"statements that are true are also known to be true" 意味着一个语句的真实性与其被知晓的真实性是相等的。如果一个语句是真的，那么它也被认为是已知的真的。这个假设是知识表达中的一种假设，称为封闭世界假设。因此，如果一个语句目前还没有被证明是真的，那么它被认为是假的。这个假设有时也被称为 " 封闭世界规则 " 。封闭世界假设的相反假设是开放世界假设，即一个语句的真实性不取决于它是否已知。

Robot Task Planning and Situation Handling in Open Worlds

Yan Ding[1], Xiaohan Zhang[1], Saeid Amiri[1], Nieqing Cao[1], Hao Yang[2], Chad Esselink[2], Shiqi Zhang[1]

***Abstract*— Automated task planning algorithms have been developed to help robots complete complex tasks that require multiple actions. Most of those algorithms have been developed for "closed worlds" assuming complete world knowledge is provided. However, the real world is generally open, and the robots frequently encounter unforeseen situations that can potentially break the planner's completeness. This paper introduces a novel algorithm (COWP) for open-world task planning and situation handling that dynamically augments the robot's action knowledge with task-oriented common sense. In particular, common sense is extracted from Large Language Models based on the current task at hand and robot skills. For systematic evaluations, we collected a dataset that includes 561 execution-time situations in a dining domain, where each situation corresponds to a state instance of a robot being potentially unable to complete a task using a solution that normally works. Experimental results show that our approach significantly outperforms competitive baselines from the literature in the success rate of service tasks. Additionally, we have demonstrated COWP using a mobile manipulator. Supplementary materials are available at: `https://cowplanning.github.io/`**

I. Introduction

Robots that operate in the real world frequently encounter complex tasks that require multiple actions. Automated task planning algorithms have been developed to help robots sequence actions to accomplish those tasks [1]. Closed world assumption (CWA) is a presumption that was developed by the knowledge representation community and states that "statements that are true are also known to be true" [2]. Most current task planners have been developed for closed worlds, assuming complete domain knowledge is provided and one can enumerate all possible world states [3]–[6]. However, the real world is "open" by nature, and unforeseen situations are common in practice [7]. As a consequence, current automated task planners tend to be fragile in open worlds rife with situations. Fig. 1 shows an example situation: *Aiming to grasp a cup for drinking water, a robot found the cup was occupied with forks and knives*. Although one can name many such situations, it is impossible to provide a complete list of them. To this end, researchers have developed open-world planning methods for robust task completions in real-world scenarios [7]–[12].

There are mainly two ways of performing open-world planning without humans in the loop in the literature. One relies on dynamically building an external knowledge base to assist a pre-defined task planner, where this knowledge base is usually constructed in an automatic way using external

[1]Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, and Shiqi Zhang are with SUNY Binghamton. {yding25, xzhan244, samiri1, ncao1, zhangs}@binghamton.edu

[2]Hao Yang and Chad Esselink are with Ford Motor Company. {hyang1, cesselin}@ford.com

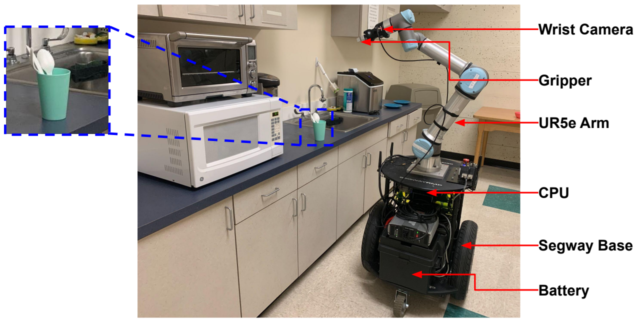


Fig. 1. An illustrative example of a *situation* in the real world, encountered during the execution of the plan "delivering a cup to a human for drinking water." The robot approached a cabinet in a kitchen room on which a cup was located. The robot then found the cup to be delivered. Before grasping it, however, the robot detected a situation that *the cup was occupied with a fork, a knife, and a spoon*. This situation prevented the robot from performing the current action (i.e., grasping) and rendered normal solutions for drinking water invalid. A mobile service robot with a UR5e arm on a Segway RMP-110 base was used for demonstrations in this work.

information [7]–[9]. Such external knowledge bases are considered bounded due to their representation and knowledge source, which limits the "openness" of their task planners. Another (more recent) way of building open-world planners is to leverage Large Language Models (LLMs) [13]. LLMs have significantly improved the performance of downstream language tasks in recent years [14]–[17]. Recent research has demonstrated that those LLMs contain a wealth of commonsense knowledge [12], [18]–[20]. While it is an intuitive idea of extracting common sense from LLMs for task planning [10]–[12], [21], there is a fundamental challenge for robots to "ground" *domain-independent* commonsense knowledge [22] to specific domains that are featured with many *domain-dependent* constraints. We propose to acquire common sense from LLMs, and aim to improve the task completion and situation handling skills of service robots.

In this paper, we develop a robot task planning framework, called *Common sense-based Open-World Planning* (**COWP**), that uses an LLM (GPT-3 in our case [14]) for dynamically augmenting automated task planners with external task-oriented common sense. COWP is based on classical planning and leverages LLMs to augment action knowledge (action preconditions and effects) for task planning and situation handling. The **main contribution** of this work is a novel integration of a pre-trained LLM with a knowledge-based task planner. Inheriting the desirable features from both sides, COWP is well grounded in specific domains while embracing commonsense solutions at large.

For systematic evaluations, we have created a dataset

考虑到正常状态下不能工作 —— 考虑随机应变能力。

with 561 execution-time situations collected from a *dining domain* [23]–[25] using a crowd-sourcing platform, where each situation corresponds to an instance of a robot not being able to perform a plan (that normally works). According to experimental results, we see COWP performed significantly better than three literature-selected baselines [6], [8], [11] in success rate. We implemented and demonstrated COWP using a mobile manipulator.

II. Background and Related Work

In this section, we first briefly discuss classical task planning methods that are mostly developed under the closed world assumption. We then summarize three families of open-world task planning methods for robots, which are grouped based on how unforeseen situations are addressed.

Classical Task Planning for Closed Worlds: Closed world assumption (CWA) indicates that an agent is provided with complete domain knowledge, and that all statements that are true are known to be true by the agent [2]. Most automated task planners have been developed under CWA [1], [5], [26]. Although robots face the real world that is open by nature, their planning systems are frequently constructed under the CWA [7], [27], [28]. The consequence is that those robot planning systems are not robust to unforeseen situations at execution time. In this paper, we aim to develop a task planner that is aware of and able to handle unforeseen situations in open-world scenarios.

Open-World Task Planning with Human in the Loop: Task planning systems have been developed to acquire knowledge via human-robot interaction to handle open-world situations [29]–[31]. For instance, researchers created a planning system that uses dialog systems to augment their knowledge bases [29], whereas Amiri et al. (2019) further modeled the noise in language understanding [30]. Tucker et al. (2020) enabled a mobile robot to ground new concepts using visual-linguistic observations, e.g., to ground the new word "box" given command of "move to the box" by exploring the environment and hypothesizing potential new objects from natural language [31]. The major difference from those open-world planning methods is that COWP does not require human involvement.

Open-World Task Planning with External Knowledge: Some existing planning systems address unforeseen situations by dynamically constructing an external knowledge base for open-world reasoning. For instance, researchers have developed object-centric planning algorithms that maintain a database about objects and introduce new object concepts and their properties (e.g., location) into their task planners [8], [9]. For example, Jiang et al. (2019) developed an object-centric, open-world planning system that dynamically introduces new object concepts through augmenting a local knowledge base with external information [8]. In the work of Hanheide et al. (2017), additional action effects and assumptive actions were modeled as an external knowledge to explain the failure of task completion and compute plans in open worlds [7]. A major difference from their methods is that COWP employs an LLM that is capable of responding to any situation, whereas the external knowledge sources of those methods limits the openness of their systems.

Open-World Task Planning with LLMs: LLMs can encode a large amount of common sense from corpus [32] and have been applied to robot systems to complete high-level tasks. Example LLMs include BERT [33], GPT-2 [17], GPT-3 [14], and OPT [15]. For example, Kant et al. (2022) developed a household robot that uses a fine-tuned LLM to reason about rearrangements of new objects [10]. Other teams used LLMs to compute plans for high-level tasks specified in natural language (e.g., "make breakfast") by sequencing actions [11], [12], [34]. Different from the above-mentioned approaches, in addition to commonsense knowledge extracted from LLMs, our system utilizes rule-based action knowledge from human experts. As a result, our planning system can be better grounded to specific domains, and is able to incorporate common sense to augment robot capabilities supported by predefined skills.

III. Algorithm

In this section, we first provide a problem statement and then present our open-world planning approach called Common sense-based Open-World Planning (COWP).

A. Problem Description

A classical task planning problem is defined by a domain description and a problem description. The domain description includes a set of actions, and action preconditions and effects. The problem description includes an initial state and goal conditions. In this paper, such a classical planning system is referred to as a **closed-world task planner**. In our case, we provide the robot with a predefined closed-world task planner (implemented using PDDL [35]), and an LLM (GPT-3 in our case). PDDL, an action-centered language, is designed to formalize Artificial Intelligence (AI) planning problems, allowing for a more direct comparison of planning algorithms and implementations [35]. A **situation** is defined as an unforeseen world state that potentially prevents an agent from completing a task using a solution that normally works. The **goal** of an open-world planner is to compute plans and handle situations towards completing service tasks or reporting "no solution" as appropriate.

B. Algorithm Description

Fig. 2 illustrates the three major components (yellow boxes) of our COWP framework. **Task Planner** is used for computing a plan under the closed-world assumption and is provided as prior knowledge in this work. **Plan Monitor** evaluates the overall feasibility of the current plan using common sense. **Knowledge Acquirer** is for acquiring common sense to augment the robot's action effects when the task planner generates no plan.

Algorithm 1 describes how the components of COWP interact with each other. Initially, Task Planner generates a satisficing plan based on the goal provided by a human user in **Line 2**. After that, the actions in the plan are performed

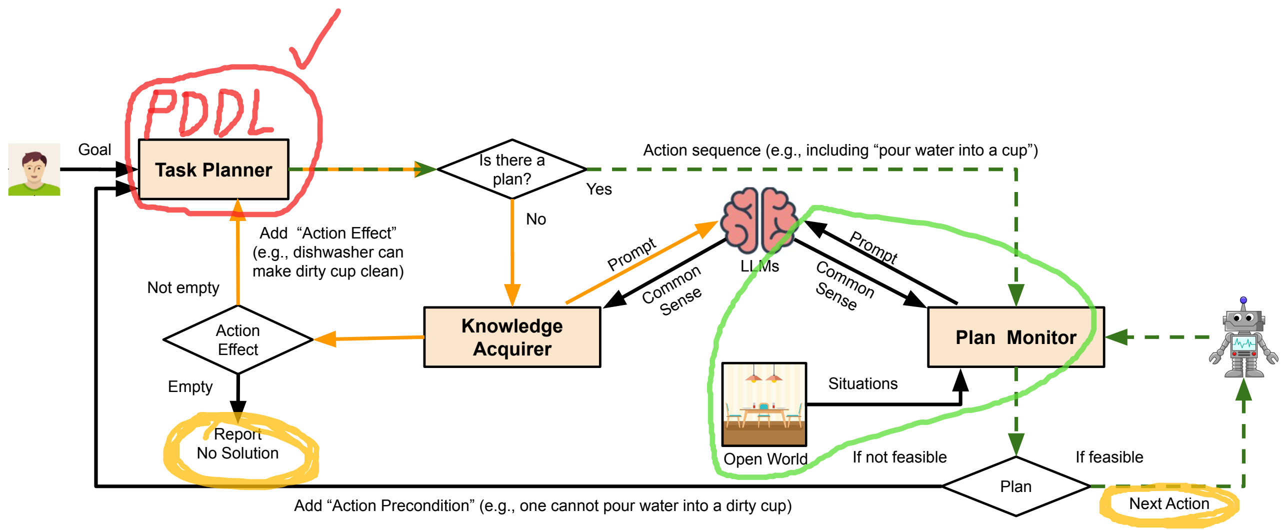


Fig. 2. An overview of **COWP** that includes the three key components of Task Planner (provided as prior knowledge under closed-world assumption), Knowledge Acquirer, and Plan Monitor. The **green** (dashed) loop represents a plan execution process where the robot does encounter no situation, or these situations have no impact on the robot's plan execution. The **orange** loop is activated when the robot's current (closed-world) task planner is unable to develop a plan, which activates Knowledge Acquirer to augment the task planner with additional action effects utilizing common sense.

Algorithm 1: COWP algorithm

```
   Require: Task Planner, Plan Monitor, Knowledge Acquirer,
            and an LLM
   Input: A domain description, and a problem description
1  while task is not completed do
2      Compute a plan $pln$ using Task Planner given the
         domain description and the problem description;
3      for each action in $pln$ do
4          Evaluate the feasibility of the current plan using
             Plan Monitor given a situation;
5          if a plan is not feasible then
6              Add an action precondition to Task Planner;
7              Compute a new task plan $pln'$, and $pln \leftarrow pln'$;
8              if $pln$ is empty then
9                  Extract common sense (action effects)
                     using Knowledge Acquirer;
10                 if action effect is not empty then
11                     Add an action effect to Task Planner;
12                     Compute a new task plan $pln''$, and
                         $pln \leftarrow pln''$;
13                 else
14                     Report "no solution";
15                 end
16             end
17         else
18             Execute the next action by the robot;
19         end
20     end
21 end
```

sequentially by a robot in the for-loop of **Lines 3-20**. If the current plan remains feasible, as evaluated by Plan Monitor, the next action will be directly passed to the robot for execution in **Line 18**; otherwise, an action precondition will be added to Task Planner in **Line 6**. For example, when Task Planner does not know anything about "dirty cups," it might wrongly believe that one can use a dirty cup as a container for drinking water. In this situation, **Line 6** will add a new statement "*The precondition of filling a cup with water is that the cup is not dirty*" into the task planner. After the domain description of Task Planner is updated (adding action preconditions or effects), the planner tries to generate a new plan pln' in **Line 7**. If no plan is generated by Task Planner (**Line 8**), Knowledge Acquirer will be activated for knowledge augmentation with external task-oriented common sense in **Line 9**. If the extracted common sense includes action effects, such information will be added into the task planner in **Line 11**. For instance, the robot might learn that a chopping board can be used for holding steak, as an action effect that was unknown before. This process continues until no additional action effects can be generated (in this case, COWP reports "no solution" in **Line 14**) or a plan is generated.

COWP leverages common sense to augment a knowledge-based task planner to address situations towards robust task completion in open worlds. The implementations of **Lines 4** and **9** using common sense are non-trivial in practice. Next, we describe how commonsense knowledge is extracted from an LLM (GPT-3 in our case) for those purposes.

C. Plan Monitor and Knowledge Acquirer

Plan Monitor is for evaluating whether there exists a situation that prevents a robot from completing its current task using a solution at hand. In particular, the input of the plan monitor includes a set of logical facts collected from the real world. The realization of our plan monitor relies on repeatedly querying GPT-3 for each action using the following prompt.

Template 1: `Is it suitable to/that [PERFORM ACTION], if there exists [SITUATION].`

One example of using the above template is "*Is it suitable that a robot pours water into a cup, if the cup is broken?*" If any action in the current plan is found infeasible, the whole plan is considered infeasible.

The objective of **Knowledge Acquirer** is to acquire common sense for augmenting the task planner's action knowledge for situation handling. The following template is for querying an LLM for acquiring common sense about action effects.

Template 2: `Is it suitable to/that [PERFORM ACTION] on/in/with/at/over [OBJECT]?`

where an example of using Template 2 is "*Is it suitable to hold steak with a chopping board?*" As a result, an additional action effect that "If a robot performs an action of unloading steak onto a chopping board, the steak will be held on the chopping board" will be added to the task planner.

Continuing the "chopping board" example, additional action effects introduced through Template 2 might enable the task planner to generate many feasible plans, e.g., using a plate (pen, or chopping board) to hold steak. It can be difficult for the task planner to evaluate which plan makes the best sense. To this end, we develop the following template (Template 3) for selecting a task plan of the highest quality amon/g those satisficing plans:

Template 3: `There are some objects, such as [OBJ-1, OBJ-2, ..., and OBJ-N]. Which is the most suitable for [CURRENT TASK]?`

where in the "chopping board" example, we expect the LLM to respond by suggesting "plate" being the most suitable for holding steak among those mentioned items.

IV. Experiments

In this section, we evaluate the performance of COWP in planning and situation handling tasks.

Experimental Setup: Our experiments were performed in a *dining domain*, where a service robot is tasked with fulfilling a user's service requests in a home setting. For simulating dining domains, we extracted six everyday tasks (e.g., serving water, and setting a dining table) and 86 objects (e.g., cup, burger, fork, table, and chair) from an existing dataset [11]. To create "situations," we randomly selected and spawned half of the available objects in each trial. A situation occurs in only one of the steps when the robot executes a task plan in each trial. In simulation experiments, we assume that our robot is provided with a perception module for converting raw sensory data into logical facts (e.g., objects and their properties), while the robot still needs to reason about the facts for planning and situation handling.

OpenAI provides several GPT-3 engines with different trade-offs between price and capability, and we used "*text-davinci-002*" – the most capable (expensive) one. We selected a "*temperature*" of 0.0 for GPT-3, which minimizes the randomness in the responses. Another important parameter is "*top_p*" for diversifying the responses, where we selected 1.0 to maximize response diversity. There are a few other parameters, such as "*maximum length*", "*presence_penalty*", and "*frequency_penalty*," where we selected 32.0, 0.0, and 0.0, respectively.

Situation Dataset for Evaluation: To evaluate the performance of COWP in dealing with situations in a dining domain, we collected a dataset of execution-time situations using Amazon Mechanical Turk. Each instance of the dataset corresponds to a situation that prevents a service robot from completing a task. We recruited MTurkers with a minimum HIT score of 70. Each MTurker was provided with a task description, including steps for completing the task. The MTurkers were asked to respond to a questionnaire by identifying one step in the provided plan and describing a situation that might occur in that step. For instance, we provided a plan of *1) Walk to dining room, 2) Find a sink, 3) Find a faucet, 4) Turn on faucet, 5) Find a cup...*, which was associated to the task of *Drinking water*. Two common responses from the MTurkers were "*Faucet has no water*" for Step 4 and "*Cup is broken*" for Step 5. We received 690 responses from MTurkers, and 561 of them were valid, where the responses were evaluated through both a validation question and a manual filtering process. For instance, some MTurkers copied and pasted text that was completely irrelevant, and those responses were removed manually. There are at least 92 situations collected for each of the six tasks. Note that some situations are similar enough to be grouped together, e.g., "cup is broken" and "cup is shattered" are considered indistinguishable. As a result, there are 16-27 distinguishable situations for each task. We constructed a situation simulator by randomly sampling indistinguishable situations from our collected dataset for the evaluation purpose in this paper.

Baselines and Evaluation Metrics: The evaluation of open-world planners is based on the *success rate* of a robot completing service tasks in a dining domain. The following three baselines have been used in our experiments:

- Closed World (CW) corresponds to classical task planning developed for closed-world scenarios. In practice, CW was implemented by repeatedly activating the closed-world task planner and updating the current world state after executing each action.
- External Knowledge (EK) [8] is a baseline approach that enables a closed-world task planner to acquire knowledge from an external source. In our implementation, this external source provides information about a half of the domain objects.
- Language Model (LM) [11] is a baseline that leverages GPT-3 to compute task plans, where domain-specific action knowledge is not utilized. The authentic system of the LM baseline [11] was not grounded. As a result, their generated plans frequently involve objects unavailable or inapplicable in the current environment. To make a meaningful comparison, we call LM for $N \leq 100$ times until it generated a plan that involves only those objects that are interactable by the robot.

Success Criteria: Not all "solutions" from an open-world task planner are considered correct. For instance, GPT-3 suggests that one can use a pan for drinking water, which is technically possible, but uncommon in our everyday life. LLM-based task planners might wrongly believe that a good-quality plan is generated, while it is actually not the case. To compare with the ground truth, we recruited a second group of people, including six volunteers, to evaluate the performance of different open-world task planners. The volunteers included two females and four males aged 20-40,

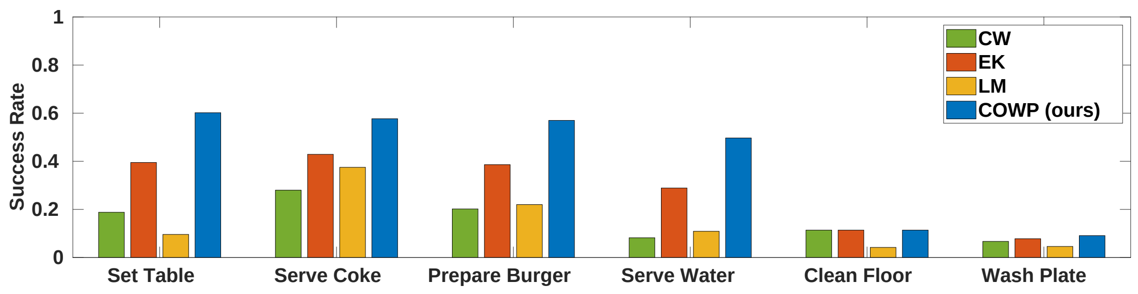


Fig. 3. Overall performances of COWP (ours) and three baseline methods under **six different tasks**, where the *x-axis* represents the task. Each success rate value is an average of 150 trials. The tasks are ranked based on the performance of COWP, where the very left corresponds to its best performance.

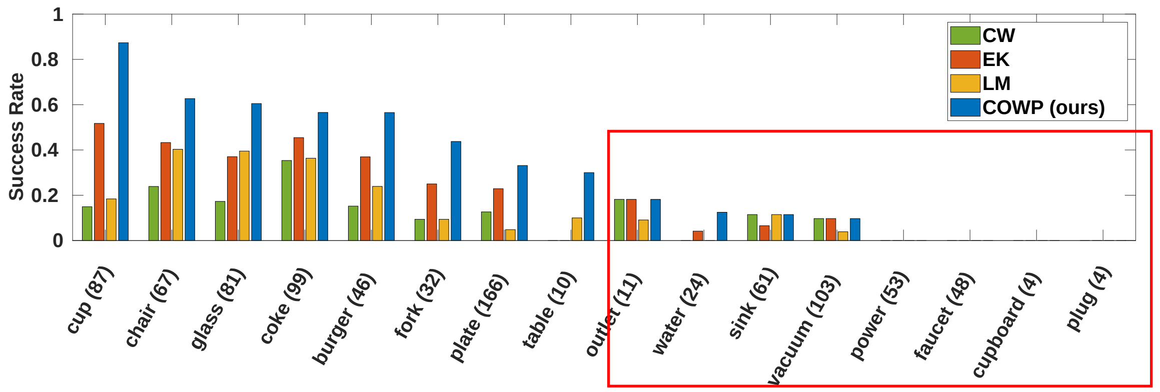


Fig. 4. Overall performances of COWP (ours) and three baseline methods under **different objects**, where the *x-axis* represents the object involved in the situation, the number beside each object is the occurrence of the object in our situation dataset, and the *y-axis* represents the success rate. The objects are ranked based on the performance of COWP, where the very left corresponds to its best performance.

and all were graduate students in engineering fields.

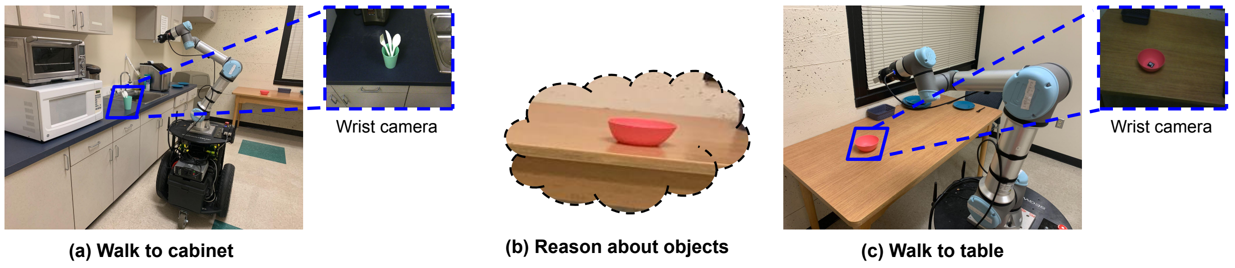
COWP vs. Three Baselines by Task: Fig. 3 shows the results of comparing COWP and three baselines under six tasks in success rate. In all six tasks, COWP outperformed the three baselines. CW and LM produced the lowest success rates in most tasks. We believe the poor performance of CW is caused by the incapability of leveraging external information for handling unforeseen situations. For LM, its poor performance is caused by its weakness in grounding general commonsense knowledge in specific domains. As a result, many "solutions" generated by LM are not executable by the robot.

It is quite interesting to see that open-world planners (including COWP) work better in some tasks than the others. For instance, in the "set table" task, there are many "missing object" situations, and it happened that the robot could easily find alternative tableware objects to address those situations under the assistance of GPT-3. Some tasks such as "Clean Floor" are more difficult because many situations are beyond the robot's capabilities, e.g., power outage and broken plugs.

COWP vs. Three Baselines by Object: Fig. 4 compares the performance of COWP and three baselines in success rate under *different objects*. The *x-axis* represents the objects in our situation dataset and their occurrences, and the *y-axis* represents the performance of four planning methods. For instance, common situations about cup include "dirty cup," "broken cup," and "missing cup." COWP (ours) performed the best over all objects, while some baselines produced comparable performances over some objects. An important observation is that COWP produced higher success rates in situations involving "cup," "coke," "glass," and "chair." This is because many of those relevant situations are about missing objects, and the robot can easily find their alternatives in dining domains. By comparison, those situations involving "power", "faucet", "cupboard", and "plug" are more difficult to the four methods (including COWP), because addressing those situations frequently require skills beyond the robot's capabilities.

Robot Demonstration: We demonstrated COWP using a mobile service robot that was tasked with delivering a cup for drinking water. Our robot is equipped with a UR5e arm, and is based on Segway RMP-110. The robot uses a Robotiq Wrist Camera for object detection, and is capable of performing basic navigation and manipulation behaviors. For visual scene analysis, the robot detects objects, and then based on their geometric relationships estimates spatial relationships between objects. While there exist more powerful tools for visual scene analysis, computer vision is not our focus, and our goal here is to implement a basic perception component to close the perceive-reason-act loop.

Fig. 5 shows a real-world demonstration where a robot used COWP for planning to complete the service task of "Deliver a cup for drinking water." The initial plan included



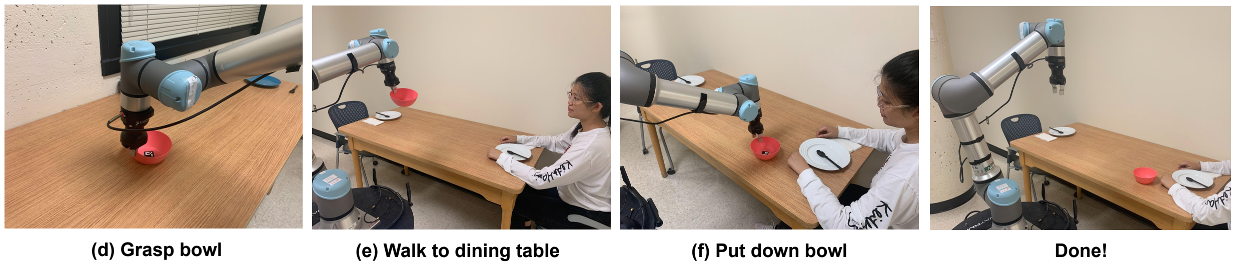


Fig. 5. An illustrative example of COWP for open-world planning, where the robot was tasked with "delivering a cup for drinking water." **(a)** The robot walked to a cabinet, and located a cup on the cabinet. However, the robot found a situation that there were objects in the cup (a knife, a fork, and a spoon in this case). This observation was entered into the plan monitor, which queried GPT-3, and suggested that the planned action "grasp" was not applicable given the occupied cup. Accordingly, COWP updated its task planner by adding the new information that one cannot pour water into a non-empty cup. **(b)** The robot reasoned about other objects that were available in the environment, and queried GPT-3 to update the task planner about whether those objects can be used for drinking water – details in Section III. It happened that the robot learned a bowl could be used for drinking water. **(c)** A new plan of delivering a bowl to the human for drinking water was generated. Following the new plan, the robot walked to the table on which a bowl was located. **(d)** The robot grasped the bowl after observing it using vision. **(e)** The robot navigated to the dining table with the bowl. **(f)** The robot put down the bowl onto the dining table, and explained that a bowl was served due to the cup being occupied, which concluded the planning and execution processes.

the following actions for the robot:

1) Walk to a cabinet on which a cup is located,
2) Grasp the cup after locating it,
3) Walk to dining table, and
4) Put down the cup to a table where the human is seated.

However, a situation was observed after executing Action #1, and the robot found *the cup is occupied*. The robot initially did not know whether this affects its plan execution. After querying GPT-3, the robot learned that one cannot pour water into an occupied cup, which renders its current plan infeasible. COWP enabled the robot to reason about other objects of the environment. The robot iteratively queried GPT-3 about whether X can be used for drinking water (using Prompt Template 2 in Section III), where X is one of the objects from the environment. It happened that the robot learned "bowl" can be used for drinking water. With this newly learned, task-oriented commonsense knowledge, COWP successfully helped the robot generate a new plan, which used the bowl instead, to fulfill the service request.

We have generated a demo video that has been uploaded as part of the supplementary materials.

V. Conclusion and Future Work

In this paper, we develop a Large Language Model-based open-world task planning system for robots, called COWP, towards robust task planning and situation handling in open worlds. The novelty of COWP points to the integration of a classical, knowledge-based task planning system, and a pretrained language model for commonsense knowledge acquisition. The marriage of the two enables COWP to ground domain-independent commonsense knowledge to specific task planning problems. To evaluate COWP systematically, we collected a situation dataset that includes 561 execution-time situations in a dining domain. Experimental results suggest that COWP performed better than existing task planners developed for closed-world and open-world scenarios. We also provided a demonstration of COWP using a mobile manipulator working on delivery tasks, which provides a reference to COWP practitioners for real-world applications.

It is evident that prompt design significantly affects the performance of Large Language Models (LLMs) [36], which has motivated the recent research on prompt engineering. For instance, we tried replacing "suitable" with "possible" and "recommended," in the prompt templates and observed a decline in the system performance. Researchers can improve the prompt design of COWP in future work. There is the potential that other LLMs (other than GPT-3) can produce better performance in open-world planning, and their performances might be domain-dependent, which can lead to very interesting future research. We present a complete implementation of COWP on a real robot, while acknowledging that there are many ways to improve the implementations. For instance, one can use a more advanced visual scene analysis tool to generate more informative observations for situation detection, or equip the robot with more skills (such as wiping a table, moving a chair, and opening a door) to deal with situations that cannot be handled now.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting*. Cambridge University Press, 2016.

[2] R. Reiter, "On closed world data bases," in *Readings in artificial intelligence*. Elsevier, 1981, pp. 119–140.

[3] C. A. Knoblock, J. D. Tenenberg, and Q. Yang, "Characterizing abstraction hierarchies for planning," in *Proceedings of the ninth National conference on Artificial intelligence-Volume 2*, 1991, pp. 692–697.

[4] J. Hoffmann, "Ff: The fast-forward planning system," *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.

[5] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "Shop2: An htn planning system," *Journal of artificial intelligence research*, vol. 20, pp. 379–404, 2003.

[6] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[7] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek *et al.*, "Robot task planning and explanation in open and uncertain worlds," *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[8] Y. Jiang, N. Walker, J. Hart, and P. Stone, "Open-world reasoning for service robots," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 725–733.

[9] S. Chernova, V. Chu, A. Daruna, H. Garrison, M. Hahn, P. Khante, W. Liu, and A. Thomaz, "Situated bayesian reasoning framework for robots operating in diverse everyday environments," in *Robotics Research*. Springer, 2020, pp. 353–369.

[10] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," *arXiv preprint arXiv:2205.10712*, 2022.

[11] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *Thirty-ninth International Conference on Machine Learning*, 2022.

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

[13] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

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

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

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[18] C. Wang, P. Liu, and Y. Zhang, "Can generative pre-trained language models serve as knowledge bases for closed-book qa?" in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 3241–3251.

[19] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar, J. Andreas, I. Mordatch, A. Torralba, and Y. Zhu, "Pre-trained language models for interactive decision-making," *Advances in Neural Information Processing Systems*, 2003.

[20] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi, "Symbolic knowledge distillation: from general language models to commonsense models," *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

[21] D. Elsweiler, H. Hauptmann, and C. Trattner, "Food recommender systems," in *Recommender Systems Handbook*. Springer, 2022, pp. 871–925.

[22] E. Davis and G. Marcus, "Commonsense reasoning and commonsense knowledge in artificial intelligence," *Communications of the ACM*, vol. 58, no. 9, pp. 92–103, 2015.

[23] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5678–5684.

[24] E. K. Gordon, S. Roychowdhury, T. Bhattacharjee, K. Jamieson, and S. S. Srinivasa, "Leveraging post hoc context for faster learning in bandit settings with applications in robot-assisted feeding," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 528–10 535.

[25] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[26] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise, "An introduction to the planning domain definition language," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 2, pp. 1–187, 2019.

[27] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 3, pp. 363–373, 2019.

[28] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and autonomous systems*, vol. 56, no. 11, pp. 955–966, 2008.

[29] V. Perera, R. Soetens, T. Kollar, M. Samadi, Y. Sun, D. Nardi, R. Van de Molengraft, and M. Veloso, "Learning task knowledge from dialog and web access," *Robotics*, vol. 4, no. 2, pp. 223–252, 2015.

[30] S. Amiri, S. Bajracharya, C. Goktolgal, J. Thomason, and S. Zhang, "Augmenting knowledge through statistical, goal-oriented human-robot dialog," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 744–750.

[31] M. Tucker, D. Aksaray, R. Paul, G. J. Stein, and N. Roy, "Learning unknown groundings for natural language interaction with mobile robots," in *Robotics Research*. Springer, 2020, pp. 317–333.

[32] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

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

[34] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *arXiv preprint arXiv:2207.05608*, 2022.

[35] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, "Pddl— the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[36] N. Zhang, L. Li, X. Chen, S. Deng, Z. Bi, C. Tan, F. Huang, and H. Chen, "Differentiable prompt makes pre-trained language models better few-shot learners," in *International Conference on Learning Representations*, 2021.