Interactive Task Planning with Language Models

Boyi Li* Philipp Wu* Pieter Abbeel Jitendra Malik
*Equal contribution University of California, Berkeley

***Abstract*— An interactive robot framework accomplishes long-horizon task planning and can easily generalize to new goals or distinct tasks, even during execution. However, most traditional methods require predefined module design, which makes it hard to generalize to different goals. Recent large language model based approaches can allow for more open-ended planning but often require heavy prompt engineering or domain specific pretrained models. To tackle this, we propose a simple framework that achieves interactive task planning with language models. Our system incorporates both high-level planning and low-level function execution via language. We verify the robustness of our system in generating novel high-level instructions for unseen objectives and its ease of adaptation to different tasks by merely substituting the task guidelines, without the need for additional complex prompt engineering. Furthermore, when the user sends a new request, our system is able to replan accordingly with precision based on the new request, task guidelines and previously executed steps. Please check more details on our Project Page and Demo Video.**

I. INTRODUCTION

The rise of Large Language Models (LLMs) and proliferation of chatbots highlight the importance of human interaction in an AI system. Beyond merely executing user commands, an autonomous agent should fluidly receive and incorporate feedback at any step during the execution process. Consider the seemingly straightforward human task of preparing a flavorful milk tea, which we study in this work. Such a task, while simple to humans, requires a robot to decompose it into numerous intermediate steps. Not only does the robot need to generate and precisely execute the steps, but the robot should also remain receptive to real-time modifications or feedback to the initial request. For example, the user might request some boba to be added to their drink. A robot should be able to seamlessly incorporate such interaction during operation.

In light of these challenges, we propose a simple framework for **I**nteractive **T**ask **P**lanning with language models, denoted as ITP. Our framework leverages LLMs to plan, execute, and adapt to user inputs throughout the task lifecycle. Figure 1 illustrates an exemplary interaction with our system. Our primary objective is to offer a blueprint for deploying real-world robotic systems that harness pretrain language models to coordinate the execution of lower-level skills of a robot in a simple manner. For our project, we utilize GPT-4 [1] as the language model backbone. ITP consists of two primary modules; (1) a high level planner which takes a input a prompt to specify the task and a user request and outputs a step by step plan and (2) a low level executor which tries to achieve a given step by converting robots skills into a functional API,

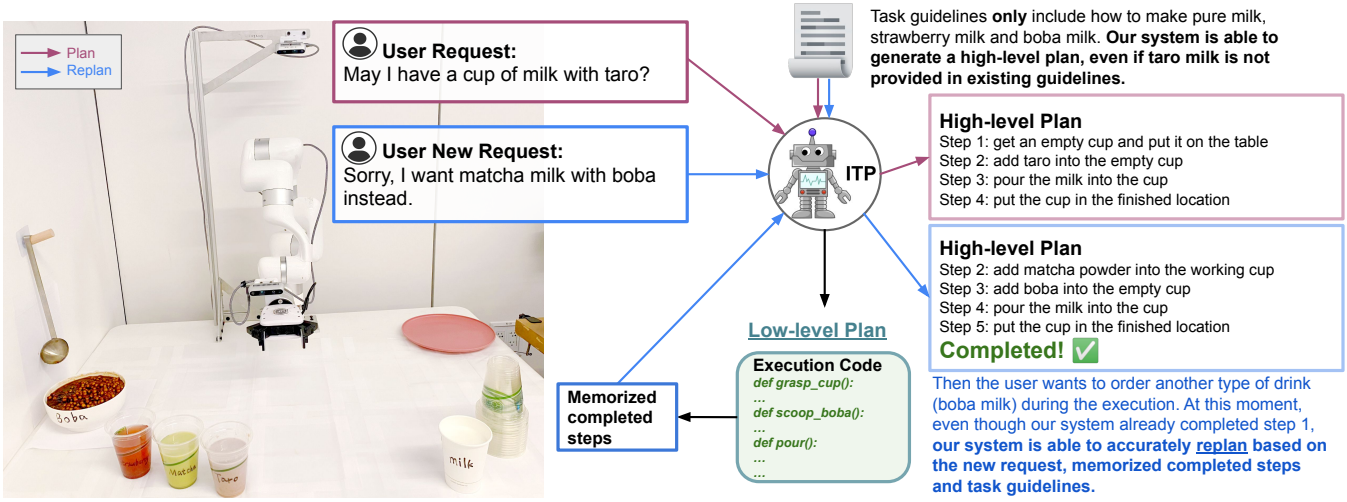

Fig. 1: An example of ITP. Our system will generate high-level plans and execute the low level robot skills through LLMs. It stores each step once complete, which we refer to as 'memorized completed steps'. We only give minimal guidelines for high-level plans on making pure milk, strawberry milk and boba milk. In the example shown, the user first requests '*May I have a cup of milk with taro?*' which is proccessed by ITP. As shown, although the recipe for taro milk is not provided in the guidelines, our system is able to generate an accurate executable high-level plan. After the robot has finished step 1, the user wants to revise the order to matcha milk with boba, which is also not provided in the guidelines. Our system is able to replan and make a new set of high-level steps based on the new request, memorized completed steps and task guidelines, which can then be completed by the lower level execution module to successfully complete the request.

which enables GPT-4s function calling capabilities to directly interact with the robot, abstracting code level details from the system. ITP does not require the training of additional value functions such as SayCan [2], [3], and does not require code level prompts such as Code as Policies[4] or ProgPrompt [5]. Furthermore, ITP dynamically generates novel plans and re-adjusts its plan based on user input. We hope our framework will be useful for accomplishing a wide range of interactive robot tasks and will release our codebase to foster advancements in this field. We outline the key features of ITP below:

1) ITP is a novel training-free robotic system for interactive task planning with language models. We showcase ITP in the context of a real world boba drink-making robot which integrates planning, vision and skill execution.
2) ITP is robust and can generate executable plans from a limited set of existing recipes, showing its adaptability.
3) Our system converts the robot skills into a functional language based API that can be leveraged by GPT-4. This enables a user to prompt the system through natural language rather than code, removing the need for code level prompt engineering.
4) Our system exhibits robustness in adapting to user request during execution, allowing it to consider the updated goals, previously completed steps, and task guidelines in order to replan new steps.

II. RELATED WORK

A. Task planning

Task planning, the problem of developing a plan to achieve a desired goal, is an integral component of our work. Traditionally, task planning in robotics commonly leverages symbolic planners which reduce the planning problem into a search problem [6], [7]. Practitioners define the problem in a declarative language, which can be restrictive as it requires meticulous definitions of the problem parameters, such as actions, their preconditions and their effects [8], [9], [10], [11]. Task and motion planning (TAMP), takes task planning a step further and also jointly considers the lower level execution during higher level planning [12], [13]. TAMP methods also consider symbolic representations and leverage search algorithms to extract the final sequence of lower-level primitives and has seen success in robotic manipulation [14], [15], [16]. As the search space can often be prohibitively large, some methods leverage hierarchy and/or sampling [17], [18], [19], [20]. Our approach replaces traditional planning pipelines with LLMs, offering common-sense reasoning, enhanced interaction capabilities, and the ability to define the problem's scope using natural language.

B. Language Models as Planners in Robotics

Due to the popularity of LLMs, there has been a rising interest in leveraging LLMs as a policy in robot systems.

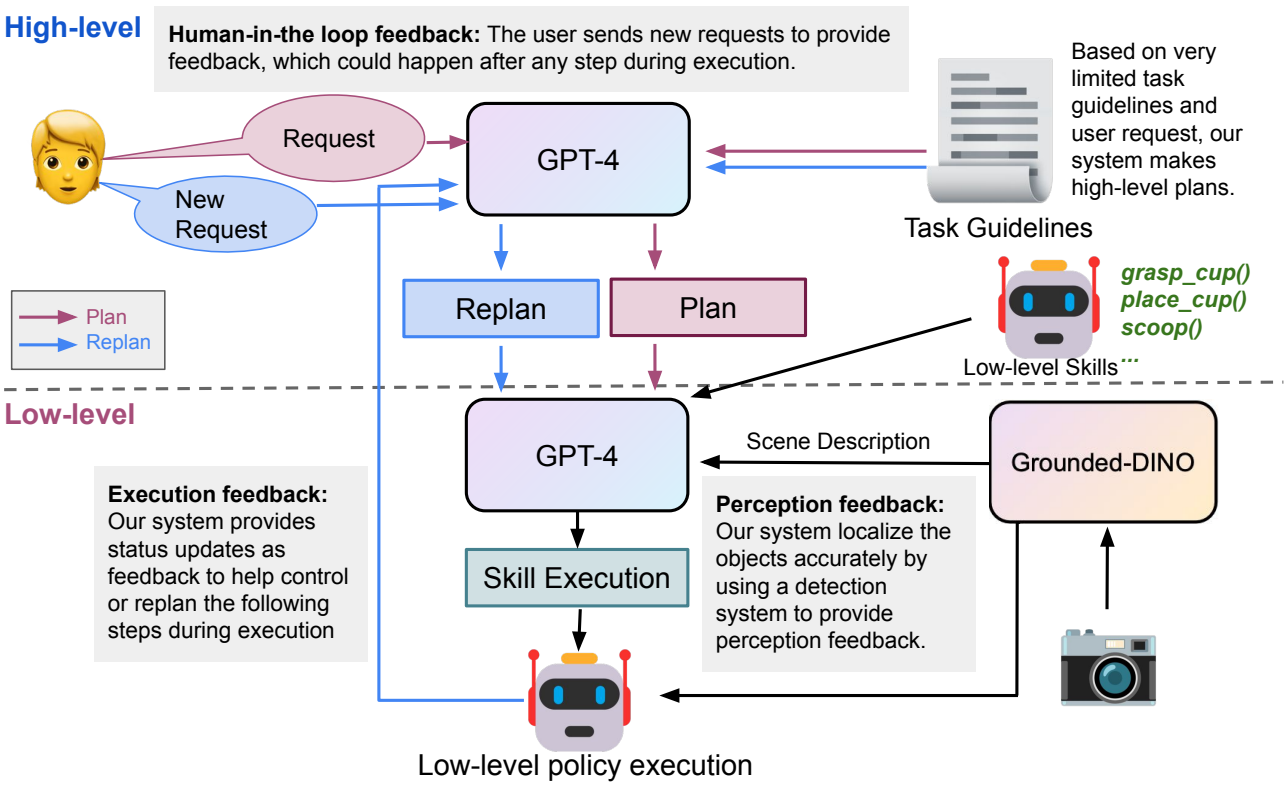

Fig. 2: Overview of ITP. In this paper, we design our system with Grounded-DINO to locate the object and GPT-4 to process the language. Our system generate high-level plans based on the user request and task guidelines. When the system is interrupted with new request, the system will replan on the basis of human-in-the loop feedback (new request), execution feedback (memorized completed steps) and task guidelines. Each generated high-level plan will go through GPT-4 to acquire the corresponding execution of low-level skills.

One work in this direction leverages LLMs as zero shot planners in simulated embodied settings [21] by converting the scene and task definitions into language, then letting the LLM directly predict actions. [2], [22], [3] follow in this line of work, coordinating many large pretrained models with a robot to solve various tasks. In contrast to approaches like SayCan [2], which necessitate a pretrained value function to ground actions, we rely on prompting the language model with task guidelines and robot skills. This implicitly encodes preconditions and effects reminiscent of traditional declarative task planning approaches but can be done so with natural language, which is more expressive and easier for the average user to tune. Tidy bot [23] shows that LLMs can help a robot follow a user's preferences based on a few examples. We also prompt the model with a small set of examples but explore generalization to new goals. Reflect [24] uses large models to make an agent recount their experiences and correct failures. LLMs have also been used to allow robots to seek help when uncertain [25].

A related approach, used in Code as Policies [5] and ProgPrompt [4] leverages the code writing capabilities of LLMs to generate code that a robot agent can execute directly. This often requires heavy prompt engineering of example code to show the model how to properly use the provided functions to accomplish a directive. Language-guided Robot Skill Learning [26], like us, takes a hierarchical approach to LLM planning, but assumes access to the simulator which provides ground truth state information. Voyager [27] uses LLMs to build a life long learning agent for Minecraft by having the agent explore and solve new tasks through writing code that interacts with the API.

Our work falls into this general category of leveraging LLMs to plan, and then execute actions in the environment. In contrast to prior work, we allow the LLM to generate a high-level plan based on contextual information. These low-level plans are then executed directly by an LLM with access to the functional API of the robot using a pre-trained VLM to ground the visual scene into primitives. Our work focuses on how to instantiate such a system in the real world.

III. METHOD

ITP offers a blend of high-level planning and low-level execution, powered by LLMs. In contrast to prior work [5], [4], our approach enables the LLM to create a high-level plan informed by contextual information in the form of a list of steps. Each step of this plan is subsequently realized by another LLM with access to the functional API of the robot. A pre-trained VLM grounds the visual scene into language. Our work focuses on how to instantiate such a system in the real world. Our framework, shown in Fig. 2, consists of three primary building blocks:

Visual Scene Grounding: ITP converts visual inputs into language using a Vision-Language Model (VLM).

LLMs for Planning and Execution: ITP generates high level plans and executes lower level robot skills.

Robot Skill Grounding: ITP translates robot skills into a functional API, enabling LLMs to dictate robot actions.

Visual Scene Grounding. The VLM's role is to process the visual scene into a concise language description, which can further be processed for planning and task execution downstream. In our drink-making system, the visual grounding system accepts a list of menu items and generates corresponding bounding boxes. Using a simple mapping algorithm, we then approximate the x and y locations of each item in the robot frame. We employ the pretrained VLM: Grounded-DINO [28], a variant of the original DINO model [29] fine-tuned for extracting 2D bounding boxes given language descriptions. The vision system gives a holistic 'understanding' of the scene, despite the location assignments being imprecise.

LLMs for Planning and Execution. We utilize GPT-4 [1] as our language model, one of the most capable LLMs available at the time of this writing. Our approach employs two language agents. The high level planner takes as input a given prompt, task guidelines, and a user request, and outputs a step-by-step plan to execute the request. It also retains past user interactions for any necessary replanning. The second LLM, provided with information about the scene and robot skills, takes each generated step and attempts to execute it. Task guidelines, described using natural language, outline the scope of the robot's tasks and are provided to the high level planner. In our milk tea system, the task guidelines consist of a select set of menu items, their corresponding preparation steps, and a list of relevant ingredients. This includes the procedures for a few drinks like 'pure milk' and 'boba milk'. Our system utilizes these guidelines to determine the feasibility of making a new drink based on available materials. Leveraging LLMs' fewshot learning capabilities, [30], ITP can generalize from the baseline guidelines to make detailed steps for other drinks such as 'boba strawberry milk' or 'taro milk'.

Robot Skill Grounding. The language model interfaces with a predefined skill set in Python that controls the robot. These skills are translated into a functional API by parsing of function definitions and related doc strings. This can be directly used with GPT's function-calling layer [1]. In contrast to methods like ProgPrompt or Code as Policies, our system does not require examples or function details when prompting the LLM. Instead, more detailed prompting of the language model can be specified via natural language in the documentation of the functions.

Beyond the three aforementioned components, ITP considers new requests from the user as *human-in-the-loop feedback*. The system will consider completed steps, task guidelines, the new request, and the chat history to generate a new plan. We explain the details of an example in Figure 3. We also showcase ITP's adeptness in planning and adaptive replanning of the same example in Figure 4.

IV. EXPERIMENTS

A. Robot Experiments

In our experiments, we focus on a drink-making system. Within the given scene, the robot is supplied with a set of ingredients which it must combine to produce a specific drink.

Our setup also has an overhead camera which feeds images to Grounded-DINO model for scene understanding.

For the robot, we provide a predefined set of skills, which include actions like "grasp_cup", "pour", and "scoop_boba_to_location". The "grasp_cup" skill is implemented with a feedback policy that centers the gripper on the cup, given the approximate location from the scene description, enabling the robot to grasp it reliably. The "pour" skill is designed to accept a location and a descriptive cue of the ingredient being poured. This level of specification enables milk to be poured more than specific flavors. For example, when making a matcha latte, the *pour* function will be provided "*matcha*" or "*milk*" as inputs. When the input is "*matcha*", the controllable tilt angle will be small, while when the input is "*milk*", the controllable tilt angle will be much larger. This ensures that the robot can pour more milk and a bit of matcha liquid.

B. Comparison on Task Planning

We consider Code as Policies as a baseline. Code as Policies provides a formulation for language model-generated programs executed on real systems by prompting a text completion model with code examples. For a fair comparison, not only do we provide Code as Policies with the same information as given in ITP in the form of comments, but we also provide an additional 40 lines of code prompts providing example usage, as is done in Code as Policies. For both ITP and Code as Policies, we provide user requests and task guidelines as inputs. The task guidelines include 3 instances, along with their associated high-level planning steps, current available material and other task-specific conditions. We show the detailed task guidelines below:

```
Options:
Pure milk, Strawberry milk, Boba milk
Instructions:
Pure milk
Material: milk
Steps:
0) get an empty cup and bring it to the working area
1) pour the milk into the working cup
2) put the working cup in the finished location
Strawberry milk
Material: strawberry jam, milk
Steps:
0) get an empty cup and bring it to the working area
1) add strawberry jam to the working cup
2) pour the milk into the working cup
3) put the working cup in the finished location
Boba milk
Material: boba, milk
Steps:
0) get an empty cup and bring it to the working area
1) add boba to the working cup
2) pour the milk into the working cup
3) put the working cup in the finished location
Available material we have now:
boba, strawberry jam, mango jam, matcha powder, taro,
milk,blueberry
```

Task Guidelines 1: making a drink

We evaluate the methods on two criteria: the number of high-level steps correctly generated and whether the real

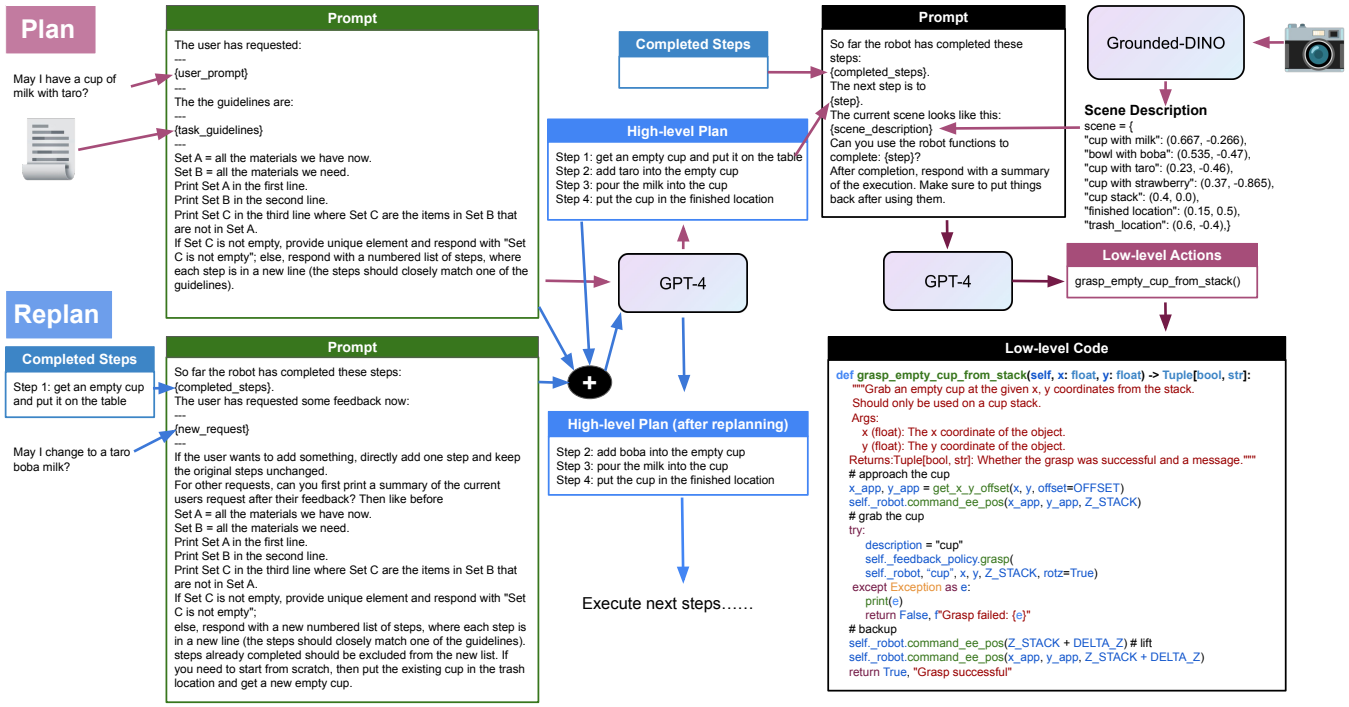

Fig. 3: Detailed diagram of ITP. ITP incorporates user requests, task guidelines, and memorized completed steps for planning or replanning. During "*Plan*": we feed user requests and task guidelines to complete the prompt and input it into GPT-4 to obtain a high-level plan. We input the completed steps and next step to complete the prompt and input the prompt into the lower level executor GPT-4 to call the corresponding low-level actions. Once the lower level executor completes a step, we will maintain the history by storing it into *Completed Steps*. GPT-4 directly makes function calls to a predefined robot skill library (which could be learned or handcrafted). During "*Replan*": we feed the completed steps and new request to create a new prompt, we append this new prompt to the previous conversation context and input the whole message into GPT-4 to obtain a new high-level plan. We refer this procedure as replanning, which previous language-based task planning methods have not considered. The low level executor then completes the next steps based on the new high-level plan.

robot successfully finished the task. We send user requests of varying complexity levels, including 'existed', 'zero-shot easy,' 'zero-shot moderate', 'zero-shot hard' and 'unavailable material'. 'Zero-shot' means the instruction for making the corresponding drink is not provided in the task guidelines. 'Unavailable' indicates that we don't have the material for the requested beverage. We show the results in Table I. We could notice that ITP is robust in high-level plan generation and can easily be generalized to novel instructions of unseen drinks or unavailable drinks. For example, the user sends the request '*I would like a cup of passion fruit milk.*' However, passion fruit jam is not available, so the system will provide the response '*Passion fruit jam is not available*' and stop the program. In comparison, Code as Policies failed to achieve this objective. To understand the failure case of Code as Policies, we provide some observations: 1) when making a cup of milk with boba, the system attempted to scoop boba from the working up, improperly adhering to the correct usage of the lower level skill. 2) When the prompt is more complex (9th row), the system adds milk first and then adds the boba, resulting in an incorrect execution order. 3) When the material is not available, it cannot justify that passion fruit doesn't exist. Additionally, since ITP is built based on task guidelines alone, it demands significantly less prompt engineering than Code as Policies, which makes our system very easy to use for various task planning purposes.

C. Replan with Human-in-the-loop Feedback

Our system is robust to diverse new requests during execution. To verify this point, we assess the task replanning performance on real robots in response to a user's new request, referred to as human-in-the-loop feedback. We display the results in Table II. We notice that ITP demonstrates its capacity to effectively handle a range of new requests, even after progressing through various steps of the task. The last example is of particular note, where ITP adds one step more ('Stir the mixture until the matcha powder is well mixed') before putting the working cup in the finished location. Here the language model assumes the need to stir the matcha due to the ambiguity of the correct procedure. Such superfluous steps can be reduced by adding restrictions in the task guidelines, which can easily be done by a general user of the system. This contrasts with methods like Code as Policies which

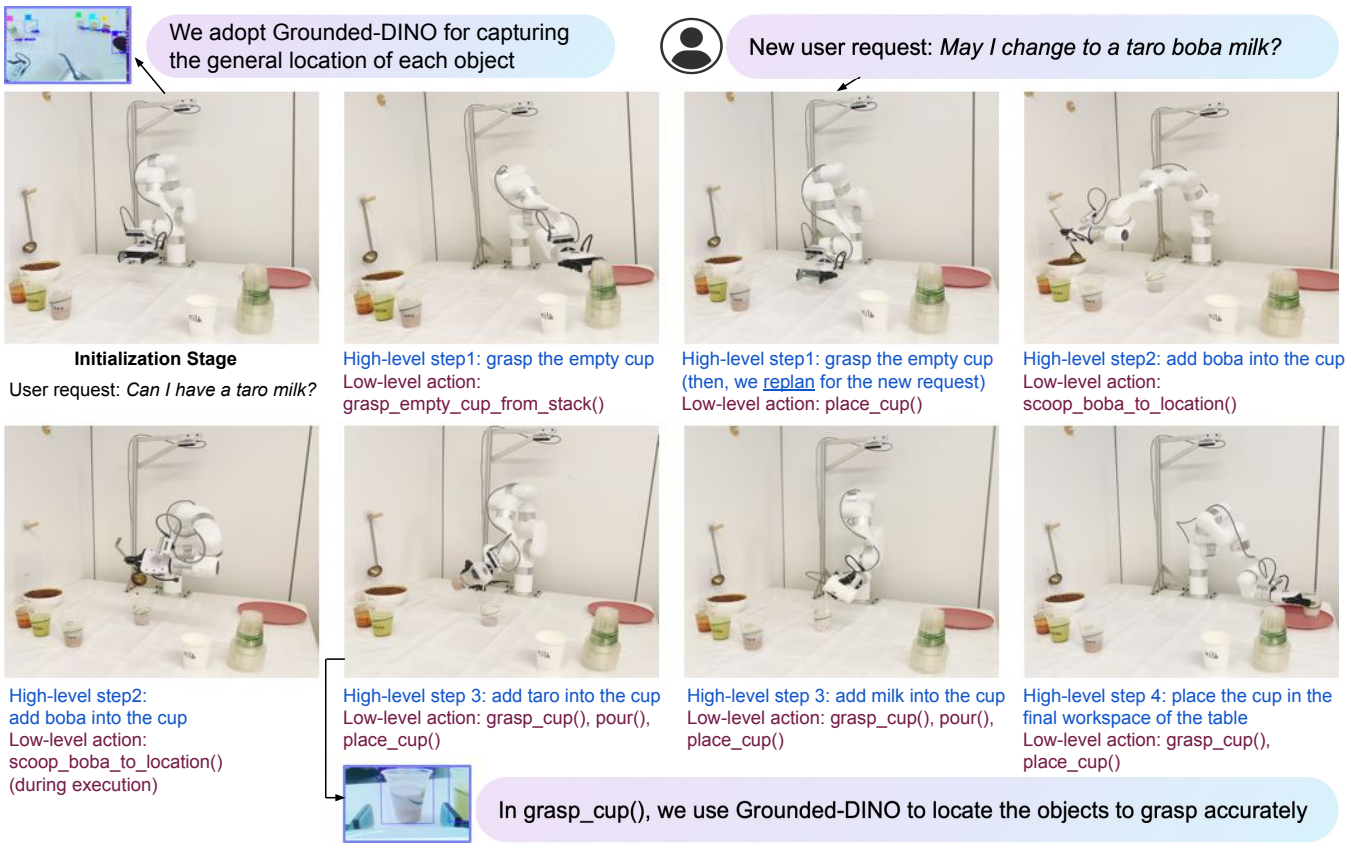

Fig. 4: An example of ITP to make a cup of taro milk with boba. Our system first makes a high-level plan based on the user request using GPT-4: step 1) grasp the empty cup, step 2) add taro into the cup, step 3) add milk into the cup, step 4) place the cup in the final workspace. For each step in the high-level plan, we feed step into another instance of GPT-4 and obtain the corresponding low-level actions which is directly executed on the robot. As for the perception component, ITP uses Grounded-DINO to capture the general location of each object and locate the object accurately when taking the actions. However, after grasping the empty cup, the user sends a new request '*May I change to a taro boba milk?*'. Considering the memorized completed steps as execution feedback, the system replans and generates the following high-level steps and low-level executions. The following plan has been changed to: step 2) add boba into the cup, step 3) add taro into the cup, step 4) add milk into the cup, step 5) place the cup in the final workspace.

User Request	Difficulty Level	Code as Policies		ITP	
		High-level Planning	Success	High-level Planning	Success
I would like to order a cup of milk.	Existed	3/3	✓	3/3	✓
I want to order a boba milk.	Existed	2/4	✗	4/4	✓
Can I have a cup of strawberry milk?	Existed	4/4	✓	4/4	✓
I want a matcha latte.	Zero-shot easy	4/4	✓	4/4	✓
May I have a cup of milk with taro?	Zero-shot easy	3/3	✓	3/3	✓
I want taro milk with boba.	Zero-shot moderate	3/5	✗	5/5	✓
Can I get a strawberry boba milk?	Zero-shot moderate	3/5	✗	5/5	✓
I want to order a strawberry matcha milk.	Zero-shot moderate	5/5	✓	5/5	✓
I'd order a strawberry matcha milk with boba.	Zero-shot hard	3/6	✗	6/6	✓
I would like a cup of passion fruit milk.	Unavailable material	-	✗	-	✓
Total	-	80%	5/10	100%	10/10

TABLE I: Quantitative results with real robots for high-level planning rate and success rate with various user requests. For high-level planning, we extract planning accuracy by dividing the number of successful steps by the total number of steps, shown as '*Successful Steps / Total Steps*'. We determine success by whether the robot successfully accomplishes the task. To calculate the overall high-level planning score, we average the performance across all user requests.

User Request	New Request	Step When New Request is Made		
		1st	2nd	3rd
Can I have a cup of strawberry milk?	*I want to add boba into the drink.*	4/4	3/3	5/5
I want a matcha latte.	*Sorry, I want boba bilk without matcha instead.*	3/3	5/5	5/5
May I have a cup of milk with taro?	*Can I replace the taro with strawberry?*	3/3	5/5	5/5
Can I get a strawberry boba milk .	*Sorry, can I reorder a strawberry milk?*	3/3	5/5	5/5
A strawberry matcha milk with boba.	*Can I just get matcha boba milk and no strawberry?*	4/4	5/4	7/6

TABLE II: Replanning performance with real robots given human-in-the-loop feedback. After the user sends a request, we interrupt the procedure before different steps (1st, 2nd, and 3rd). Note that our replanning system is robust in handling these new requests. Interestingly, for the last example, after the 2nd and 3rd step, ITP adds one step more ('Stir the mixture until the matcha powder is well mixed') before putting the working cup in the finished location, leading to 5 and 7 steps instead of 4 and 6 steps respectively. We assume this is because GPT-4 assumes matcha powder is hard to mix, while we select water-soluble matcha powder. Including the instruction 'matcha powder is water-soluble' in the task guidelines could address this issue.

require tuning prompts at the code level.

```
Options:
Wash one plate with rose flavor,
Wash all the plates and there are two plates,
Wash one plate and one fork
Instructions:
Wash one plate with rose flavor
Material: rose detergent
Steps:
0) grasp the dirty plate
1) remove large particle from the plate
2) open the dishwasher
3) pull out the rack
4) put one plate on the third rack
5) add rose detergent into the detergent dispenser
6) close the dishwaster
7) select the cycle and start dishwasher
8) after the dishwasher cycle is complete and the
dishwasher has stopped, wait a few minutes for the dishes
to cool down
9) make sure the plate is clean and dry, otherwise
go into step 8)
10) return the clean plate to the finished location
Wash all the plates and there are two plates
Material: original detergent
0) grasp the first dirty plate
1) remove large particle from the plate
2) open the dishwasher
3) pull out the rack
4) put the plate on the third rack
5) grasp the second dirty plate
6) remove large particle from the plate
7) put the plate on the third rack
8) add original detergent into the detergent dispenser
9) close the dishwaster
10) select the cycle and start dishwasher
11) after the dishwasher cycle is complete and the
dishwasher has stopped, wait a few minutes for the dishes
to cool down
12) make sure the plate is clean and dry, otherwise
go into step 8)
13) return all clean utensils to the finished location
Wash one plate and one fork
Material: original detergent
0) grasp the dirty plate
1) remove large particle from the plate
2) open the dishwasher
3) pull out the rack
4) put the plate on the third rack
5) grasp the fork
6) remove large particle from the fork
7) put the fork on the first rack
8) add original detergent into the detergent dispenser
9) close the dishwaster
10) select the cycle and start dishwasher
11) after the dishwasher cycle is complete and the
dishwasher has stopped, wait a few minutes for the dishes
to cool down
12) make sure the plate and fork are clean and dry,
otherwise go into step 8)
13) return all clean utensils to the finished location
Available location we have now:
* first rack for forks and small kitchen utensils
* second rack for bowl/cup
* third rack for plate/big kitchen utensils
Available material we have now:
rose detergent, original detergent
```

Task Guidelines 2: dishwashing

User Request	Task Type	High-level Planning
Wash one dirty plate with rose flavor.	Existed	11/11
Please wash 1 dirty bowl with rose flavor.	Zero-shot easy	11/11
Please clean the 2 dirty cups.	Zero-shot easy	14/14
Wash all forks, there are 3.	Zero-shot easy	17/17
Can you wash 2 plates? (New request: Can you wash another?)	Zero-shot easy	17/17
Please wash 2 forks and one bowl.	Zero-shot moderate	17/17
May you wash 2 cups and 2 plates?	Zero-shot moderate	20/20
Please wash 2 fork, 2 plate and 2 bowl.	Zero-shot hard	27/27
Wash 2 plates,1 bowl, 1 fork and 1 knife with rose flavor.	Zero-shot hard	23/23
Wash one dirty plate with lemon flavor	Unavailable material	-
Total	-	100%

TABLE III: Generalization to dishwashing task. We only need to change the text guidelines to make an accurate high-level plan. Since using the dishwasher to clean the dishes doesn't contain misleading material or content, the high-level planning rate is 100%. Please note that different utensils should be placed in different locations in the dishwasher, while ITP remains resilient in generating precise plans for each step, ensuring the correct order and appropriate location for different utensils. We envision the versatility of ITP's capabilities being applicable to a wide range of tasks.

D. Generalization on Other Tasks

ITP is simple to adapt to new tasks. The system is principally reliant on task guidelines during high-level planning and predefined function during low-level execution. This structure negates the need for intricate code implementation examples, subsequently making the system easier to adapt to new tasks. This structure negates the need for intricate code implementation examples, subsequently making the system's generalization to other tasks remarkably straightforward. Refer to Figure 3 for the necessary components that need to be adapted. *For adapting to a new task, only the* ***Task Guidelines*** *for task specification and documentation for the provided* ***Low-level Skills*** *need to be modified.* Optionally, the **Prompts** for the high and low level planner can also be tuned.

We adapt our system to study the high level task planning capabilities of a completely distinct task: dishwashing. We simply replace the task guidelines for 'making a drink' with 'dishwashing' and add function definitions that are needed for low-level execution. We show the dishwashing task guidelines below: We evaluate the generalization ability on two criteria: how many high-level steps are generated correctly and whether all the steps align with the ground truth (referred to as *Completed Status*). We show our results in Table III. We find out that ITP performs very well on the novel dishwashing task. It has the capability not only to produce precise and novel instructions for new objectives but also to exhibit resilience when faced with entirely different tasks.

V. DISCUSSION

Conclusion. In this paper, we propose a simple yet effective system, ITP, which melds the capabilities of Large Language Models in an interactive system that constructs plans, and performs tasks centered around the users needs. Encouragingly, it precisely interprets user requests, generates pertinent step-by-step plans, and achieves the desired outcome — a testament to the potential of such systems for real-world applications. We embody our system in a robot designed to make various drinks according to user preferences and adeptly demonstrate its ability to respond to feedback during execution. Our system is capable in the context of interactive task planning and replanning for robotics.

Limitations and Future Work. While ITP provides a working proof of concept of an interactive robot system, there is room for enhancing its capabilities with more powerful robot skills to tackle more intricate tasks. Similarly, the integration of more precise visual information that leverages 3D information would significantly elevate the robot's proficiency in understanding, planning, and interacting with its surroundings. We hope that our open-source system could stimulate further exploration of how established and emerging models can be harnessed to advance the realm of real-world robotics.

ACKNOWLEDGMENT

Philipp Wu was supported in part by the NSF Graduate Research Fellowship Program. We thank the Machine Common Sense project and ONR MURI award number N00014-21-1-2801. We thank Valts Blukis for the insightful and valuable discussion.

REFERENCES

[1] OpenAI, "Gpt-4 technical report," 2023.

[2] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[3] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for robot control," *arXiv preprint arXiv:2303.00855*, 2023.

[4] 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," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

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

[6] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, ser. The Morgan Kaufmann Series in Artificial Intelligence. Amsterdam: Morgan Kaufmann, 2004. [Online]. Available: http://www.sciencedirect.com/science/book/9781558608566

[7] B. Bonet and H. Geffner, "Planning as heuristic search," *Artificial Intelligence*, vol. 129, no. 1, pp. 5–33, 2001.

[8] Y. Jiang, S. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-based and asp-based systems," 2019.

[9] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL—The Planning Domain Definition Language," 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212

[10] V. Lifschitz, "What is answer set programming?" in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI'08. AAAI Press, 2008, p. 1594–1597.

[11] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," in *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, ser. IJCAI'71. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1971, p. 608–620.

[12] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," 2020.

[13] M. Mansouri, F. Pecora, and P. Schüller, "Combining task and motion planning: Challenges and guidelines," *Frontiers in Robotics and AI*, vol. 8, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.637888

[14] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004. [Online]. Available: https://doi.org/10.1177/0278364904045471

[15] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, nov 2017. [Online]. Available: https://doi.org/10.1177%2F0278364917739114

[16] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," 2020.

[17] F. Bacchus and Q. Yang, "The downward refinement property," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI'91. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, p. 286–292.

[18] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 5002–5008.

[19] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[20] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *Int. J. Rob. Res.*, vol. 32, no. 9–10, p. 1194–1227, aug 2013. [Online]. Available: https://doi.org/10.1177/0278364913484072

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

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

[23] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023.

[24] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724*, 2023.

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

[26] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," *arXiv preprint arXiv:2307.14535*, 2023.

[27] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

[28] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[29] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[30] 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. 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," 2020.