DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning

Xiaohan Zhang[1], **Zainab Altaweel**[1], **Yohei Hayamizu**[1], **Yan Ding**[1], **Saeid Amiri**[1], **Hao Yang**[2], **Andy Kaminski**[2], **Chad Esselink**[2], **and Shiqi Zhang**[1]

[1]State University of New York at Binghamton
[2]Ford Research

Abstract

Vision-language models (VLMs) have been applied to robot task planning problems, where the robot receives a task in natural language and generates plans based on visual inputs. While current VLMs have demonstrated strong vision-language understanding capabilities, their performance is still far from being satisfactory in planning tasks. At the same time, although classical task planners, such as PDDL-based, are strong in planning for long-horizon tasks, they do not work well in open worlds where unforeseen situations are common. In this paper, we propose a novel task planning and execution framework, called DKPROMPT, which automates VLM prompting using domain knowledge in PDDL for classical planning in open worlds. Results from quantitative experiments show that DKPROMPToutperforms classical planning, pure VLM-based and a few other competitive baselines in task completion rate. [1]

1 Introduction

Prompting foundation models such as large language models (LLMs) and vision-language models (VLMs) require extensive domain knowledge and manual efforts, resulting in the so-called "prompt engineering" problem. One can provide examples explicitly (Brown et al. 2020) or implicitly (Lester, Al-Rfou, and Constant 2021), or encourage intermediate reasoning steps (Wei et al. 2022; Yao et al. 2024) to improve the performance of foundation models. Those methods, when applied to LLMs and VLMs, however, still lack the theoretical guarantee and provable correctness. Our idea is to leverage the foundation of classical AI, i.e., knowledge representation and reasoning, to develop a prompting strategy that enables the VLMs to verify the correctness of an agent's behavior at execution time in the real world.

Given the natural connection between planning symbols and human language, this paper investigates how pre-trained VLMs can assist the robot in realizing symbolic plans generated by classical planners while avoiding the engineering efforts of checking the outcomes of each action. Specifically, we propose a novel closed-loop task planning and execution framework called DKPROMPT, which prompts VLMs using domain knowledge in PDDL, generating visually grounded, provably correct task plans. DKPROMPT leverages VLMs to detect action failures and verify action affordances towards successful plan execution (Figure 2). We take advantage of the domain knowledge encoded in classical planners, including the actions defined by their effects and preconditions. By simply querying current observations against the action knowledge, similar to applying VLMs to Visual Question Answering (VQA) tasks, DKPROMPT can trigger the robot to repeat an unsuccessful action recovering from previous failures or call the symbolic planner to generate a new valid plan.

We conducted quantitative evaluations in the OmniGibson simulator (Li et al. 2023). We assume that robot actions are *imperfect* by nature, frequently causing *situations* [2] during execution (1). Results demonstrate that DKPROMPT utilizes domain knowledge to generate task plans adaptively, recovers from action failures, and re-plans when situations occur. In addition, we believe that researchers working on VLMs, robot planning, or both find our evaluation platform useful for their research. In particular, the open-world situations and structured world knowledge present a new playground for comparing robot planning and vision-language understanding using large-scale models.

2 Related Work

This section starts with covering a wide range of downstream applications of classical planners in symbolic task planning. It then explores the role of Large Language Models (LLMs) in robot planning, discussing their strengths (e.g., rich in common sense) and limitations (e.g., lack of correctness guarantee). Finally, it examines the recent advancements in vision-language models (VLMs) and their impact on the robotics community.

2.1 Classical Planning for Robots

Automated planning algorithms have a long-standing history in the literature of symbolic AI and have been widely used in robot systems. Shakey is the first robot equipped with a planning component, which was constructed using

[1]Project webpage: https://dkprompt.github.io/

[2]Situation is an unforeseen world state that potentially prevents an agent from completing a task using a solution that normally works (Ding et al. 2023a).

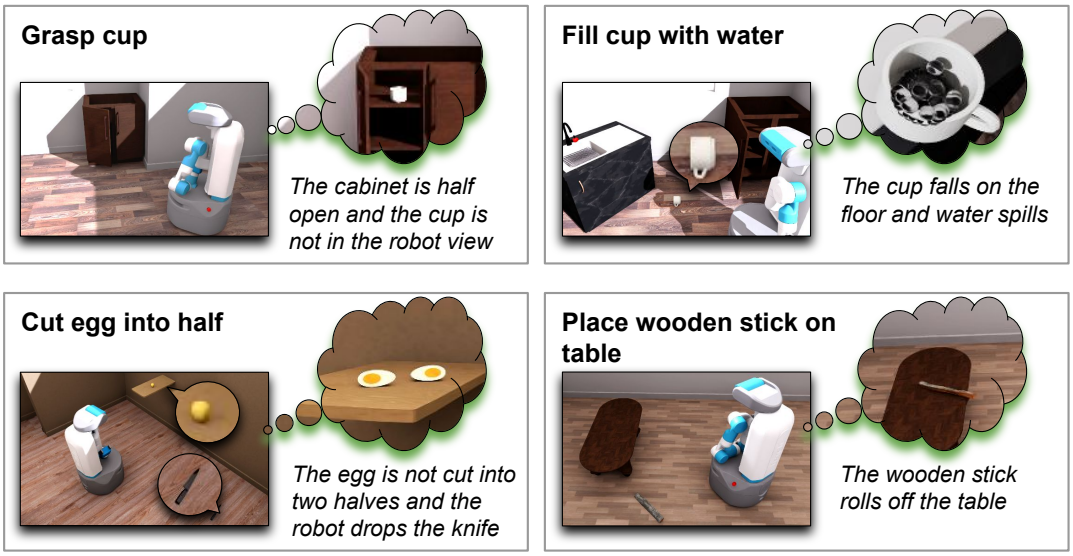

Figure 1: A few unforeseen situations during action execution. In the top-left example, the robot "opened" the cabinet door to get prepared for grasping the cup. It was expected that the cup in white would have been in the robot's view after the "opening" action, while a situation occurred, i.e., the cabinet was only half-open. DKPROMPT prompts vision-language models (VLMs) using domain knowledge to detect and address such situations. The goal is to compute visually grounded, provably correct plans.

STRIPS (Nilsson et al. 1984). Recent classical planning systems designed for robotics commonly employ Planning Domain Description Language (PDDL) or Answer Set Programming (ASP) as the underlying action language for planners (Jiang et al. 2019b; Brewka, Eiter, and Truszczyński 2011; Lifschitz 2002; Fox and Long 2003; Lagriffoul et al. 2018; Kaelbling and Lozano-Pérez 2013; Zhang et al. 2015; Ding et al. 2020; Jiang et al. 2019a; Ding et al. 2022). Most classical planning algorithms designed for robot planning do not consider perception. Though some recent works have already shown that training vision-based models from robot sensory data can be effective in plan feasibility evaluation (Zhu et al. 2021; Zhang et al. 2022b; Driess, Ha, and Toussaint 2020; Driess et al. 2020; Wells et al. 2019), their methods did not tightly bond with language symbols which are the state representations for classical planning systems. The most relevant work to our study is probably the research by Migimatsu and Bohg, which trained domain-specific predicate classifiers from webscale data and deployed on a robot planning system (Migimatsu and Bohg 2022). We propose DKPROMPT that investigates how off-the-shelf VLMs connect perception with symbolic language used to represent robot knowledge.

2.2 Classical Planning with Large Language Models for Robots

In the light of the recent advancement in artificial intelligence, many LLMs have been developed in recent years (Devlin et al. 2018; OpenAI 2023a; Chen et al. 2021; Zhang et al. 2022a). These LLMs can encode a large amount of common sense (Liu et al. 2023b) and have been widely applied to robot task planning (Kant et al. 2022; Huang et al. 2022a; Ahn et al. 2022; Huang et al. 2022b; Singh et al. 2022; Zhao, Lee, and Hsu 2023; Liu et al. 2022; Wu et al. 2023; Rana et al. 2023). However, a major drawback of existing LLMs is their lack of long-horizon reasoning/planning abilities for complex tasks (Valmeekam et al. 2022, 2023; OpenAI 2023b). Specifically, output plans LLMs produce for such tasks are often incomplete or unsatisfiable in solving the actual tasks. As a result, a wide range of studies have investigated approaches that combine the classical planning methodology with LLMs in robotic domains (Silver et al. 2022; Pallagani et al. 2022; Arora and Kambhampati 2023; Silver et al. 2024; Chen et al. 2023; Wang et al. 2024; Liu et al. 2023a; Stein and Koller 2023; Guan et al. 2023; Ding et al. 2023b). However, neither LLMs nor classical planners are inherently *grounded*, often necessitating complex interfaces to bridge the symbolic-continuous gap between language and robot perception. Our approach seeks to ground classical planners by utilizing pre-trained VLMs through a novel but straightforward domain knowledge prompting strategy.

2.3 Vision-language Models in Robotics

VLMs have emerged as powerful methods integrating visual and linguistic information for complex AI tasks (Zhang et al. 2024; Radford et al. 2021; Achiam et al. 2023; Team et al. 2023; Anthropic 2023). Researchers have started to employ such models in robot systems (Wake et al. 2023; Lykov et al. 2024; Guan et al. 2024; Majumdar et al. 2024; Sermanet et al. 2023), where these models have shown effectiveness in, for example, semantic scene understanding (Ha and Song 2022), open-ended agent learning (Fan et al. 2022), guiding robot navigation (Shafiullah et al. 2022) and manipulation behaviors (Shridhar, Manuelli, and Fox 2021; Stone et al. 2023). Recent VLMs have also been used for building *planning* frameworks (Lv et al. 2024; Zhao et al. 2023). Adaptive planning significantly improves task performance

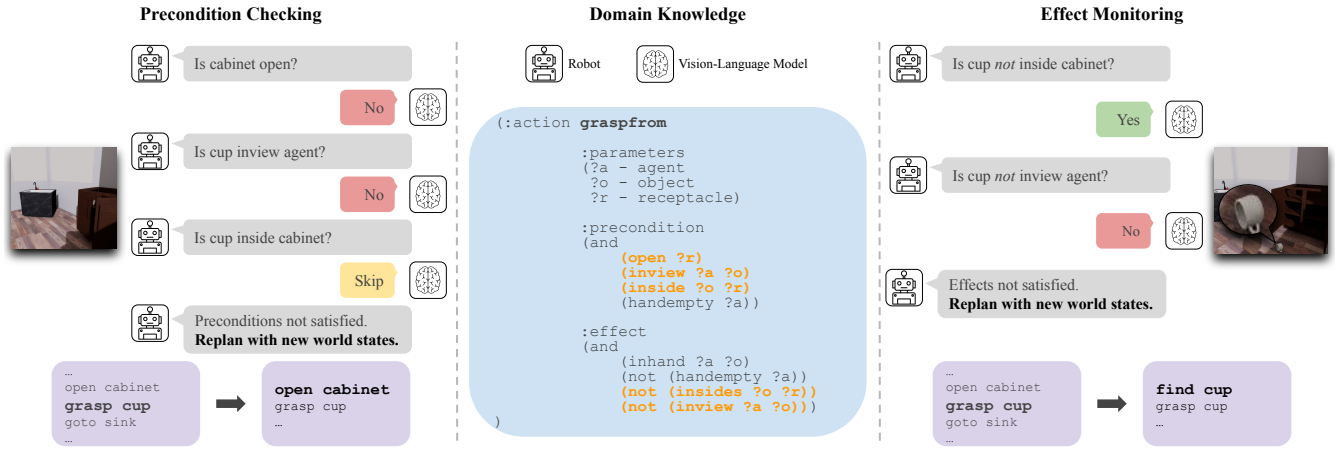

Figure 2: An overview of DKPROMPT. By simply querying the robot's current observation against the domain knowledge (i.e., action preconditions and effects) as VQA tasks, DKPROMPT can call the classical planner to generate a new valid plan using updated world states. Note that DKPROMPT only queries about predicates. The left shows how DKPROMPT checks every precondition of the action to be executed next, and the right shows how it verifies the expected action effects are all in place after action execution. After updating the planner's action knowledge, re-planning is triggered when preconditions or effects are unsatisfied.

through better environment awareness and fault recovery, and language understanding allows robots to seek human assistance in handling uncertainty (Ren et al. 2023; Zhi et al. 2024). There have been recent methods, which are similar to ours, that query VLMs for action success, failures, and affordances (Du et al. 2023; Driess et al. 2023; Guo et al. 2023). Different from those methods that rely on pre-trained models for planning, DKPROMPT uses classical planners to generate executable symbolic plans. Additionally, DKPROMPT builds the synergy between classical planners and VLMs by prompting with domain knowledge. As a result, DKPROMPT can perceive and handle unforeseen situations in open worlds while retaining the optimal planning proficiency of classical planning.

3 DKPROMPT for Planning in Open Worlds

This section presents the implementation details of DKPROMPT in a robot planning system. Robot actions are described in Sec. 3.1. We present the *action knowledge* that includes each action's preconditions and effects. These preconditions and effects are further represented as *objects* and *propositions*, i.e., predicates (Sec. 3.2). We then introduce how DKPROMPT takes advantage of the action knowledge for state update and online re-planning (Sec. 3.3).

3.1 Robot Actions

Our system considers ten actions (as listed in Table 1), including basic navigation and manipulation. Situations can occur after actions are successfully triggered by the agent, i.e., an unforeseen world state may be observed after an agent's action. Table 1 also provides examples of situations that happen following specific actions. Some of these situations impact the world states, while others do not. For example, the robot may fail on a "grasp" action, resulting in the target object, originally on the table, falling on the floor nearby (changing the state from `on(obj, table)` to `on(obj, floor)`). On the other hand, the object might also remain on the table, with the world states unchanged. To quantify the openness of different environments, we created the simulation platform in such a way that one can easily adjust the probability of a situation's occurrence. The source code of our benchmark system will be made available in our project website.

Actions are implemented in a discrete manner for simplification purposes since continuous action execution is not this paper's focus. For instance, "find" action is implemented by teleporting the agent from its initial position to a randomly sampled obstacle-free goal position near the target, and "fill" action is by adding fluid particles directly into the container that the robot is holding.

Actions are subject to several constraints. For example, "grasp" action is deemed executable only if the target object is in the agent's view (assuming vision-based manipulation) and the agent's hand is empty. Similarly, "cut" action is considered executable only if the object to be cut is in the agent's view and the agent is currently holding a knife. Calling an action with at least one unsatisfied constraint will result in an action failure, but without any changes to the world states. Note that such constraints are not made available to agents, instead, they are partially encoded as domain (action) knowledge that the agent possesses.

We assume that situations occur only during action execution and can be observed exclusively by agents either before or after the action execution phase. This assumption indicates that situations are solely caused by actions, and we are aware of a few recent robotic research that has started to consider more generalized situation handling (Ding et al. 2023a). We leave situations that are caused by external environmental factors (human or other embodiments) to future work.

3.2 Predicates

A single action is usually defined by multiple preconditions and effects in the domain knowledge. VLMs, especially

Table 1: Actions, constraints, and their uncertain outcomes.

Actions	Constraints	Situations
find	(1) The object and the agent are in the same room.	(1) The robot succeeds in navigation but the object is not inview. (2) There is no free space near the object so navigation fails. (3) The object that the robot is holding drops during navigation.
grasp	(1) The object is inview. (2) The agent's hand is empty.	(1) The robot fails to grasp, and the object position remains unchanged. (2) The robot fails to grasp, and the object drops nearby.
placein	(1) The object is inhand. (2) The receptacle is inview. (3) The receptacle is not closed.	(1) The robot fails to place, and the object remains in the robot's hand. (2) The robot fails to place, and the object drops nearby.
placeon	(1) The object is inhand. (2) The receptacle is inview.	(1) The robot fails to place, and the object remains in the robot's hand. (2) The robot fails to place, and the object drops nearby.
fillsink	(1) The sink is inview.	(1) The robot fails to open the faucet.
fill	(1) The container is inhand. (2) The agent is near sink. (3) The container is empty.	(1) The container is not fully filled. (2) The container drops nearby.
open	(1) The object is inview.	(1) The robot fails to open and the object remains closed.
close	(1) The object is inview.	(1) The robot fails to close and the object remains open.
turnon	(1) The object is inview.	(1) The robot fails to turn on the switch and the object remains off.
cut	(1) The object is inview. (2) A knife is inhand.	(1) The object is not cut into half, and the knife is still in the robot's hand. (2) The object is not cut into half, and the knife drops nearby.

Table 2: DKPROMPT assumptions for predicates.

Perceptible in vision	inview, closed, open, inside, halved, onfloor ontop, cooked
Perceptible in non-vision	handempty, inhand, hot
Imperceptible	turnedon, filled, inroom, insource

for those that are not trained using domain-specific data, frequently produce inaccurate answers that cause disagreements among the given preconditions (or effects). For instance, the VLM might answer "Yes" to both `on(apple, table)` and `inhand(apple)` after the robot picks up an apple from the table. In this paper, DKPROMPT categorizes predicates into three: *perceptible in vision*, *perceptible in non-vision*, and *imperceptible*, shown in Table 2. DKPROMPT will only ask about *"perceptible in vision"* predicates. Intuitively, we believe VLMs should be and will be only good at visually-perceptible predicates. The robot will then have ground truth access to *perceptible in non-vision* predicates (this assumption also applies to all other baselines). We leave identifying these predicates using more advanced Multimodal Language Models to future work. As for the remaining *imperceptible* predicates, the DKPROMPT agent maintains a positive attitude and assumes they are always True. This suggests that DKPROMPT believes these predicates will never be affected by any situation.

3.3 DKPROMPT

Before every action execution, DKPROMPT extracts knowledge about action preconditions from the planner's domain description. For instance, as indicated in Figure 2, action `graspfrom(`a`, `o`, `r`)` has preconditions of `open(`r`)`, `inview(`a`, `o`)`, `inside(`o`, `r`)`, and `handempty(`a`)`, meaning that to grasp an object o from a receptacle r, r should be open (not closed), o should be in the agent's current first person view, o should be inside r, and the agent's hand should be empty. Then, we convert each action precondition into a natural language query by using manually defined templates, though it has been evident that LLMs can be used to translate PDDL and natural language (Liu et al. 2023a). Examples include *"Is* $\langle o \rangle$ *inview agent?"* and *"Is* $\langle o \rangle$ *inside* $\langle r \rangle$*?"* Paring each natural language query with the current observation from the robot's first-person view, we call the VLM to get answers indicating if the precondition is satisfied. The answers are either "yes", "no", or "skip" if unsure.

According to the answers from the VLM, DKPROMPT updates the current state information in the classical planning system. Figure 2 (Left) shows an example where the robot wants to execute `graspfrom(cup, cabinet)` but fails to detect *"cabinet is open"*, *"cup is inview of agent"*, and is suspicious about if *"cup is in the cabinet"* (i.e., the VLM answers "skip" to this question) given the current observation. As a result, DKPROMPT updates the current state by changing `open(cabinet)` to `closed(cabinet)`, and removing `inview(agent, cup)`. `inside(cup, cabinet)` remains the same because we do not update the state if the VLM answers "skip", indicating the agent holds a positive attitude that situations will not commonly occur. We then provide the updated world state to the classical planner as the "new" initial state to re-generate a plan. In the above example, instead of `graspfrom(cup, cabinet)`, the robot takes the action of `open(cabinet)` again according to the newly-generated action plan. After every action execution, DKPROMPT extracts knowledge about action effects from

Table 3: Task descriptions and initial plan length.

Name	Descriptions	Initial plan length
boil water in the microwave	Pick up an empty cup in a closed cabinet, fill it with water using a sink, and boil it in a microwave.	12
bring in empty bottle	Find two empty bottles in the garden and bring them inside.	8
cook a frozen pie	Take an apple pie out of the fridge and heat it using an oven.	8
halve an egg	Find a knife in the kitchen and use it to cut a hard-boiled egg into half.	4
store firewood	Collect two wooden sticks and place them on a table.	8

the planner's domain description, illustrated in Figure 2 (Right). It queries action effects by using the VLM. If the effects are not satisfied, the robot will update its belief on the current states and re-plan accordingly. The knowledge-based automated prompting strategy of VLMs enables our planning system to adaptively capture and handle unforeseen situations at execution time.

We show the prompt template that was used to query the VLMs. The prompt includes three components: "System," "DKPROMPT," and an image. All prompts share the "System" part to contextualize the interaction with the VLMs and specify the output format. The "DKPROMPT" part lists the questions that are translated from the domain knowledge in PDDL format using the manually defined templates. Each VLM prompt includes a 256x256 image taken from the current robot's observation ("a cup in an open cabinet" in the following example).

DKPROMPT

System: Imagine you are an intelligent agent that can answer questions based on what you see. You will be given a single image as the agent's current view, and one or more yes/no question(s) asking about the image. Questions will be separated by semicolon. For each question, you should answer "yes", "no", or "skip" without any explanation. Answer "yes" or "no" only if you are pretty sure about what you see in the image. It's fine to answer "skip" to skip the question if you are not confident about your answer. Answers should be separated by semicolon (e.g., "yes;no;skip" for three questions).

DKPROMPT: Is cup inview agent?;Is cup inside cabinet?;Is cabinet open?

4 Experiments

We conducted extensive experiments to evaluate the performance of DKPROMPT comparing with baselines from the literature. Our hypothesis is DKPROMPT produces the highest task completion rate because of its effectiveness in plan monitoring and online re-planning using domain knowledge and perception.

4.1 Experiment Setup

Quantitative evaluation results are collected in the OmniGibson simulator (Li et al. 2023). The agent is equipped with a set of skills, and aims to use its skills to interact with the environment, completing long-horizon tasks autonomously. In the experiment, we consider five everyday tasks that are "boil water in the microwave", "bring in empty bottle","cook a frozen pie", "halve an egg", and "store firewood". Their detailed descriptions are shown in Table 3. These five tasks are originally from the Behavior 1K benchmark (Li et al. 2023) that are accompanied with the simulator. Task descriptions including initial and goal states are written in PDDL and symbolic plans are generated using the fast-downward planner (Helmert 2006).

4.2 Results

Comparisons with Baselines. Figure 3 presents the main experimental results and details the comparative success rates of DKPROMPT and five methods from the literature. The baseline methods include:

- VLM-planner, which uses the VLM as a planner to generate task plans, similar to (Huang et al. 2022a). For fair comparisons, we also provide domain knowledge (as natural language) in the prompts for the VLM.
- Classical-planner, which is a typical classical planning approach without perception, assuming all action executions are successful;
- Suc.-QA, which uses a classical planner to generate plans, and asks about action success after each action execution. This baseline is inspired by (Du et al. 2023), and we use the same query provided in their paper, which is *"Did the robot successfully <action>?"* Suc.-QA does not consider if the next action is executable;
- Aff.-QA, which uses a classical planner to generate plans, and asks about action affordance before each action execution. This baseline is designed with prompts provided in the original PaLM-E paper (Driess et al. 2023), which are *"Is it possible to <action> here?"* and

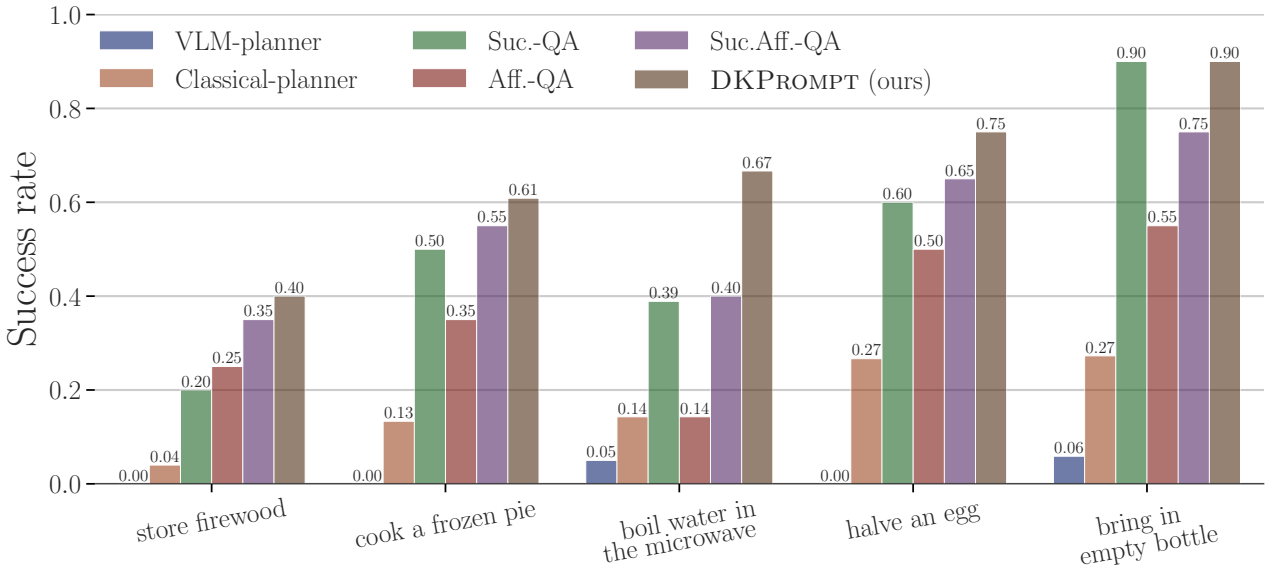

Figure 3: DKPROMPT v.s. baselines in success rate over five everyday tasks. The most competitive baseline is "Suc.Aff.-QA" that includes a classical planning component and reasons about both action affordances and action effects.

"Was <action> successful?" Aff.-QA does not consider whether the previous action is successful;

- Suc.Aff.-QA, which uses a classical planner, asks about both action affordance (before each action execution) and action success (after each action execution), similar to (Huang et al. 2022b). [3]

When using VLM itself as the planner, the agent frequently fails in finding an executable plan, resulting in the lowest success rate. This finding is consistent with recent work (Valmeekam et al. 2022) and motivates the development of other research that combines classical planning with large models (Liu et al. 2023a). Classical-planner, which operates without visual feedback during task execution, shows the second lowest success rate across five tasks compared to other evaluated methods, highlighting its limited effectiveness in handling situations and recovering from potential action failures. In contrast, methods that involve querying for action affordances, success probabilities, or both, acheive much higher success rates as compared to the "blind" classical planning approach. This improvement demonstrates the general advantage of incorporating visual feedback and high-level reasoning in task planning systems. While it is always a good practice to verify both before and after an action (like Suc.Aff.-QA), we found that Suc.-QA also surpasses the performance of Aff.-QA, indicating that there is a greater positive impact on task completion from action failure recovery, and VLMs have better zero-shot reasoning capabilities on the direct effects caused by actions.

We observed that DKPROMPT consistently outperforms baselines in task completion rates, which supports our hypothesis. By incorporating domain knowledge (i.e., action preconditions and effects) for prompting, DKPROMPT is significantly better than other methods, including Suc.Aff.-QA that also cares about affordance prediction and failure detection. However, Suc.Aff.-QA queries about actions solely by their names, which provides less information than the detailed domain knowledge used by DKPROMPT, indicating that action knowledge is more informative for pretrained VLMs to reason over.

[3]We use the same VLM as ours (GPT4) for implementing all baselines that require a VLM.

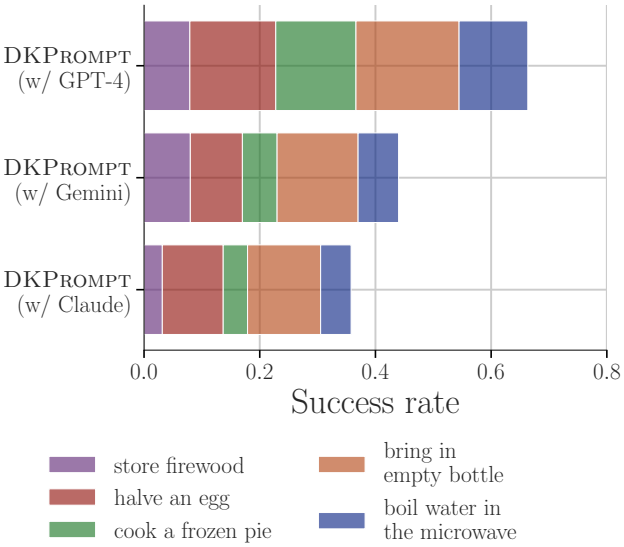

Figure 4: Three implementations of DKPROMPT using different off-the-shelf VLMs (in five tasks).

Ablation Study on Preconditions and Effects. Table 4 presents an ablation study comparing the performance of different versions of our approach across the same set of

Table 4: Ablation study on preconditions and effects. The results justify the necessity of prompting VLMs with both action preconditions and effects, where the knowledge is extracted from classical planners.

#	Methods	Tasks					avg. (%)
		boil water in the microwave	bring in empty bottle	cook a frozen pie	halve an egg	store firewood	
Ours							
1	DKPROMPT	**66.7**	90.0	**60.9**	**75.0**	**40.0**	**66.5**
Ablation							
2	Eff.-only	50.0	**93.8**	26.7	66.7	28.0	53.0
3	Pre.-only	17.6	75.0	35.0	55.0	20.0	41.5

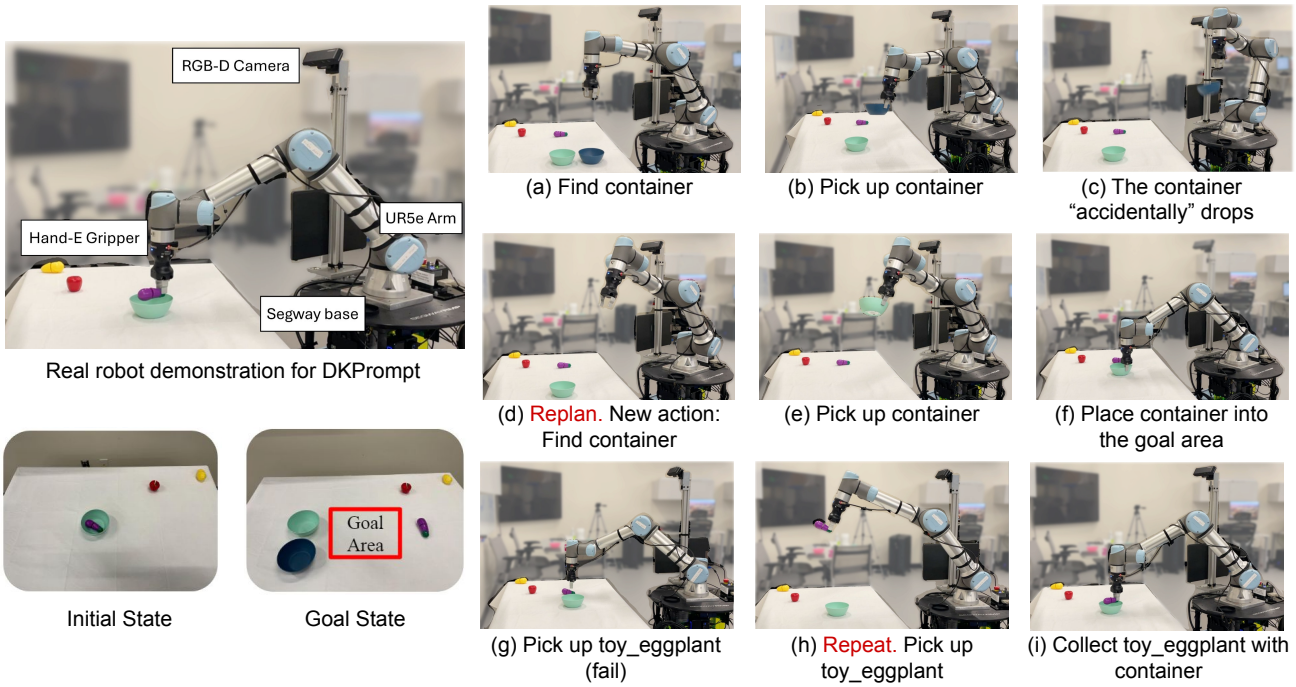

Figure 5: Screenshots showing the full demonstration trial of DKPROMPT as applied to a real robot.

tasks. DKPROMPT integrates both action effects and action preconditions, while we are also curious to know how they affect the overall task completion independently. DKPROMPT achieves an average success rate of 66.5%. For ablation methods where only action effects are considered (Eff.-only), the average success rate drops to 53.0%, and for methods considering only preconditions (Pre.-only), it further decreases to 41.5%. This suggests that the integration of both effects and preconditions in DKPROMPT significantly enhances task performance compared to considering these components separately.

Performance of Other VLMs. We also run experiments on various VLMs, including GPT-4 (as being used in the original implementation of DKPROMPT) from OpenAI (OpenAI 2023b), Gemini 1.5 from Google (Reid et al. 2024), and Claude 3 from Anthropic. Figure 4 shows that GPT-4 consistently performs better than Gemini and Claude. By looking at the highest accuracy among all the VLMs (i.e., less than 65%), our evaluation benchmark (designed with challenging open-world situations and rich domain knowledge) presents a simulation platform, dataset and success criteria that other researchers working on AI planning, VLMs or both might find useful. We will open source the benchmark including software and data to the public after the anonymous review phase.

4.3 Real-Robot Deployment

We also deployed DKPROMPT on real robot hardware to perform object rearrangement tasks (Figure 5), where the goal is to "collect" toys using a container and place them in the middle of the table (i.e., goal area). Our real-robot setup includes a UR5e Arm with a Hand-E gripper mounted on a Segway base, and an overhead RGB-D camera (rela-

tively fixed to the robot) for perception. We assume that the robot has a predefined set of skills, including `pick`, `place`, and `find`. `Pick` and `place` actions are implemented using GG-CNN (Morrison, Corke, and Leitner 2018), and `find` action simply uses base rotation for capturing tabletop images from different angles.

Given the task description, the robot first decided to execute "find container" and "pick up container". These two actions were successfully executed as shown in Figure 5(a), 5(b). When the robot was preparing for the next action (i.e., "Place container into the goal area"), the blue container accidentally dropped from the robot's gripper to the ground (Figure 5(c)). Instead of directly executing the next action, DKPROMPT enabled the robot to check preconditions by querying the VLM *"Is the container in a robot's hand?"* After receiving negative feedback from the VLM, DKPROMPT updated the world state by removing `in_hand(container)` and called the planner to generate a new plan that started the task again by finding another container (Figure 5(d)). Then the robot picked up the cyan container and placed it in the middle of the table as shown in Figure 5(e), 5(f). The subsequent actions in the plan were to find and pick up a toy, but the `pick` action failed (Figure 5(g)). DKPROMPT managed to detect the failure by querying 1) *"Is there a toy_eggplant on the table?"*, and 2) *"Is the toy_eggplant in a robot's hand?"*, and receiving Yes and No answers respectively. As a result, our system suggested the robot repeat the `pick` action again (Figure 5(h)). Finally, the robot successfully collected the toy by putting it into the cyan container that was previously placed in the goal area (Figure 5(i)).

5 Conclusion

In this paper, we built the synergy between classical planners and vision-language models (VLMs). We propose DKPROMPT which is unique in prompting VLMs with domain knowledge in PDDL and leveraging the VLM output for plan monitoring. DKPROMPT is able to perceive and handle unforeseen situations in open worlds while retaining the optimal planning proficiency of classical planning. Experimental results demonstrate that DKPROMPT adaptively generates visually-grounded task plans, recovers from action failures and re-plans when situations occur, outperforming classical planning, pure VLM-based and a few other competitive baselines.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Anthropic. 2023. Claude 3 Family. Accessed: 2024-05-21.

Arora, D.; and Kambhampati, S. 2023. Learning and Leveraging Verifiers to Improve Planning Capabilities of Pre-trained Language Models. *arXiv preprint arXiv:2305.17077*.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM*, 54(12): 92–103.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, Y.; Arkin, J.; Zhang, Y.; Roy, N.; and Fan, C. 2023. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. *arXiv preprint arXiv:2306.06531*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, Y.; Zhang, X.; Amiri, S.; Cao, N.; Yang, H.; Kaminski, A.; Esselink, C.; and Zhang, S. 2023a. Integrating action knowledge and LLMs for task planning and situation handling in open worlds. *Autonomous Robots*, 47(8): 981–997.

Ding, Y.; Zhang, X.; Paxton, C.; and Zhang, S. 2023b. Task and Motion Planning with Large Language Models for Object Rearrangement. *arXiv preprint arXiv:2303.06247*.

Ding, Y.; Zhang, X.; Zhan, X.; and Zhang, S. 2020. Task-Motion Planning for Safe and Efficient Urban Driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Ding, Y.; Zhang, X.; Zhan, X.; and Zhang, S. 2022. Learning to ground objects for robot task and motion planning. *IEEE Robotics and Automation Letters*, 7(2): 5536–5543.

Driess, D.; Ha, J.-S.; and Toussaint, M. 2020. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. *arXiv preprint arXiv:2006.05398*.

Driess, D.; Oguz, O.; Ha, J.-S.; and Toussaint, M. 2020. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9563–9569. IEEE.

Driess, D.; Xia, F.; Sajjadi, M. S.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. 2023. PaLM-E: An Embodied Multimodal Language Model. *arXiv preprint arXiv:2303.03378*.

Du, Y.; Konyushkova, K.; Denil, M.; Raju, A.; Landon, J.; Hill, F.; de Freitas, N.; and Cabi, S. 2023. Vision-Language Models as Success Detectors. *arXiv preprint arXiv:2303.07280*.

Fan, L.; Wang, G.; Jiang, Y.; Mandlekar, A.; Yang, Y.; Zhu, H.; Tang, A.; Huang, D.-A.; Zhu, Y.; and Anandkumar, A. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv preprint arXiv:2206.08853*.

Fox, M.; and Long, D. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 20: 61–124.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36: 79081–79094.

Guan, L.; Zhou, Y.; Liu, D.; Zha, Y.; Amor, H. B.; and Kambhampati, S. 2024. " Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors. *arXiv preprint arXiv:2402.04210*.

Guo, Y.; Wang, Y.-J.; Zha, L.; Jiang, Z.; and Chen, J. 2023. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. *arXiv preprint arXiv:2307.00329*.

Ha, H.; and Song, S. 2022. Semantic abstraction: Open-world 3d scene understanding from 2d vision-language models. In *Conference on Robot Learning*.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, 9118–9147. PMLR.

Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; et al. 2022b. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Jiang, Y.; Yedidsion, H.; Zhang, S.; Sharon, G.; and Stone, P. 2019a. Multi-robot planning with conflicts and synergies. *Autonomous Robots*, 43(8): 2011–2032.

Jiang, Y.-q.; Zhang, S.-q.; Khandelwal, P.; and Stone, P. 2019b. Task planning in robotics: an empirical comparison of pddl-and asp-based systems. *Frontiers of Information Technology & Electronic Engineering*, 20: 363–373.

Kaelbling, L. P.; and Lozano-Pérez, T. 2013. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10): 1194–1227.

Kant, Y.; Ramachandran, A.; Yenamandra, S.; Gilitschenski, I.; Batra, D.; Szot, A.; and Agrawal, H. 2022. Housekeep: Tidying virtual households using commonsense reasoning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, 355–373. Springer.

Lagriffoul, F.; Dantam, N. T.; Garrett, C.; Akbari, A.; Srivastava, S.; and Kavraki, L. E. 2018. Platform-independent benchmarks for task and motion planning. *IEEE Robotics and Automation Letters*, 3(4): 3765–3772.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, C.; Zhang, R.; Wong, J.; Gokmen, C.; Srivastava, S.; Martín-Martín, R.; Wang, C.; Levine, G.; Lingelbach, M.; Sun, J.; et al. 2023. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, 80–93. PMLR.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2): 39–54.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Liu, W.; Hermans, T.; Chernova, S.; and Paxton, C. 2022. Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects. In *Workshop on Language and Robotics at CoRL 2022*.

Lv, Q.; Li, H.; Deng, X.; Shao, R.; Wang, M. Y.; and Nie, L. 2024. RoboMP2: A Robotic Multimodal Perception-Planning Framework with Mutlimodal Large Language Models. In *International Conference on Machine Learning*.

Lykov, A.; Litvinov, M.; Konenkov, M.; Prochii, R.; Burtsev, N.; Abdulkarim, A. A.; Bazhenov, A.; Berman, V.; and Tsetserukou, D. 2024. Cognitivedog: Large multimodal model based system to translate vision and language into action of quadruped robot. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 712–716.

Majumdar, A.; Ajay, A.; Zhang, X.; Putta, P.; Yenamandra, S.; Henaff, M.; Silwal, S.; Mcvay, P.; Maksymets, O.; Arnaud, S.; et al. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16488–16498.

Migimatsu, T.; and Bohg, J. 2022. Grounding predicates through actions. In *2022 International Conference on Robotics and Automation (ICRA)*, 3498–3504. IEEE.

Morrison, D.; Corke, P.; and Leitner, J. 2018. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*.

Nilsson, N. J.; et al. 1984. Shakey the robot.

OpenAI. 2023a. ChatGPT. Accessed: 2023-02-08. Cit. on pp. 1, 16.

OpenAI. 2023b. GPT-4 Technical Report. arXiv:2303.08774.

Pallagani, V.; Muppasani, B.; Murugesan, K.; Rossi, F.; Horesh, L.; Srivastava, B.; Fabiano, F.; and Loreggia, A. 2022. Plansformer: Generating symbolic plans using transformers. *arXiv preprint arXiv:2212.08681*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.;

et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Rana, K.; Haviland, J.; Garg, S.; Abou-Chakra, J.; Reid, I.; and Suenderhauf, N. 2023. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. *arXiv preprint arXiv:2307.06135*.

Reid, M.; Savinov, N.; Teplyashin, D.; Lepikhin, D.; Lillicrap, T.; Alayrac, J.-b.; Soricut, R.; Lazaridou, A.; Firat, O.; Schrittwieser, J.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Ren, A. Z.; Dixit, A.; Bodrova, A.; Singh, S.; Tu, S.; Brown, N.; Xu, P.; Takayama, L.; Xia, F.; Varley, J.; et al. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*.

Sermanet, P.; Ding, T.; Zhao, J.; Xia, F.; Dwibedi, D.; Gopalakrishnan, K.; Chan, C.; Dulac-Arnold, G.; Maddineni, S.; Joshi, N. J.; et al. 2023. RoboVQA: Multimodal Long-Horizon Reasoning for Robotics. *arXiv preprint arXiv:2311.00899*.

Shafiullah, N. M. M.; Paxton, C.; Pinto, L.; Chintala, S.; and Szlam, A. 2022. CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory. *arXiv preprint arXiv: Arxiv-2210.05663*.

Shridhar, M.; Manuelli, L.; and Fox, D. 2021. CLIPort: What and Where Pathways for Robotic Manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*.

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L.; and Katz, M. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20256–20264.

Silver, T.; Hariprasad, V.; Shuttleworth, R. S.; Kumar, N.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. PDDL planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2022. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

Stein, K.; and Koller, A. 2023. AutoPlanBench:: Automatically generating benchmarks for LLM planners from PDDL. *arXiv preprint arXiv:2311.09830*.

Stone, A.; Xiao, T.; Lu, Y.; Gopalakrishnan, K.; Lee, K.-H.; Vuong, Q.; Wohlhart, P.; Zitkovich, B.; Xia, F.; Finn, C.; and Hausman, K. 2023. Open-World Object Manipulation using Pre-Trained Vision-Language Model. In *arXiv preprint*.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36: 75993–76005.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *arXiv preprint arXiv:2206.10498*.

Wake, N.; Kanehira, A.; Sasabuchi, K.; Takamatsu, J.; and Ikeuchi, K. 2023. GPT-4V (ision) for robotics: Multimodal task planning from human demonstration. *arXiv preprint arXiv:2311.12015*.

Wang, S.; Han, M.; Jiao, Z.; Zhang, Z.; Wu, Y. N.; Zhu, S.-C.; and Liu, H. 2024. LLMˆ 3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning. *arXiv preprint arXiv:2403.11552*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wells, A. M.; Dantam, N. T.; Shrivastava, A.; and Kavraki, L. E. 2019. Learning feasibility for task and motion planning in tabletop environments. *IEEE robotics and automation letters*, 4(2): 1255–1262.

Wu, J.; Antonova, R.; Kan, A.; Lepert, M.; Zeng, A.; Song, S.; Bohg, J.; Rusinkiewicz, S.; and Funkhouser, T. 2023. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 47(8): 1087–1102.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zhang, D.; Yu, Y.; Li, C.; Dong, J.; Su, D.; Chu, C.; and Yu, D. 2024. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, S.; Yang, F.; Khandelwal, P.; and Stone, P. 2015. Mobile Robot Planning Using Action Language BC with an Abstraction Hierarchy. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 502–516. Springer.

Zhang, X.; Zhu, Y.; Ding, Y.; Zhu, Y.; Stone, P.; and Zhang, S. 2022b. Visually grounded task and motion planning for mobile manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, 1925–1931. IEEE.

Zhao, X.; Li, M.; Weber, C.; Hafez, M. B.; and Wermter, S. 2023. Chat with the environment: Interactive multimodal perception using large language models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3590–3596. IEEE.

Zhao, Z.; Lee, W. S.; and Hsu, D. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhi, P.; Zhang, Z.; Han, M.; Zhang, Z.; Li, Z.; Jiao, Z.; Jia, B.; and Huang, S. 2024. Closed-Loop Open-Vocabulary Mobile Manipulation with GPT-4V. arXiv:2404.10220.

Zhu, Y.; Tremblay, J.; Birchfield, S.; and Zhu, Y. 2021. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 6541–6548. IEEE.

Appendix

This appendix document presents additional information about our DKPROMPT work. DKPROMPT assists robots in open-world planning tasks by leveraging domain knowledge to automate vision-language model (VLM) prompting. In this appendix document, we present our domain knowledge in PDDL format, environment settings of open-world planning, and additional experiment results. The main goal of this appendix is to improve the reproducibility of this research and we hope robot learning practitioners finds it useful. Note that other than this appendix, we have a webpage (https://dkprompt.github.io/) that serves as a central place where people can find relevant documents about DKPROMPT.

We show Table 5 that presents the parameters for specifying the **openness** of the simulation environments used for experiments. This table overlaps with Table 1 in the main paper on the list of actions and situations. Beyond that, we present the probability of each individual situation taking place in the execution of the corresponding action. One can realize testing domains with different levels of openness by adjusting those probabilities. There are situations whose probabilities are out of our control, which are labeled "N/A" in the table. For instance, the occurrence of "object is not inview" depends on the robot's motion planner used for navigation. As a result, we are sure that there exist such situations in the experiments but the chance cannot be specified. We also present the checkpoints of the VLMs used in the experiments of this research (Table 6). We tried our best to use the state-of-the-art VLMs when the experiments were conducted.

Next, we present Table 7 that includes the complete results of a number of methods, where each column corresponds to a different task. The table includes four parts: ours, baselines, ablations and other VLMs. The "Baseline in Literature" part corresponds to the results presented in Figure 3. The "Ablations" part corresponds to the results presented in Table 4. The "Other VLMs" part corresponds to the results presented in Figure 4. Overall, we do not see a huge variance in those methods' performance in different tasks, indicating that our claims about the superiority of DKPROMPT are valid in different tasks.

Finally, the set of PDDL-formatted domain knowledge provided for the robot to perform task planning and VLM prompting is shown in Figures 6 through 9. In line with all PDDL-based planning systems, the domain knowledge includes a complete description of the robot's actions (e.g., "find" and "graspon"), where each action is specified by its preconditions and effects. Note that we only present the domain description file here. A complete planning problem would further require a problem description that includes a description of the current and goal states. Since the problem description file changes in each trial, please refer to our GitHub page (link available on the project page shared at the beginning of this appendix) on the instructions of extracting it from the simulator.

Table 5: Actions and their situation parameters.

	Uncertaint outcomes	
Actions	Situations	Prob
find	(1) The robot succeeds in navigation but the object is not inview.	N/A
	(2) There is no free space near the object so navigation fails.	N/A
	(3) The object that the robot is holding drops during navigation.	0.1
grasp	(1) The robot fails to grasp, and the object position remains unchanged.	0.25
	(2) The robot fails to grasp, and the object drops nearby.	0.25
placein	(1) The robot fails to place, and the object remains in the robot's hand.	0.1
	(2) The robot fails to place, and the object drops nearby.	0.1
placeon	(1) The robot fails to place, and the object remains in the robot's hand.	0.1
	(2) The robot fails to place, and the object drops nearby.	0.1
fillsink	(1) The robot fails to open the faucet.	0.1
fill	(1) The container is not fully filled.	0.05
	(2) The container drops nearby.	0.05
open	(1) The robot fails to open and the object remains closed.	0.1
close	(1) The robot fails to close and the object remains open.	0.1
turnon	(1) The robot fails to turn on the switch and the object remains off.	0.1
cut	(1) The object is not cut into half, and the knife is still in the robot's hand.	0.25
	(2) The object is not cut into half, and the knife drops nearby.	0.25

Table 6: Model checkpoints we used for off-the-shelf VLMs.

VLM	**Model**
GPT-4	gpt-4-turbo
Gemini	gemini-1.5-pro
Claude	claude-3-opus-20240229

Table 7: Full results with the total numbers of trials and successful trials. The very right column ("avg.") reports the average success rates over all tasks, which are already reported in the main paper. The other columns present a breakdown over different tasks.

#	Methods	Tasks					avg. (%)
		boil water in the microwave	bring in empty bottle	cook a frozen pie	halve an egg	store firewood	
Ours							
1	DKPROMPT (w/ GPT-4)	12/18	18/20	14/23	15/20	8/20	**66.5**
Baselines in Literature							
2	VLM-planner	1/20	2/34	0/20	0/19	0/22	2.2
3	Classical-planner	5/35	3/11	4/30	8/30	1/25	17.1
4	Aff.-QA	4/28	11/20	7/20	10/20	5/20	35.9
5	Suc.-QA	7/18	18/20	10/20	12/20	4/20	51.8
6	Suc.Aff.-QA	8/20	12/16	11/20	13/20	7/20	54.0
Ablations							
7	Eff.-only	15/30	15/16	4/15	10/15	7/25	53.0
8	Pre.-only	3/17	15/20	7/20	11/20	5/20	41.5
Other VLMs							
9	DKPROMPT (w/ Gemini-1.5)	7/20	14/20	6/20	9/20	8/20	44.0
10	DKPROMPT (w/ Claude-3)	5/20	12/28	4/14	10/20	3/13	33.9

```
(define (domain omnigibson)

    (:requirements :strips :typing :
        negative-preconditions :
        conditional-effects)

    (:types
        movable liquid furniture room
            agent - object

        wooden_stick tupperware brownie
            beer_bottle water_bottle mug
            pie carving_knife
            hard__boiled_egg - movable
        water - liquid
        countertop electric_refrigerator
             oven cabinet sink floor
            microwave table - furniture

        kitchen living_room - room

        water-n-06 - water
        mug-n-04 - mug
        cabinet-n-01 - cabinet
        sink-n-01 - sink
        floor-n-01 - floor
        microwave-n-02 - microwave
        pie-n-01 - pie
        oven-n-01 - oven
        electric_refrigerator-n-01 -
            electric_refrigerator
        carving_knife-n-01 -
            carving_knife
        countertop-n-01 - countertop
        hard__boiled_egg-n-01 -
            hard__boiled_egg
        water_bottle-n-01 - water_bottle
        beer_bottle-n-01 - beer_bottle
        brownie-n-03 - brownie
        tupperware-n-01 - tupperware
        wooden_stick-n-01 - wooden_stick
        table-n-02 - table

        agent-n-01 - agent
    )
```

Figure 6: Domain knowledge in PDDL format (Part 1/4).

```
    (:predicates
        (inside ?o1 - object ?o2 -
            object)
        (insource ?s - sink ?w - liquid)
        (inroom ?o - object ?r - room)
        (inhand ?a - agent ?o - object)
        (inview ?a - agent ?o - object)
        (handempty ?a - agent)
        (closed ?o - object)
        (filled ?o - movable ?w - liquid
            )
        (filledsink ?s - sink ?w -
            liquid)
        (turnedon ?o - object)
        (cooked ?o - object)
        (found ?a - agent ?o - object)
        (frozen ?o - object)
        (hot ?o - object)
        (halved ?o - object)
        (onfloor ?o - object ?f - floor)
        (ontop ?o1 - object ?o2 - object
            )
    )

    (:action find
        :parameters (?a - agent ?o -
            object ?r - room)
        :precondition (and (inroom ?a ?r
            ) (inroom ?o ?r))
        :effect (and (inview ?a ?o) (
            found ?a ?o) (forall
                (?oo - object)
                (when
                    (found ?a ?oo)
                    (not (found ?a ?oo))
                        )))
    )

    (:action graspon
        :parameters (?a - agent ?o1 -
            movable ?o2 - object)
        :precondition (and (inview ?a ?
            o1) (found ?a ?o1) (handempty
             ?a) (ontop ?o1 ?o2))
        :effect (and (not (inview ?a ?o1
            )) (not (handempty ?a)) (
            inhand ?a ?o1) (not (ontop ?
            o1 ?o2)))
    )

    (:action graspin
        :parameters (?a - agent ?o1 -
            movable ?o2 - object)
        :precondition (and (inview ?a ?
            o1) (found ?a ?o1) (handempty
             ?a) (inside ?o1 ?o2))
        :effect (and (not (inview ?a ?o1
            )) (not (handempty ?a)) (
            inhand ?a ?o1) (not (inside ?
            o1 ?o2)))
    )
```

Figure 7: Domain knowledge in PDDL format (Part 2/4).

```
    (:action placein
        :parameters (?a - agent ?o1 -
            movable ?o2 - object)
        :precondition (and (not (
            handempty ?a)) (inhand ?a ?o1
            ) (inview ?a ?o2) (found ?a ?
            o2) (not (closed ?o2)))
        :effect (and (handempty ?a) (not
             (inhand ?a ?o1)) (inside ?o1
             ?o2) (forall
                (?oo - object)
                (when
                    (inside ?oo ?o1)
                    (inside ?oo ?o2))
            ))
    )

    (:action placeon
        :parameters (?a - agent ?o1 -
            movable ?o2 - object)
        :precondition (and (not (
            handempty ?a)) (inhand ?a ?o1
            ) (inview ?a ?o2) (found ?a ?
            o2))
        :effect (and (handempty ?a) (not
             (inhand ?a ?o1)) (ontop ?o1
            ?o2))
    )

    (:action fillsink
        :parameters (?a - agent ?s -
            sink ?w - liquid)
        :precondition (and (inview ?a ?s
            ) (found ?a ?s) (insource ?s
            ?w))
        :effect (filledsink ?s ?w)
    )

    (:action fill
        :parameters (?a - agent ?o -
            movable ?s - sink ?w - liquid
            )
        :precondition (and (inhand ?a ?o
            ) (not (handempty ?a)) (
            filledsink ?s ?w) (inview ?a
            ?s) (found ?a ?s))
        :effect (and (filled ?o ?w) (not
             (filledsink ?s ?w)))
    )

    (:action microwave_water
        :parameters (?a - agent ?m -
            microwave ?o - movable ?w -
            water)
        :precondition (and (inview ?a ?m
            ) (found ?a ?m) (closed ?m) (
            inside ?o ?m) (filled ?o ?w))
        :effect (and (turnedon ?m) (
            cooked ?w))
    )
```

Figure 8: Domain knowledge in PDDL format (Part 3/4).

```
    (:action openit
        :parameters (?a - agent ?o -
            object ?r - room)
        :precondition (and (inview ?a ?o
            ) (found ?a ?o) (inroom ?o ?r
            ))
        :effect (and (not (closed ?o)) (
            forall
                (?oo - object)
                (when
                    (inside ?oo ?o)
                    (inroom ?oo ?r))
            ))
    )

    (:action closeit
        :parameters (?a - agent ?o -
            object ?r - room)
        :precondition (and (inview ?a ?o
            ) (found ?a ?o) (inroom ?o ?r
            ))
        :effect (and (closed ?o) (forall
                (?oo - object)
                (when
                    (inside ?oo ?o)
                    (not (inroom ?oo ?r)
                        ))
            ))
    )

    (:action heat_food_with_oven
        :parameters (?a - agent ?v -
            oven ?f - object)
        :precondition (and (inview ?a ?v
            ) (found ?a ?v) (inside ?f ?v
            ))
        :effect (and (hot ?f) (turnedon
            ?v))
    )

    (:action cut_into_half
        :parameters (?a - agent ?k -
            carving_knife ?o - object)
        :precondition (and (inview ?a ?o
            ) (found ?a ?o) (not (
            handempty ?a)) (inhand ?a ?k)
            )
        :effect (halved ?o)
    )
)
```

Figure 9: Domain knowledge in PDDL format (Part 4/4).