IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Text2Reaction : Enabling Reactive Task Planning Using Large Language Models

Zejun Yang[1,2], Li Ning[1,2], Haitao Wang[1,3], Tianyu Jiang[1,3], Shaolin Zhang[1,3], Shaowei Cui[1,3*], Hao Jiang[1,2*], Chunpeng Li[1,2], Shuo Wang[1,3], Zhaoqi Wang[1,2]

***Abstract*—To complete tasks in dynamic environments, robots need to timely update their plans to react to environment changes. Traditional stripe-like or learning-based planners struggle to achieve this due to their high reliance on meticulously predefined planning rules or labeled data. Fortunately, recent works find that Large Language Models (LLMs) can be effectively prompted to solve planning problems. Thus, we investigate the strategies for LLMs to master reactive planning problems without complex definitions and extra training. We propose Text2Reaction, an LLM-based framework enabling robots to continuously reason and update plans according to the latest environment changes. Inspired from human's step-by-step re-planning process, we present the Re-planning Prompt, which informs LLMs the basic principles of re-planning and fosters the gradual development of a current plan to a new one in a three-hop reasoning manner - cause analysis, consequence inference, and plan adjustment. Additionally, Text2Reaction is designed to first generate an initial plan based on the task description before execution, allowing for subsequent iterative updates of this plan. We demonstrate the superior performance of Text2Reaction over prior works in reacting to various environment changes and completing varied tasks. Additionally, we validate the reliability of our re-planning prompt through ablation experiments and its capability when deployed in real-world robots, enabling continuous reasoning in the face of diverse changes until the user instructions are successfully completed.**

***Index Terms*—Planning under Uncertainty, AI-Based Methods, Learning from Demonstration**

I. Introduction

ROBOT task planning studies how a robot generates an action sequence to accomplish a task [10], [17], and many representative methods are proposed for this problem [23], [24]. However, real-world environments are quite variable since they are often changed by humans or agents, each impacting the robot differently. Some changes have no effect on robot's execution (open a microwave when the robot opens a fridge), some have facilitative impacts (open the fridge when the robot is about to do so), and others can obstruct the execution (close the fridge before the robot prepares to place an item in). Robots need to update their plans in a goal-oriented manner to adapt to these changes. Thus, reactive planning methods [12], [19] are required to comprehend various environment changes that may hinder robots' progress, react to them in real-time, and reason on how to update plans to ensure the achievement of task goals.

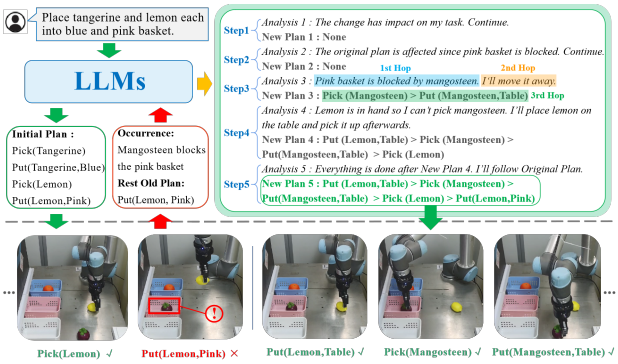

Fig. 1: Text2Reaction uses LLMs to generate initial plans of language instructions and to update them through step-by-step three-hop reasoning based on sources of feedback.

Modeling re-planning logic is the key for the reactive task planning problem. Strip-like methods [6] explicitly define logic rules such as action postconditions and re-plan through deterministic algorithms based on the rules. However, they lack flexibility in adapting to vast variability of environment changes under complex scenarios. Recently, significant advances have been made by learning-based methods [27] which use machine learning techniques to model re-planning process without the need to define complex rules. However, most of these methods highly depend on large-scale datasets and carefully designed training strategies.

Fortunately, Large Language Models (LLMs) [4], [34] offer new solutions to this problem. Given that LLMs have learned rich semantic knowledge of the world, they can be naturally used for everyday planning without complex definitions or additional training. Some recent studies have attempted to provide various types of feedback to LLMs and prompt them to handle execution failures, and have achieved promising results [15], [28]. However, these methods either generate and execute fixed plans, predict erroneous actions involving non-existent objects, or overlook other crucial environment feedback aside from actions' preconditions.

In this paper, we study an LLM-based framework capable of comprehensively analyzing various feedback and continuously

Manuscript received: September, 28, 2023; Revised January, 2, 2024; Accepted February, 15, 2024.

This paper was recommended for publication by Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by National Key Research and Development Program of China under Grant No.2018AAA0103002 and Innovation Research Program of ICT under Grant No.E261070.

[1]University of Chinese Academy of Sciences
[2]Institute of Computing Technology Chinese Academy of Sciences
[3]Institute of Automation Chinese Academy of Sciences
*Corresponding-author: shaowei.cui@ia.ac.cn, jianghao@ict.ac.cn

Digital Object Identifier (DOI): see top of this page.

re-planning in response to environment changes.We mimic human's step-by-step analyzing when facing changes and propose the Re-planning Prompt, designed to guide LLMs in updating plans through chained reasoning. The re-planning prompt first offers a concise overview of chained thinking patterns, and then formalizes the environment setup and the robot's skills. It also provides structured case demonstrations which show how LLMs reason over various feedback such as the current plan, to achieve task goals. It fosters the gradual development of a current plan to a new one and promotes procedural reasoning at each step in a three-hop cycle - cause analysis, consequence inference, and plan adjustment. Considering that each re-planning requires a current plan as the input, we propose that initializing a plan based on the task description in advance can facilitate its iterative updates. Text2Reaction integrates all these components (Fig.1), enabling complex feedback-action reasoning in highly dynamic environments.

Comparison experiments show that Text2Reaction achieves a generally higher success rate than prior methods especially when adapting to complex environment changes and accomplishing long-horizon tasks. Remarkably, we find that it can efficiently react to various types of human-induced interference, including non-impactful, facilitative and obstructive ones. Ablation studies show that Text2Reaction, offered with only a few cases for learning, has notable re-planning performance even for unseen tasks, when using different pre-trained language models. Real-world experiments highlight Text2Reaction's notable transferability in adapting to unseen changes without predefined rules and additional training. Demonstrations of real-world implementations further showcase its more surprising abilities including continuous adaptation to multiple interferences and the completion of tasks that require complex logic reasoning.

II. Related Work

Task and Motion Planning. Task and motion planning is a problem that studies how robots decompose high-level tasks and generate executable action sequences [10], [17]. Strip-like methods [7], [11] explicitly model the planning rules. Machine learning particularly deep learning, is widely applied [16], [29] to implicitly map tasks to actions without defining complex rules. And many of these works use language models as the backbones for instructions understanding, grounding and planning [2], [26], [33].

Our work focuses on reactive task planning that requires robots to face various environment changes. Prior strip-like and learning-based works [3], [6], [25], [27], [35] exhibit obvious shortcomings requiring either lots of predefined planning rules, or high-quality training data. Thus, we use LLMs for reactive planning without complex modeling of re-planning rules and additional training.

LLMs for Task Planning Large language models (LLMs) are large-scale neural networks with billions of parameters and are pretrained in an unsupervised manner with numerous corpora. Besides classic natural language processing problems, recent works have discovered that LLMs can also be used in robot task planning through prompting [1], [15], [21], [28], [32]. Typical methods using LLMs can be categorized into **Scorer** and **Generator**. In LLMs-as-Scorer methods, LLMs are set to score an enumeration of actions based on common sense, together with other modules independently scoring from different factors, such as affordance [1], [5], geometric dependency [21], safety and preference [14]. All scoring results collectively influence planning. In LLMs-as-generator methods, LLMs are set to generate formatted texts based on task descriptions which need to be parsed into PDDL domains [37], executable actions [8] or both [31]. These methods explore how LLMs plan to complete instructions in certain environments but lack mechanisms for adapting to unpredictable circumstances.

LLMs for Reactive Planning. Several recent works use LLMs in handling execution failures [20], [39]. *InnerMono* [15] presents an agent-environment closed loop framework via inner monologue, where several sources of textual feedback are provided for LLMs to predict next actions. However, it sometimes ignores environment feedback, leading to erroneous actions involving non-existent objects. *Progprompt* [32] represents robot plans as pythonic functions and utilizes assertion checking to provide an environment feedback mechanism. However, since the planning functions are fixed during execution, the robot struggles to respond to unforeseen environment changes that are not predefined in the assertions. *CAPE* [28] artificially defines a precondition-error-type table for categorizing all execution failures caused by mismatches in current actions' preconditions. When an error occurs, *CAPE* first retrieves error type from table and then informs LLMs of the error to query for corrective next actions. However, this method focuses on whether the preconditions of actions are met, but overlooks other important environment feedback that is not directly related to current action's execution. Consider a robot which is asked to place a clean apple in a fridge. If the clean apple gets dirty again while the robot is about to place this apple in the fridge, *CAPE* will ignore the cleanliness of the apple and continue its current action.

In this work, Text2Reaction first generates a task plan before acting, and then fully assesses the impacts of environment changes on the task and updates the plan globally while interacting with the scene.By using our re-planning prompts, LLMs can incrementally learn to analyze the impacts and develop new plans step-by-step, enabling Text2Reaction to tackle complex reactive planning challenges.

III. Text2Reaction

A. Problem Formulation

In this work, a robot adapts to constant environment changes through multiple re-planning periods. During the t-th period, the LLM-based planner receives four types of textual feedback - the history sequence of executed actions H_t, the current sequence of unexecuted actions P_t, the observed environment change E_t and the task description T. It then generates a new plan P_{t+1} under the planning policy π for the next period $t+1$. This re-planning process can be formulated as $P_{t+1} = \pi\,(\,H_t,\,P_t,\,E_t,\,T)$.

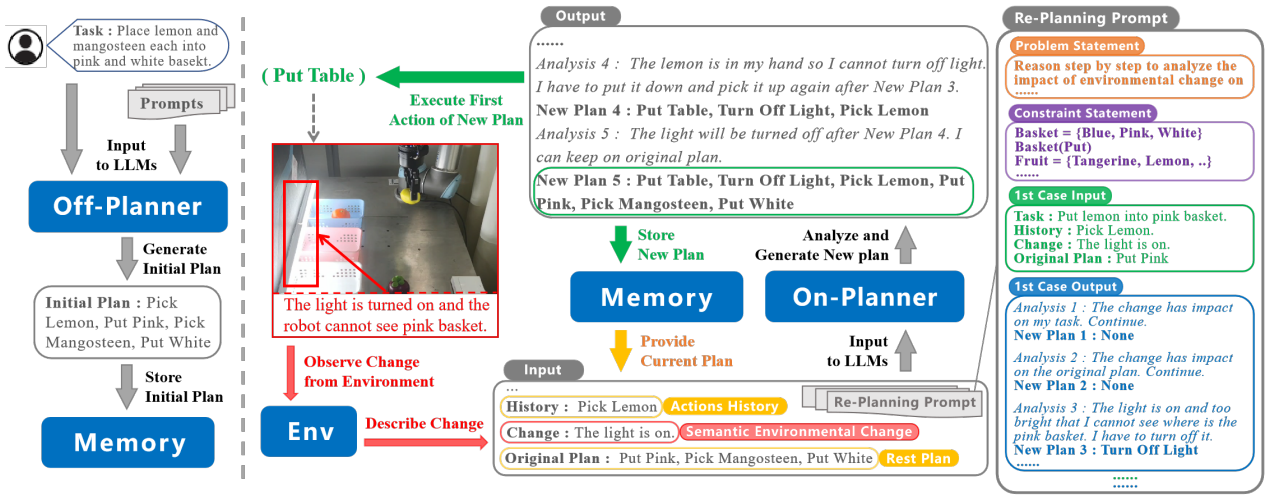

Fig. 2: Overview of Text2Reaction. Upon receiving a new user instruction, Off-Planner first generates an action sequence in the offline phase. Then in the online closed-loop phase, if Env observes an environment change, Memory supplies necessary task-related information to On-Planner. After in-context learning guided by the re-planning prompt, On-Planner can globally analyze and update the plan step-by-step to adapt to the change. Finally, this newly generated plan is stored in Memory and executed.

B. Structure Overview of Text2Reaction

The overview of Text2Reaction is shown in Fig. 2. Off-Planner is an LLM-based planner that generates initial plans, prompted by prior methods [13], [32]. On-Planner is another planner, which updates plans under the guidance of the re-planning prompts (see Section III-C). The Memory module continuously stores the newest plans. Env is designed to map world states into semantic descriptions, closing the agent-environment loop. This can be implemented using large-scale pretrained visual-language models (VLMs) [9], [18] or grounding neural networks [22].

After receiving a new task, Text2Reaction completes it through two phases. Before execution, Off-Planner generates an action sequence based only on the task description and sends it to Memory. When the environment changes during the robot's execution, On-Planner will receive various textual feedback about environment states and instruction executions. After being prompted by the re-planning prompt, On-Planner reasons over the feedback and gradually formulates a new manipulation sequence. Finally, the new plan replaces the current plan in Memory and will be executed until completion.

C. Informing LLMs of Re-Planning Principles

We propose the Re-Planning Prompt to guide LLMs in mapping the current plan P_t to a new plan P_{t+1} during each re-planning process. The structure of the re-planning prompts can be represented by a triple $\mathcal{L} = (\mathcal{P}, \mathcal{C}, \mathcal{D})$. It conveys re-planning principles to LLMs by providing not only a summarized introduction through the Problem Statement $\mathcal{P}$ and the Constraint Statement $\mathcal{C}$ but also specific details through Case Demonstrations $\mathcal{D}$ (see Fig. 2).

Problem Statement concisely conveys three key points to LLMs. First, LLMs are tasked with learning to tackle reactive planning challenges. Secondly, when addressing distinct re-planning issues, it necessitates multi-step thinking, where in each step LLMs are required to engage in reasoning based on the principles of cause analysis, consequence inference and plan adjustment. Lastly, LLMs are obligated to produce re-planning content in adherence to a specified format.

Constraint Statement uses two types of formulaic declarations to display all interactive objects in the environment and the robot's executable operations on them. Object declaration (`Z={A,B,...}`) first gathers objects such as `A` and `B`, which have similar superordinate categories and support the same robot operations, into a group. It then assigns a new name `Z` to denote this group. Operation declaration (`Z(O1,...)`) associates the group of objects `Z` with a shared set of operations such as `O1`. For instance, `Fruit={Apple,Pear}` declares `Fruit` for `Apple` and `Pear` since they belong to fruit in reality. `Fruit(Pick,Slice)` indicates that every object in the `Fruit` class can be picked and sliced by robots.

Case Demonstrations illustrate the details of re-planning. Upon detecting an environment change, On-Planner receives four types of feedback ordered as user instruction (*Task*), a sequence of executed actions (*Process*), semantic description of the environment change (*Change*), and a sequence of actions yet to be completed (*Original Plan*). Additionally, On-Planner is required to adhere to the reasoning principle and the output format involved in the demonstrations for re-planning.

D. Multi-Step Three-Hop Reasoning

Chain-of-Thought has shown that LLMs perform better when they reason step-by-step rather than directly providing final answers [36]. Inspired by this, our re-planning prompt requires On-Planner's built-in LLMs to analyze feedback through multiple steps and revise the plan iteratively instead of producing a final new plan at once (Fig. 3).

The process of updating a current plan through n iterative steps can be represented as $P_t = P_t^0 \rightarrow P_t^1 \rightarrow \ldots \rightarrow P_t^n = P_{t+1}$. This chained re-planning adheres to some basic principles. In Step 1 and Step 2, assess whether the environment change has impacts on the overall task and on

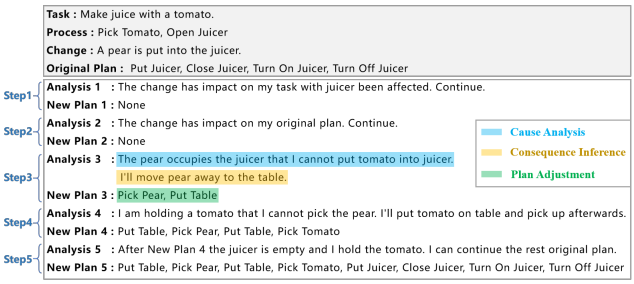

Fig. 3: Multi-step three-hop Reasoning. LLMs are required to reason in chains over various textual feedback and adopt three-hop thinking cycle in each step, including causal analysis, consequence inference and plan adjustment (taking Step 3 as example). ***Note:*** *Under the AI2THOR simulation platform setup, a single manipulator is permitted to execute 'Pick' followed by 'Open'.*

the current plan. In Step 3, determine the necessary actions of the new plan based on environment change. In subsequent steps, consider all types of feedback in various combinations and iteratively adjust the last step's action sequence. If an obvious error is found in the provided Original Plan, make corrections on the spot.

Re-planning prompt also requires LLMs to adopt procedural reasoning in each step. This includes the first two hops of the analytical process and a third hop for adjusting P_t^{i-1} to P_t^i. Hop 1 is the cause analysis, formulated as $Cause = f_{cause}(Feedback, P_t^{i-1})$. LLMs check whether the action sequence from last step P_t^{i-1} can serve as the final plan P_{t+1}. If not, they find the reasons and identify any unconsidered sources of feedback. Hop 2 is the consequence inference, formulated as $Consequence = f_{conse}(Feedback, Cause)$. LLMs determine the necessary modifications to P_t^{i-1} based on the causes from 1st hop. Hop 3 is the plan adjustment, formulated as $P_t^i = f_{adjust}(Consequence, P_t^{i-1})$. LLMs generate P_t^i from P_t^{i-1} according to the consequence inferred from 2nd hop, and output this adjusted sequence as P_{t+1} if all sources of feedback are considered comprehensively.

IV. Experimental Results

We compare Text2Reaction with prior LLM-based methods to evaluate its capability in reactive planning (Sec.IV-A). We conduct a threefold evaluation of Text2Reaction. We have ablation experiments on our re-planning prompt to evaluate its reliability as guidance (Sec.IV-B). Finally, we implement Text2Reaction in two real-world scenes (Sec.IV-C).

A. Text2Reaction Comparison Experiment

We consider three previously proposed LLM-based methods capable of handling reactive planning tasks for comparison and we make adaptations in these methods to our experiment. *CAPE* checks and deals with predefined precondition errors [28]. *InnerMono* provides various types of feedback for LLMs to address execution failures [15]. *ProgPrompt* generates pythonic plan functions with assertions [32]. We also construct two variants of Text2Reaction for further evaluation of its planning mechanism. *Ours w/o Online* only uses Off-Planner to generate initial plans without online planning. *Ours Err Offline* receives initial plans from the ground truth with a random error action instead of from Off-Planner. All methods use GPT-4 for planning.

We consider sampling various tasks from ALFRED [30] and construct our re-planning task family for evaluation. ALFRED is a widely used benchmark for studying how domestic robots learn a mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. ALFRED provides 2,685 specific tasks categorized in 7 task types including *Pick&Place*, *Pick2&Place*, *Stack&Place*, *Clean&Place*, *Heat&Place*, *Cool&Place*, and *Examine in Light*. It describes each task with different natural language directives, generating over 25,000 directives in total. ALFRED is built on AI2THOR simulation platform, covering 4 types of single-room household scenes including bedrooms, kitchens, bathrooms, living rooms, and involving 108 interactive objects. Additionally, it supports 7 types of interactive actions for single manipulator including *Pick*, *Put*, *Open*, *Close*, *Toggle On*, *Toggle Off* and *Slice*. Notably, *Pick* then *Open* is feasible under AI2THOR settings.

In order to simulate a robot's reactive planning in a real home environment, we create 20 tasks, each with their respective language directives, from various scenarios within ALFRED to test these methods. We introduced three different types of human-induced interferences at appropriate points during the robot's execution process, ensuring that these interferences could maximally affect modifications on the robot's plan. *None-Impactful Interference* has no effect on the directive's execution, *Facilitative Interference* inadvertently assists in accomplishing some task goals, *Obstructive Interference* hinders the robot's progress by preventing it from completing tasks as originally planned. Based on the aforementioned setup, we construct three variants for each task, resulting in a total of 60 task variants for our evaluation experiments. In Table II we display three tasks with three human interferences introduced in each task.

We employ 3 metrics to evaluate a method's performance on completing all tasks under each type of interference. *Executability Rate (ER)* is the fraction of tasks where the method always generates executable actions. *Success Rate (SR)* is the fraction of successfully completed tasks. Given that there is more than one way to complete a task and methods are encouraged to complete with fewer actions, we further evaluate with *Success weighted by Path Length (SPL)* which is calculated as the weighted average of *SR* over n tasks:

$$SPL = \frac{1}{n}\sum_{i=1}^{n} w_i \times SR_i = \frac{1}{n}\sum_{i=1}^{n} \frac{L_{gt\_i}}{\max(L_{gt\_i}, L_i)} \times SR_i,$$

where $n = 20$ in this experiment, L_i denotes the number of actions executed by method in i-th task, and L_{gt_i} indicates the number of actions from ground truth.

From Table I, Text2Reaction generally outperforms other methods, achieving generally the highest success rate in generating executable actions of at least 95%, and reacting to various environment changes with up to an 85% success rate. By comparing our methods with others, we can draw the

TABLE I: Comparison results of Text2Reaction with its 2 variants and 3 prior proposed methods. Text2Reaction generally outperforms other methods, doing a great job at generating executable plans and in reacting to various environment changes, due to the contributions of re-planning prompts.

Method	*None-Impactful Interference*			*Facilitative Interference*			*Obstructive Inference*		
	ER	SR	SPL	ER	SR	SPL	ER	SR	SPL
InnerMono [15]	0.90	0.85	0.82	0.90	0.85	0.77	0.85	0.20	0.19
ProgPrompt [32]	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.91	**1.00**	0.15	0.15
CAPE [28]	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.94	**1.00**	0.45	0.39
Ours w/o Online	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.89	**1.00**	0.00	0.00
Ours Err Offline	0.95	0.90	0.90	**1.00**	0.80	0.80	0.95	0.75	0.72
Ours	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.95	**0.85**	**0.82**

following analyses and key conclusions. (1) *InnerMono* may fail to complete tasks even under *None-Impactful Interference* due to the significant instability in its planning process. Besides, it can only handle a small portion of the effects caused by *Obstructive Interferences*. (2) *Progprompt* does well in generating executable actions but struggles to complete tasks under *Obstructive Interference* because its fixed plans can only address a limited number of anticipated uncertainties through assertions. (3) *CAPE* shows marked improvement in SR and SPL under *Obstructive Interference* compared to the previous two methods due to its error detection and correction mechanisms. However, this improvement is still limited as it cannot take measures against disturbances that do not trigger errors in the preconditions of actions. (4) *Ours w/o Online* cannot address any obstructive environment change, which shows that online planning plays a crucial role in solving reactive planning problems. (5) *Ours Err Offline* performs online corrections to initial plans with erroneous actions, leading to a substantial increase in SR from 0% to 75%. This indicates that our method is effective in correcting imperfect plans, thereby ensuring successful completions of tasks.

TABLE II: Examples of Text2Reaction successfully completing tasks with different number of required actions $\|A\|$ and addressing environment changes which results in varied increase number of extra needed actions $\Delta\|A\|$.

Task	$\|A\|$	*Interference*	$\Delta\|A\|$
Put the bowl with pencil on the tv stand	4	open drawer move bowl to tv stand block bowl with apple	+0 -2 +4
Put washed lettuce in the refrigerator.	13	move apple into fridge open fridge make lettuce dirty again	+0 -1 +5
Put a heated apple slice in the cabinet.	17	throw ladle into bin cut apple into pieces cool heated apple piece	+0 -1 +10

Table II illustrates examples of tasks successfully completed by TextReaction, covering three levels of task complexity from short to long horizon as well as three different types of interference. We show that Text2Reaction can adeptly adapt to various types of changes resulting from different interferences. It preserves the original plan under *None-Impactful Interference*, modifies the plan by removing certain parts in the case of *Facilitative Interference* and strategically expands the plan under *Obstructive Interference*. Furthermore, Text2Reaction successfully planned a total of 27 actions while completing the third task under obstructive interference, demonstrating its capability for long-term planning and replanning. In the second task listed in Table II, we make the washed lettuce dirty just as it is about to be placed in the fridge. Under this circumstance, none of the three prior methods could successfully complete the task. *InnerMono* does not re-plan as it overlooks the crucial feedback regarding the cleanliness of the lettuce. *Progprompt* fails to anticipate that the lettuce would become dirty upon refrigeration, thereby generating a fixed plan without assertions for this unforeseen event. *CAPE* also fails as its focus is strictly on the execution of actions but the lettuce being dirty does not conflict with the action preconditions. In contrast, our method can fully consider the impact of each feedback on the plan and update the plan from a global view of task completion.

B. Re-Planning Prompt Ablation Experiment

In order to assess the effectiveness of each component in our re-planning prompt and the multi-step three-hop reasoning mechanism in enhancing LLMs' re-planning capabilities, we evaluate our prompting method with its eight ablations - without Problem Statement (w/o $\mathcal{P}$), without Constraint Statement (w/o $\mathcal{C}$), without Case Demonstrations (w/o $\mathcal{D}$), without Consequence Inference (w/o *CI*), without Plan Adjustment except in the final step (w/o *PA*), without both Cause Analysis and Consequence Inference (w/o *CA&CI*), reasoning for only one step (*1step*), and responding with only answers devoid of any reasoning (*Answer*).

We develop several tasks based on ALFRED kitchen scenarios such as washing fruits and making juice. We introduce 15 specific human interferences at random intervals during the robot's execution. These interferences include blocking the juicer with a pear, grasping an apple, placing an orange into the microwave, etc. This leads to the creation of 51 reactive planning task variations in total. Of these, 8 are designated as seen tasks for in-context learning, while the remaining 43 are set as unseen tasks for testing. We implement Seen Success Rate (SSR) and Unseen Success Rate (USR) for evaluation, measuring the proportion of tasks successfully completed in the seen and unseen categories respectively. Moreover, we utilize GPT-3.5 and GPT-4.0 in our experiment, with the results detailed in Table III.

We can draw several conclusions from Table III. From an overall view, our prompting method demonstrates superior per-

TABLE III: Success rates of two GPT models in completing seen and unseen tasks under our prompting method and its eight ablations. Each part of our re-planning prompt and the multi-step three-hop mechanism can enhance LLMs' re-planning capabilities. ***Note:*** *"-" signifies the absence of seen tasks as Case Demonstrations ($\mathcal{D}$) in the prompting method.*

Method	GPT-3.5		GPT-4.0	
	SSR	USR	SSR	USR
w/o $\mathcal{P}$	0.875	0.837	**1.000**	0.884
w/o $\mathcal{C}$	0.750	0.767	**1.000**	0.814
w/o $\mathcal{D}$	-	0.326	-	0.372
1step	0.875	0.721	**1.000**	0.791
Answer	0.625	0.535	**1.000**	0.698
w/o *CI*	0.875	0.767	**1.000**	0.837
w/o *PA*	0.875	0.814	**1.000**	0.884
w/o *CA&CI*	0.625	0.302	0.875	0.395
Ours	**1.000**	**0.884**	**1.000**	**0.907**

formance over all ablations in every metric when implemented with identical LLMs backbones. This underscores that each component of our prompts coupled with multi-step three-hop reasoning mechanism significantly contributes to the effective learning of LLMs on seen tasks and enhances their ability to generalize on unseen tasks. To be specific, according to the comparison among our re-planning prompt and w/o $\mathcal{P}$, w/o $\mathcal{C}$, w/o $\mathcal{D}$, we find that every component plays a crucial role in prompting LLMs for re-planning within a given embodiment. Removing each of these components from the re-planning prompt might lead LLMs to generate non-existent objects, undefined manipulations, unexecutable actions or infeasible new plans, which will reduce LLMs' re-planning performance on all metrics. Moreover, simplifying the multi-step three-hop reasoning mechanism to one-hop, two-hop, or single-step reasoning will also lead to a decline in LLMs' re-planning capability. The absence of one or two hops of reasoning (ours vs. w/o *CI*, w/o *CA&CI*, w/o *PA*) can make LLMs' re-planning logic unclear and incoherent, which may lead to the generation of erroneous new plans. Replacing multi-step reasoning with single-step reasoning makes it difficult for LLMs to fully analyze the impact of each environment feedback, which results in inadequate adjustments of plans. Notably, we find that LLMs perform better on unseen tasks across metrics if they directly provide final new plans without any reasoning rather than solely showing step-by-step plan adjustments (Answer vs. w/o *CA&CI*). From this, we can conclude that without cause analysis and consequence inference, it is challenging for LLMs to learn how to adjust plans from the Case Demonstration.

We also conduct a user study to further evaluate six prompting methods with distinct reasoning mechanisms. They require LLMs to explicitly demonstrate their analytical content, mirroring their reasoning logic during re-planning. Given that these analytical content are purely textual and challenging to compare quantitatively, we opt for a user study for evaluation. We invite 20 participants to rate the clarity and understandability of the content in each prompting method on a scale from 1 to 10 via a questionnaire.

Results from the user study (as shown in Fig. 4) show a clear preference for our re-planning method, which obtained the highest scores among all evaluated methods. This indicates that the analytical content of our method aligns most closely with the human cognitive logic in the context of re-planning. Three methods have simplified the three-hop mechanism, yet the scores of w/o *CI* and w/o *PA* are substantially higher than that of w/o *CA&CI*. A similar trend is evident in Table II. This suggests that methods involving explicit analysis offer clearer and more understandable logic compared to those merely adjusting plans. *1step* has lower ratings than w/o *CI*, although the former method contains a more comprehensive causal analysis than the latter. We can conclude that reasoning through multiple steps rather than a single step may better facilitate LLMs' re-planning even though the analytical content is not entirely complete.

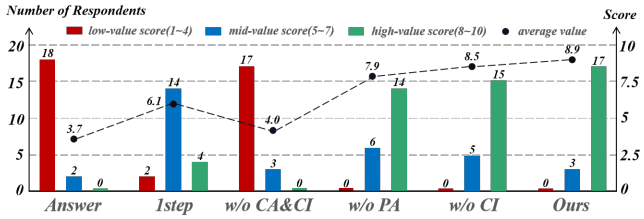

Fig. 4: User study results of evaluating six prompting methods with distinct reasoning mechanisms. Our method is best preferred by participants with highest average score of 8.9.

C. Implement Text2Reaction in Real-World Scenes

We implement Text2Reaction in two real-world scenes as shown in Figure 5. In the first scene, Text2Reaction is integrated with a UR tabletop robot dedicated to object manipulation. This robot is assigned to complete three distinct tasks - *Place1*, *Place2*, and *Place3*. In each task, the robot is required to respectively place 1, 2, and 3 different fruits into baskets of various colors. To test its adaptability, we interfere with its execution by turning on bright lights to impair visibility, filling the baskets to obstruct its usage or displacing fruits from the baskets back onto the table. In the second scene, Text2Reaction is integrated into a mobile manipulator equipped with a JAKA arm, necessitating navigational capabilities. The robot is asked to complete *Walk-Place* tasks which require transporting food items to a microwave situated in a different room. We interfere by closing the room door to obstruct its path, closing the microwave to hinder its use. We conduct ten rounds of testing for each task type, each involving two to three human interferences. These interferences may either facilitate, obstruct, or have no impact on the robots' execution. Moreover, we employ a grounding network [22] as the Env to provide environment feedback for planning.

Real-world environment changes are complex and varied, and it is impossible to enumerate all changes in re-planning prompts. To evaluate the adaptability of Text2Reaction to unseen changes in real-world experiments, we intentionally expose robots to new environment changes involving objects and their attributes that are unmentioned in the prompts. We employ success rate for evaluation, defined as the proportion

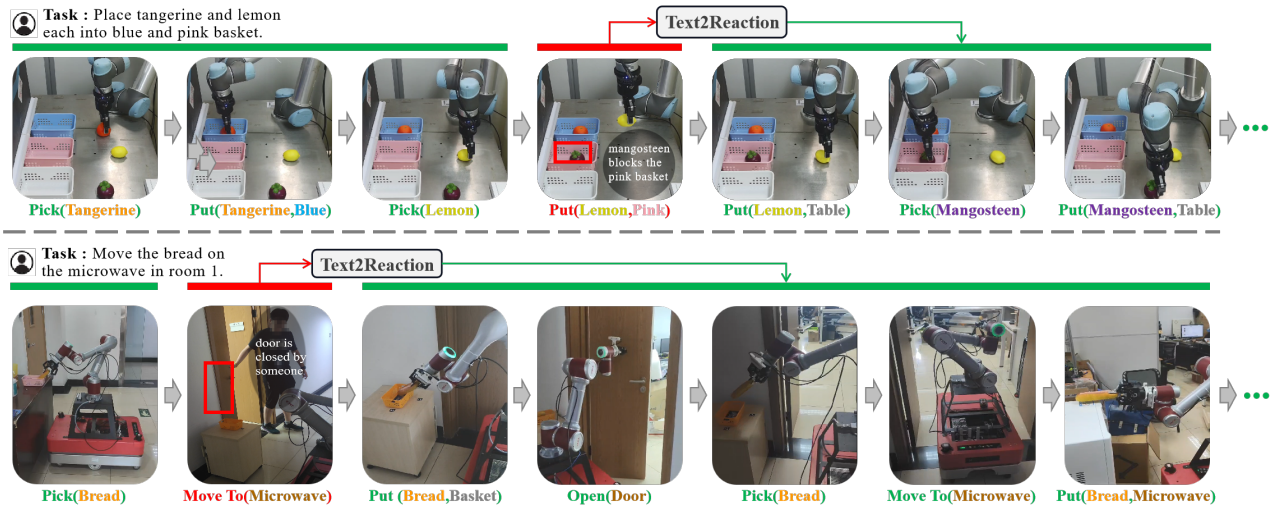

Fig. 5: Two demonstrations of real-world implementations. Tabletop robot plans to put the lemon into a pink basket but it is occupied by mangosteen (top). Mobile manipulator plans to near the microwave but the door is closed by a passerby (bottom).

of testing rounds in which robots successfully manage all interferences and ultimately complete the task.

From Table IV we show that robots achieve an average success rate of 87.5%. Specifically, the mobile robot succeeds in all test rounds while the tabletop robot succeeds 25 times, fails 5 times. The primary cause of failures is attributed to the complex reasoning logic required for tabletop tasks, which is distinct from everyday tasks. Despite the re-planning prompt providing only a limited number of examples, we are surprised to find that Text2Reaction still successfully completes most tasks amidst various human interferences. This indicates that equipping Text2Reaction with our re-planning prompts enables LLMs to exhibit a remarkable ability in complex reactive planning when faced with diverse environment feedback. Relevant demonstrations are included in the accompanying videos.

TABLE IV: Results of Text2Reaction and two traditional methods in adapting to unseen environment changes in real-world scenes.

Method	*Place1*	*Place2*	*Place3*	*Walk-Place*
Strip-like [6]	4	4	4	4
Learning-based [27]	4	4	3	4
Text2Reaction	**10**	**9**	**6**	**10**

To demonstrate the advantages of using LLMs in our framework for adapting to unseen environmental changes, we further introduce a strip-like method [6] and a learning-based method [27] for comparison under the same experimental setup. The experimental results from Table IV show that both methods succeed at most 4 test rounds. The strip-like approach cannot adapt to some new changes without additional detailed definitions of new objects and attributes. Meanwhile, the deep learning network can only generalize to a limited range of specific objects and attributes encountered during training and is unable to extend to unseen changes. Conversely, Text2Reaction neither requires extensive time for manually defining rules as strip-like methods do, nor depends on a large amount of data or carefully designed training strategies as learning-based methods do. It shows notable transferability in adapting to unseen changes with only a few cases for learning.

Furthermore, to demonstrate the advantages of chained reasoning used in Text2Reaction, we replace it with Tree-of-Thought (ToT) for testing under the same experimental setup. ToT [38] is a recently proposed reasoning mechanism that allows LLMs to consider multiple different reasoning paths, look ahead and backtrack when necessary to make global choices. We show that, when achieving a comparable success rate, ToT requires approximately 15 times more calls to LLMs and 10 times longer reasoning time compared to ours.

V. Conclusions, Limitations & Future Works

We introduce Text2Reaction, a framework that generates initial task plans before executing user instructions and updates these plans through multi-step three-hop reasoning using re-planning prompt, ensuring the instructions are successfully performed to completion. Experimental results underscore our prompting method's excellent performance in aiding LLMs with reactive planning. They also demonstrate that Text2Reaction can significantly enhance continuous reactive planning in highly dynamic environments even with minimal learning examples. This is particularly notable in challenging long-horizon scenarios with multiple disturbances.

Despite demonstrating impressive performance, our approach still has some limitations. Firstly, in the real world, multiple environmental changes may occur simultaneously, but our method currently handles only one at a time. In future work, it could be beneficial to consider prioritizing multiple environmental changes based on task progress, feeding one or more critically needed sources of feedback into the task planning model. Secondly, large language models possess comprehensive world knowledge but may fall short when dealing with more complex manipulation tasks in specific domains, such as customized assembly tasks. In the future, we will explore integrating explicit expert knowledge with large language models, enabling LLMs to adapt to different specialized tasks and achieve accurate task planning.

REFERENCES

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[2] Y. Z. and. Joyce Yue Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. In *Findings*, 2021.

[3] S. Bagchi, G. Biswas, and K. Kawamura. Task planning under uncertainty using a spreading activation network. *IEEE Trans. Syst. Man Cybern. Part A*, 30:639–650, 2000.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 35, 2022.

[5] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11509–11522, 2022.

[6] M. Colledanchise, D. Almeida, and P. Ögren. Towards blended reactive planning and acting using behavior trees. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8839–8845, 2016.

[7] S. Cooper and S. Lemaignan. Towards using behaviour trees for long-term social robot behaviour. *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 737–741, 2022.

[8] I. Dasgupta, C. Kaeser-Chen, K. Marino, A. Ahuja, S. Babayan, F. Hill, and R. Fergus. Collaborating with language models for embodied reasoning. *ArXiv*, abs/2302.00763, 2023.

[9] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. H. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. R. Florence. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, 2023.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Perez. Integrated task and motion planning. *Annu. Rev. Control. Robotics Auton. Syst.*, 4:265–293, 2020.

[11] C. R. Garrett, C. Paxton, T. Lozano-Perez, L. P. Kaelbling, and D. Fox. Online replanning in belief space for partially observable task and motion problems. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5678–5684, 2019.

[12] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *AAAI Conference on Artificial Intelligence*, 1987.

[13] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *Proceedings of Machine Learning Research*, 162:9118 – 9147, 2022.

[14] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. R. Florence, I. Mordatch, S. Levine, K. Hausman, and B. Ichter. Grounded decoding: Guiding text generation with grounded models for robot control. *ArXiv*, abs/2303.00855, 2023.

[15] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. R. 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 *Conference on Robot Learning*, 2022.

[16] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

[17] L. P. Kaelbling and T. Lozano-Perez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32:1194 – 1227, 2013.

[18] J. Li, D. Li, S. Savarese, and S. C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, 2023.

[19] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy. Reactive task and motion planning under temporal logic specifications. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12618–12624, 2021.

[20] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. R. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2022.

[21] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, 47:1345 – 1365, 2023.

[22] P. Lv, L. Ning, H. Jiang, Y. Huang, J. Liu, and Z. Wang. Neural symbol grounding with multi-layer attention for robot task planning. *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 155–162, 2022.

[23] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren. Towards a unified behavior trees framework for robot control. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427, 2014.

[24] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins. Pddl-the planning domain definition language. 1998.

[25] T. Migimatsu and J. Bohg. Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 5:844–851, 2019.

[26] S. Y. Min, D. S. Chaplot, P. Ravikumar, Y. Bisk, and R. Salakhutdinov. Film: Following instructions in language with modular methods. *International Conference on Learning Representations*, 2022.

[27] S. Mukherjee, C. Paxton, A. Mousavian, A. Fishman, M. Likhachev, and D. Fox. Reactive long horizon task execution via visual skill and precondition models. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5717–5724, 2020.

[28] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex. Cape: Corrective actions from precondition errors using large language models. *Arxiv*, abs/2211.09935, 2022.

[29] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2021.

[30] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746, 2019.

[31] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, and M. Katz. Generalized planning in pddl domains with pretrained large language models. *ArXiv*, abs/2305.11014, 2023.

[32] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2022.

[33] A. Suglia, Q. Gao, J. Thomason, G. Thattai, and G. S. Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *ArXiv*, abs/2108.04927, 2021.

[34] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[35] M. Tzes, V. G. Vasilopoulos, Y. Kantaros, and G. Pappas. Reactive informative planning for mobile manipulation tasks under sensing and environmental uncertainty. *2022 International Conference on Robotics and Automation (ICRA)*, pages 7320–7326, 2022.

[36] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 2022.

[37] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *ArXiv*, abs/2302.05128, 2023.

[38] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601, 2023.

[39] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022.