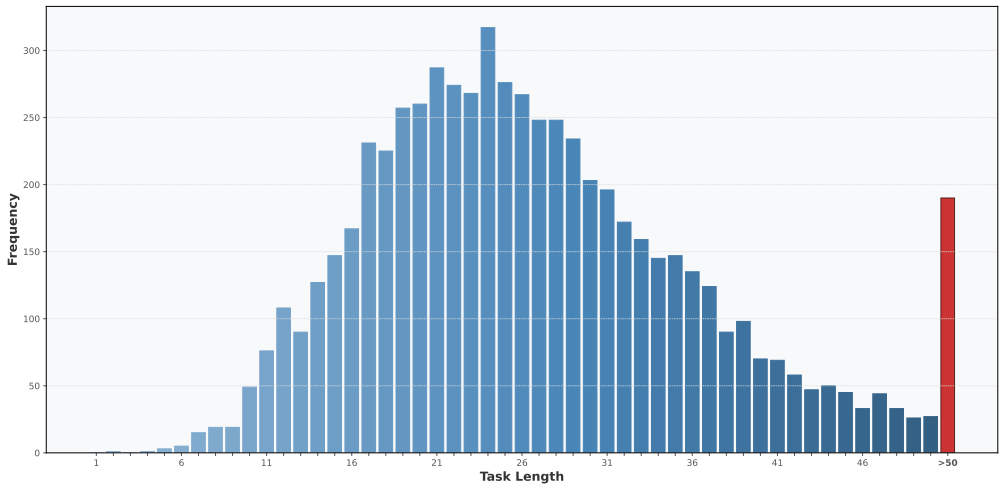

Figure 7: Distribution of total task lengths in HP-KG.

Algorithm 1 Procedural Knowledge Graph Retrieval and Planning.

Input: General instruction, hierarchical procedural knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, target level L_{target}, parameters k_1, k_2, hop limit K, encoder model

Output: Retrieved procedural knowledge in textual format

Query Retrieval:
Extract task objectives as query x_q using LLM
Encode query: $z_q = \text{Encoder}(x_q) \in \mathbb{R}^d$
Retrieve initial candidates: $V_{k_1} = \text{TopK}_{n \in \mathcal{V}} \text{ sim}(z_q, z_n)$ ▷ Top-k_1 semantically similar nodes
Entity Expansion and Re-Ranking:
Expand retrieved nodes: $V_{k_1}^{'} = \bigcup_{n \in V_{k_1}} \{n' \mid \text{dist}(n', n) \leq K, n' \in \mathcal{V}\}$ ▷ K-hop expansion
Filter by target level: $V_{\text{target}} = \{n \in V_{k_1}^{'} \mid \text{Level}(n) = L_{\text{target}}\}$
Re-rank candidates: $V_{k_2} = \text{TopK}_{n \in V_{\text{target}}} \text{ sim}(z_q, z_n)$ ▷ Final top-k_2 nodes
Procedural Text Formatting:
Text $\leftarrow \emptyset$ ▷ Initialize formatted text collection
for each node $n \in V_{k_2}$ **do**
    Retrieve attributes and child nodes of n
    Format node information into structured text segment t_n
    Text $\leftarrow$ Text $\cup \{t_n\}$
end for
Concatenate all text segments in Text into final textual context T
Return: T ▷ Formatted procedural text

ActPlan-1K [18] benchmark. Since some models cannot fully adhere to instructions and produce complete planning outputs, we record the count of successfully generated planning solutions for each model. Our HP-KG approach demonstrates consistent effectiveness across models of varying scales. We present detailed ablation results of our construction framework in Table 9.

Additional comparison on RLBench. In addition to GPT-4o, we integrate several open-source models (Qwen2.5-7B-Instruct, Qwen2.5-72B-Instruct [76]) into Voxposer and conduct efficiency analysis. The success rates across different tasks and inference time comparisons are presented in Table 10 and Figure 8, respectively. Compared to these models without HP-KG, our approach improves their success rates by up to 10.83% and 5.83%, respectively. Furthermore, with HP-KG enhancement, Qwen2.5-7B-Instruct achieves performance comparable to Qwen2.5-72B-Instruct while requiring less than 20% of its inference time.

Detailed Inference Time Analysis. We conduct experiments to evaluate the efficiency of our retrieval method. Specifically, we measure the average Instruction Encode Time (Encode Time),Retrieval Time, K-hop Time, Reranking Time, Graph Conversion Time and LLM Inference Time during

Table 8: We evaluated additional large multimodal language models on the ActPlan-1K benchmark. In our analysis, the Success metric indicates the number of instances where models successfully generated planning solutions in the required format.

Model	HP-KG	Top-K	Success	LCS ↑	Mix LCS ↑	Average ↑
InternVL-2.5-26B [74]	-	-	237/237	8.4430	8.6920	8.5675
	✓	Top-3	237/237	8.5147	8.9915	8.7531(**+2.16%**)
	✓	Top-5	237/237	8.4599	8.9957	8.7278(**+1.87%**)
Llama-3.2-90B-Vision-Instruct [75]	-	-	197/237	10.3756	10.9086	10.6421
	✓	Top-3	204/237	11.6715	12.4166	12.0440(**+13.17%**)
	✓	Top-5	181/237	10.8563	11.1602	11.0082(**+3.44%**)

Table 9: Detailed results of each stage in our automated graph construction framework on Actplan-1K.

Procedures Generator	Iterative Verification and Refinement	Summary Agent	LCS	Mix LCS	Average LCS
×	×	×	8.5189	9.0464	8.7826
✓	×	×	9.2320	9.7299	9.4809
✓	✓	×	9.2953	9.9578	9.6265
✓	✓	✓	9.5527	10.0759	9.8143

the Top-3 and Top-5 search processes on the Actplan-1K[18]. All settings are consistent with the Experimental Details in Appendix A. Table 11 presents the detailed timing breakdown. Our retrieval pipeline is highly efficient, with the combined overhead (encoding, retrieval, k-hop, reranking, and graph conversion) being significantly smaller than the MLLM inference time.

Additional experiments on complex manipulation benchmark. We conduct experiments on a more complex benchmark, VLABench[77], which contains multiple complex tasks involving common sense knowledge, physical rules, and reasoning capabilities. VLABench is designed to evaluate not only VLAs but also MLLMs. In the MLLM evaluation setting, models must output both the skills to be invoked and their corresponding parameters.

We select four task categories: CommonSense, Complex, M&T, and PhysicsLaw, with each category containing 5-8 tasks. The CommonSense, M&T, and PhysicsLaw categories evaluate the model's understanding of world procedural knowledge, while the Complex category requires not only world knowledge but also long-horizon planning capabilities.

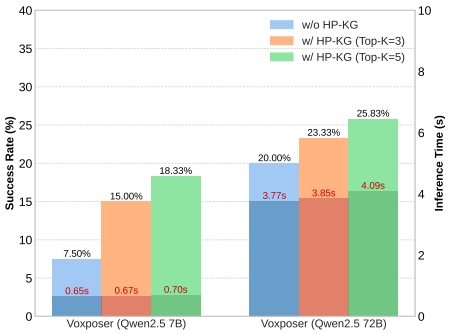

Figure 8: Performance and inference time comparison of different model sizes in the Voxposer framework.

We compare the accuracy of MLLMs enhanced with our HP-KG against those without such enhancement, with each task evaluated over 50 trials. As shown in Table 12, our HP-KG consistently improves planning performance even on complex manipulation benchmark. This indicates that our approach effectively enhances MLLMs' planning capabilities by providing structured prior knowledge that guides decision-making in challenging scenarios.

Table 10: Additional comparison of task success rates (%) for zero-shot robotic manipulation on RLBench using Voxposer [9]. We integrated various open-source large language models into the Voxposer framework.

Models	HP-KG	Top-K	Open_wine	Take_lid	Take_scale	Take_umbrella	Slide_block	Play_jenga	Average
Voxposer (Qwen2.5-7B-Instruct [76])	-	-	15.00%	0.00%	5.00%	10.00%	15.00%	0.00%	7.50%
	✓	Top-3	25.00%	0.00%	15.00%	5.00%	45.00%	0.00%	15.00%(**+7.50%**)
	✓	Top-5	30.00%	0.00%	25.00%	20.00%	35.00%	0.00%	18.33%(**+10.83%**)
Voxposer (Qwen2.5-72B-Instruct [76])	-	-	5.00%	0.00%	25.00%	55.00%	35.00%	0.00%	20.00%
	✓	Top-3	30.00%	0.00%	25.00%	65.00%	20.00%	0.00%	23.33%(**+3.33%**)
	✓	Top-5	35.00%	0.00%	35.00%	65.00%	20.00%	0.00%	25.83%(**+5.83%**)

Table 11: Detailed Inference Time Analysis with Qwen2-VL-72B-Instruct on ActPlan.

Top-k	Encode Time	Retrieval Time	K-hop Time	Reranking Time	Graph Conv. Time	LLM Inference Time
Top-3	0.038s	0.045s	0.001s	0.025s	0.001s	1.28s
Top-5	0.039s	0.046s	0.001s	0.027s	0.001s	1.49s

Table 12: Performance Comparison with Different Top-k Settings on VLABench

Models	HP-KG	Top-k	CommonSense	M&T	PhysicsLaw	Complex	Average
Intern3-VL-8B [78]	-	-	26.15	28.19	10.21	19.01	20.89
	✓	3	28.70	29.21	18.95	21.47	24.58
	✓	5	29.55	28.70	18.22	21.16	24.41

D Long-Horizon Planning Experiments

We further investigate the capability of our HP-KG to support embodied robots in long-horizon planning tasks by conducting evaluations in the more sophisticated simulated environment Kitchen World [67].

Kitchen World Settings. The environment comprises either a single-arm or dual-arm robot and generated kitchen scenarios containing multiple movable objects and articulated objects or surfaces. The robot is required to complete tasks based on given natural language instructions. Following Yang et al. [67], we consider a task where a single-arm robot is required to make chicken soup in this kitchen environment. It is worth noting that this represents a highly complex long-horizon task, requiring approximately a dozen sequential actions for successful completion. Specifically, we consider a complex scenario where all cabinet doors are initially closed and the pot is covered with its lid.

We adopt GPT-4o [54] as our base model for generating sub-goals and apply the default TAMP approach from Yang et al. [67] to solve the generated sub-goals. Similar to Yang et al. [67], we use Task Completion Percentage as our metric, defined as the proportion of solved subproblems among all subgoals.

Results. Table 13 presents a performance comparison between models enhanced with our proposed HP-KG and those without such enhancement. The results demonstrate that base models enhanced with HP-KG achieve higher Task Completion Percentage, indicating that our HP-KG can effectively facilitate robot execution of complex long-horizon tasks. Additionally, we present a qualitative comparison in Figure 9, which illustrates the subgoals generated by different methods during the experiments. GPT-4o enhanced with our HP-KG can generate more reasonable subgoals, while the baseline model without HP-KG may omit certain essential operations (e.g., opening cabinets and picking up salt and pepper).

Table 13: Comparison of Task Completion Percentage for Kitchen Task Execution.

Method	Top-K	Task Completion Percentage
GPT-4o [54]	-	77.4%
GPT-4o w/HP-KG	Top-3	86.9%(**+9.5%**)
GPT-4o w/HP-KG	Top-5	83.5%(**+6.1%**)

E Implementation Details of the Automated Graph Construction Framework

Details of Verification Rules In our Iterative Verification and Refinement process, we construct a comprehensive set of validation rules to evaluate whether the generated procedures adhere to procedural principles. We categorize these rules based on their application types (e.g., tasks, steps, actions) and present several examples in Table 14.

Instruction Template in Construction Framework. We provide the prompt templates used at each stage of our Framework. Specifically, Table 15 and Table 16 present the prompts for procedure generation and completion. Table 17 shows the prompt that guides the model to aggregate rule verification results and generate corresponding feedback and scores, while the prompt in Table 18 instructs the Refiner to modify procedures based on this feedback. Finally, the Summary Agent employs the prompt in Table 19 to merge redundant procedures. In each stage of our framework, we

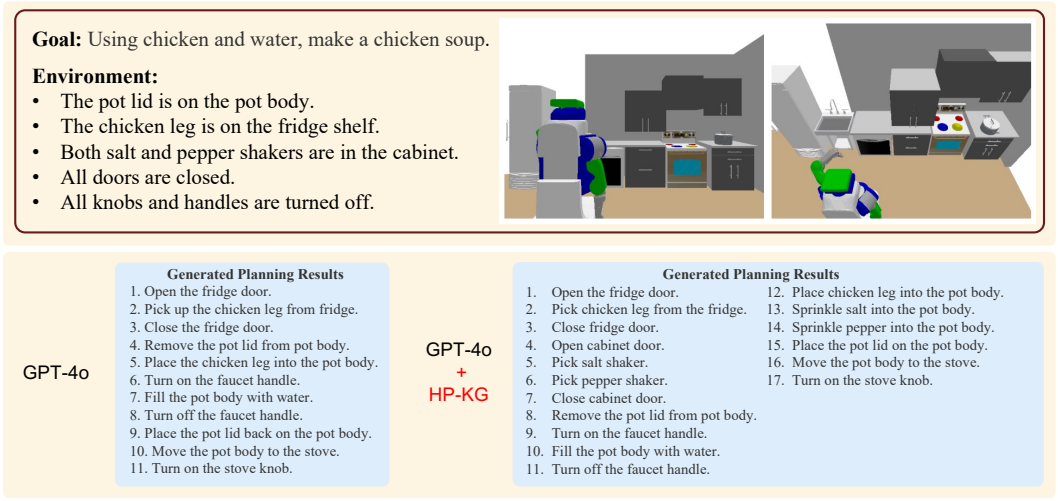

Figure 9: Qualitative Results in the Kitchen World Environment. GPT-4o enhanced with our HP-KG generates more reasonable subgoals, while the baseline model without HP-KG may omit certain essential operations.

Table 14: Validation rules based on established procedural principles

Procedure Type	Rules
Task Procedure	Check if task name contains specific and unambiguous descriptors. Check if task name avoids vague terms. Check if task name clearly indicates the purpose.
Step Procedure	Check if step name avoids ambiguous terms descriptors. Check if description explains purpose and necessity. Check if description avoids redundant information. Check if steps follows logical sequence.
Action Procedure	Check if action follows basic structure. Check if action is atomic and independently executable. Check if description explains specific goal. Check if actions follows logical sequence. Check if actions follows logical sequence. Check if actions avoids redundancy with other actions.

employ GPT-4o as our primary LLM to generate outputs based on the prompts. Notably, for verifying procedures' compliance with rules, we utilize Deepseek-V3, which offers comparable performance to GPT-4o but at a lower operational cost.

F Example of Procedural Graph

We present an example of our HP-KG, as illustrated in Figure 10. This case details the procedural steps and associated actions for completing the task 'clean broom and remove stains'. Each procedure includes three components: a name, detailed description, and practical tips.

Table 15: This designed prompt is used to extract tasks and their corresponding step-by-step instructions from WikiHow documents.

```
Analyze multiple similar Wikihow examples to extract canonical procedural knowledge.

Input:
{examples}

Thinking Process:
1. Compare all examples to identify:
- Common goal across all tasks
- Essential steps present in examples
- Conflicting instructions (resolve via majority vote)

2. For each step:
- Find the most precise action verb used
- Note critical tips from all variants

Output in the following JSON format:
{
    "task": "[specific purpose/goal] [main how-to]",
    "description": "Contextual summary containing: [Operational significance and scope boundaries, Key differentiators
from alternative approaches, Optimal application scenarios]",
    "tips": "Comprehensive list of potential issues, important tips, and warnings",
    "steps": [{
        "step": "[PrecisionVerb] [TargetObject] [specific purpose]",
        "description": "Detailed step description including: [ What actions to take, Why these actions are necessary,
Expected outcomes, Important considerations or precautions"],
        "tips": "Step-specific considerations, timing requirements, measurements, or safety precautions"
}]
}
Please ensure your analysis:
- Includes safety considerations and edge cases
- Provides clear, actionable steps
- Lists relevant tips and warnings for each step
- Provides generalized solutions that can be adapted to various scenarios
- Considers different environmental conditions and constraints that might affect the task execution

Naming and Description Requirements:

1. Task Names:
- Follow the structure: "[specific purpose/goal] [main purpose]"
- Use clear, specific, and unambiguous descriptions
- Clearly indicate the purpose

2. Task Descriptions:
- Explain the purpose and importance
- Include task-specific tips
- Highlight key warnings or special requirements

3. Step Names:
- Follow a logical sequence
- Avoid ambiguous terms
- Focus on what to do, not why or how
- Ensure logical flow between steps

4. Step Descriptions:
- Detail specific actions and their purposes
- Explain why each action is necessary
- Describe expected outcomes
- Provide step-specific tips and precautions
```

Table 16: This action completion prompt facilitates the generation of specific actions.

```
I will provide a complete task breakdown with the following components:
Task Objective: {task}
Task Description: {description}
Important Tips and Considerations: {tips}
Detailed Steps: {steps}
Please Generate Action For All Steps.

Requirements for Action Generation:
1. Action Requirements
   Action sentences structure: [Action] [Object] [Spatial: Object] [Orientation] [Direction][State]

2. Required Information for Each Action:
   - Description: Detailed explanation including specific goal of this atomic action, Why it's necessary at this step,
Required preconditions, Expected outcome/success criteria.
   - Tips: List of specific considerations: Safety precautions, Quality control points, Common mistakes to avoid,
Environmental considerations.

3. Anti-Redundancy Guidelines:
   A. Consolidate Actions When: Identical action verbs on similar object types, Same execution method/tool/pattern,
Order is flexible/interchangeable, Common success metrics apply, Single verification works for all items.
   B. Keep Separate When: 1. Different tools/methods per item, 2. Strict sequence dependencies exist, 3. Unique
safety protocols per item, 4. Individual verification required, 5. Different quality standards apply.

Please, think through this step:
1. What is the goal of this step?
2. What are the key constraints and requirements?
3. What safety factors need consideration?
4. What tools or resources are needed?

Then, break down the step into atomic actions, finally provide your response in this JSON format:
{
      "action": "[Action:verbnet verb] [Object:wordnet noun] ...",
      "tips": "Key safety and execution tips",
      "description": "Why this action and what to expect"
}
```

Table 17: This prompt is used for rule-based validation, compiling different rules results and generated procedures, then prompting the language model to produce comprehensive feedback and score.

```
Given validation results from rule checks, please analyze the findings and generate improvement recommendations.

Input:
      Original Prompt: {prompt}
      Generated Response: {response}
      Rule Check Results: {rules checking results}

Analysis Tasks:
      1. Review each rule validation result
      2. Identify patterns in rule violations
      3. Determine root causes
      4. Provide corrections feedback

Please provide your evaluation in the following JSON format:
{
      "reason": "Detailed analysis explaining the scoring rationale and key observations",
      "score": number
      "feedback": [
            "Clear and actionable improvement suggestions",
            "Each point focusing on a specific aspect to enhance",
            ...
      ]
}
```

Table 18: This prompt is utilized to optimize generated procedures based on verification results.

Given an original response and multiple feedback points, along with their corresponding refined versions, please synthesize a final refined response.

Original Response:{response}

Feedback Points:{feedbacks}

Rule Cheks: {rules checking results}

Please analyze all refinements and combine their improvements to generate an optimized final response that addresses all feedback points cohesively.

Please return only the final refined response.

Please provide your response refinements in the following JSON format:

```
{
    "reason": "Detailed analysis of why these refinements are suggested"
    "refined": {refine result}
}
```

Table 19: We employ this prompt to synthesize multiple candidate procedures into a consolidated summary.

Given multiple semantically similar items, analyze and synthesize them into a single, comprehensive and actionable summary.

Input:
{candidate list}

Follow these steps to analyze and summarize the items:

1. Semantic Analysis:
   - Identify core actions and objectives in each item
   - Find common elements and key differences
   - Extract the essential meaning and purpose

2. Content Synthesis:
   - Use specific verbs and nouns, avoid abstract terms.
   - Preserve all critical operational steps and important details.
   - Distill redundant information while keeping context-specific requirements.
   - Synthesize similar concepts into unified expressions.

3. Structured Output Requirements:
   - Title: Use concrete verbs that reflect the main task, include both the action and the target object, keep it brief but specific, make it immediately understandable.

   -Description: Start with the concrete purpose and scope, detail specific steps or methods, include key conditions or requirements, explain the expected outcome, use precise, actionable language.

   -Tips: List specific, executable actions, include crucial warnings or requirements, focus on practical guidance, avoid vague suggestions, synthesize similar tips into stronger, unified points.
4. Quality Verification:
   - Ensure it capture all essential information from original items.
   - Ensure all instructions specific and actionable.
   - Ensure all content logically organized.
   - Ensure the language clear and concrete.

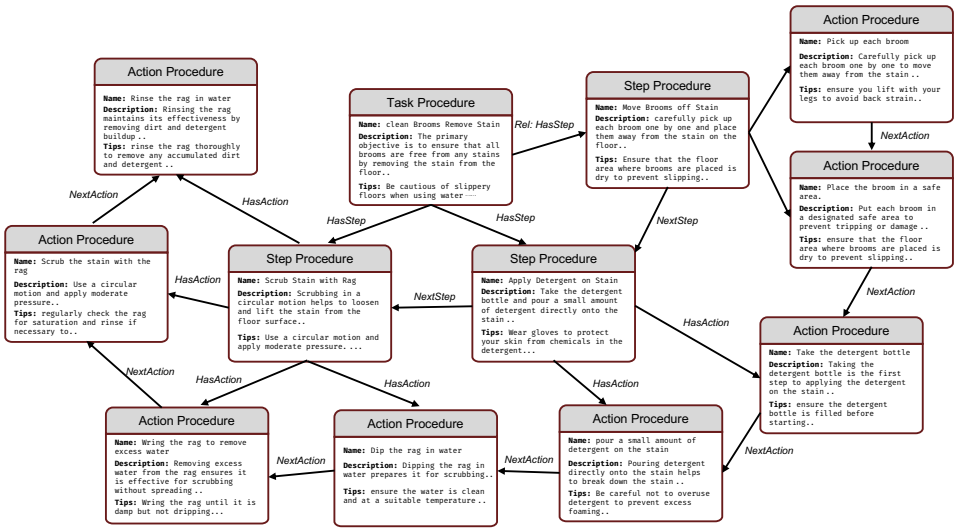

Figure 10: An example of cleaning and removing stains from brooms in our procedural knowledge graph.