RoboGPT: an LLM-based Embodied Long-term Decision Making agent for Instruction Following Tasks

Yaran Chen, Wenbo Cui, Yuanwen Chen, Mining Tan, Xinyao Zhang, Jinrui Liu, Dongbin Zhao, and He Wang

***Abstract*—Robotic agents are tasked with mastering common sense and making long-term sequential decisions to execute daily tasks based on natural language instructions. Recent advancements in Large Language Models (LLMs) have catalyzed efforts for complex robotic planning. However, despite their superior generalization and comprehension capabilities, LLM task plans sometimes suffer from issues of accuracy and feasibility. To address these challenges, we propose RoboGPT,an agent specifically designed to make embodied long-term decisions for instruction following tasks. RoboGPT integrates three key modules: 1) RoboPlanner, an LLM-based planning module equipped with 67K embodied planning data, breaks down tasks into logical subgoals. We compile a new robotic dataset using a template feedback-based self-instruction method to fine-tune the Llama model. RoboPlanner with strong generalization can plan hundreds of instruction following tasks; 2) RoboSkill, customized for each subgoal to improve navigation and manipulation capabilities; 3) Re-Plan, a module that dynamically adjusts the subgoals based on real-time environmental feedback. By utilizing the precise semantic map generated by RoboSkill, the target objects can be replaced by calculating the similarity between subgoals and the objects present in the environment. Experimental results demonstrate that RoboGPT exceeds the performance of other state-of-the-art (SOTA) methods, particularly LLM-based methods, in terms of task planning rationality for hundreds of unseen daily tasks and even tasks from other domains.**

***Index Terms*—Embodied planning, Daily instruction following tasks, Self-instruction data generation, Embodied AI, Large language model**

I. Introduction

Embodied AI tasks, encompassing visual navigation and robotic manipulation, have progressed rapidly [1]–[3]. Anticipated future robots are projected to assist humans in executing complex daily tasks by following natural language instructions, such as '*make dinner*' or '*wash dishes*' [1], [4]. Existing methodologies, including template planning [5], [6] and expert-guided planning [7], [8], have demonstrated some success in handling seven types of instruction following tasks. Nevertheless, current agents fall short in fully comprehending instruction tasks, including aspects like object quantities, prefix content, and object dependencies.

Y. Chen, W. Cui, M. Tan, X. Yao, J. Liu and D. Zhao are with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China (dongbin.zhao@ia.ac.cn)

H. Wang is with Center on Frontiers of Computing Studies, Peking University

Large Language Models (LLMs) have made significant progress in the field of natural language processing [9]. Due to their extensive internalized world information, LLMs can solve complex embodied planning problems [10] more generically than template-based methods [5]. However, generic LLMs are overly broad and lack robotics expertise, resulting in plans that are frequently unfeasible for direct implementation by robots [10]–[12]. For instance, given the task '*If you are a robot and give me a plan of: slice an apple*', the generated plan is '*gather material*', '*wash your hands*', '*prepare the apple*', '*...*', which is not executable for a robot. Even LLM-Planner with a designated prompt [10] produces some illogical planning of complex daily tasks.

To address the accuracy issue of LLM planning, this paper seeks to enhance the planning capabilities of LLMs by integrating expertise from the robotics domain. We aim to fine-tune and enhance the LLM planning process to ensure both logical validity and optimal execution. To overcome the lack of domain-specific data, we create a specialized robotic dataset consisting of 67K embodied commands covering complex robotic activities. Data collection and fine-tuning are key steps in this enhancement process. We employ an automatic approach, utilizing a self-instruction method that enables ChatGPT to generate a batch of data. However, we find that over 60% of the generated data exhibits issues, predominantly incorrect logical relationships in the planning. To address this, we introduce a template feedback mechanism that prompts ChatGPT to introspect and rectify the generated data. The model initially identifies the data type and then inputs the relevant considerations for that type. This process guides ChatGPT to autonomously correct erroneous planning, substantially reducing the labor associated with manual data correction. Unlike LLM-Planner, RoboGPT trained on the above data: 1) can understand the prefix content to modify the planning subgoals according to the environments; 2) can understand the object dependencies to find invisible objects that are in containers, e.g., '*Put an apple from the microwave into the garbage*', RoboGPT plans '*find the microwave first*', while current methods directly find apples, failing to find them always; 3) can understand object quantities to handle tasks with more than two objects beyond other methods.

The implementation of the template feedback mechanism not only guarantees the production of high-quality data but also promotes the formation of logical thought chains, greatly improving the accuracy of the generated data. The fine-tuned RoboPlanner, trained on this refined dataset, demonstrates ro-

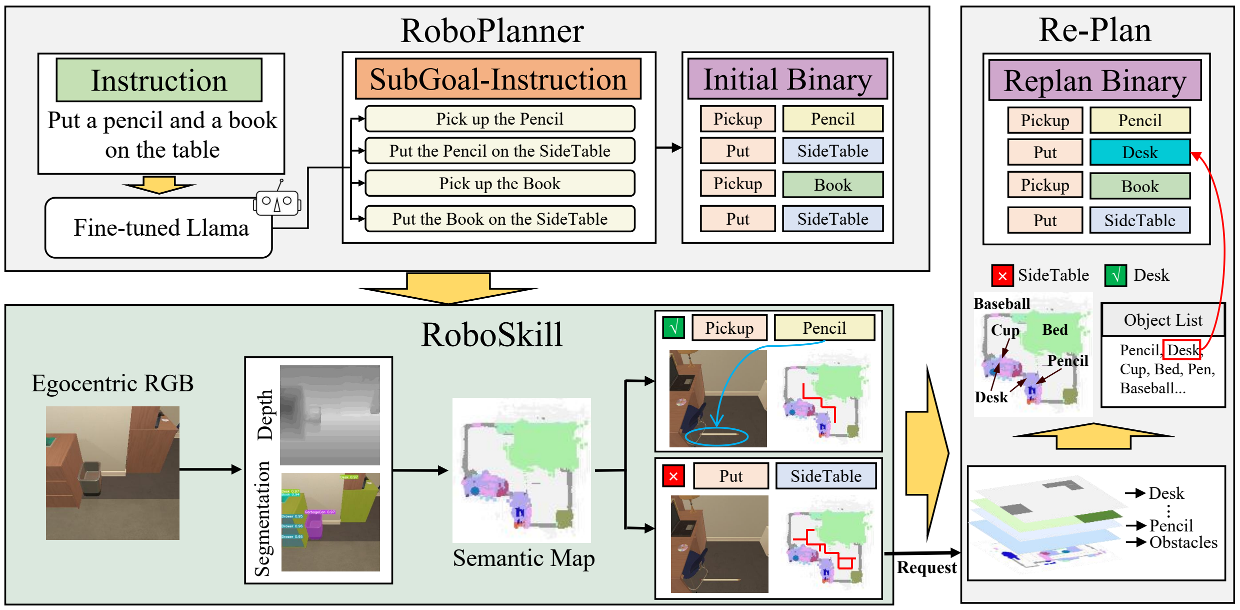


Fig. 1. The architecture of RoboGPT. RoboPlanner decomposes an instruction into logical subgoals. RoboSkill encompasses navigation, manipulation, and interaction with the environment sequentially based on subgoals. If a subgoal fails, Re-Plan receives feedback and generates a new plan based on environmental information, e.g., replace '*SideTable*' with '*Desk*'.

bust generalization capabilities and outperforms other planning methods. Ultimately, the LLM fine-tuned using this dataset exhibits improved performance in embodied instruction following tasks. This innovative approach to data collection and refinement combines with the unique application of a template feedback mechanism and constitutes a significant contribution to our work.

During instruction following tasks, the key to addressing feasibility issue is to enable the agent to perceive and consider the environment. Mapping instruction targets to objects in the environment remains a challenge, i.e., the instruction nomenclature diversity challenge. For example, the task is '*Put a pencil and a book on the table*', while common table types in the environment are '*sidetable*', '*diningtable*', and '*desk*'. The previous methods [5], [6] typically predict a type of table '*sidetable*' based on experience and plan '*find sidetable*'. However, when the '*sidetable*' is unavailable in the environment, the agent fails to locate a suitable table, leading to task failure. To address this problem, we propose introducing environment feedback and re-planning to align the environment objects with instructed task.

In summary, our primary contributions are as follows:

(1) To address LLM planning's accuracy issue, we create an embodied instruction planning dataset and propose RoboPlanner. This planner demonstrates robust generalization capabilities, enabling it to plan for hundreds of daily tasks. We also design a template-feedback mechanism, enabling the LLM to autonomously generate and modify planning data, thereby significantly reducing human effort in data correction. This is, to the best of our knowledge, the first high-quality, large-scale dataset for daily instruction planning task in the field of robotics.

(2) Re-Plan is developed to address the feasibility issue of LLM planning. It can dynamically adapt to the environment, effectively tackling the challenges posed by nomenclature diversity in instruction following tasks. Moreover, Re-Plan utilizes more accurate environmental information from the integrated segmentation model, thereby enhancing success rate (SR) in ALFRED tasks.

(3) We develop RoboGPT designed for handling complex daily tasks, including RoboPlanner, RoboSkill, and Re-Plan. This agent exhibits superior performance compared to state-of-the-art (SOTA) methods on both the ALFRED benchmark (SR: 60.7% versus 50.3%) and tasks requiring generalization (SR: 78.0% versus 34.0%).

II. Related Work

Conventional instruction planning methods. Languages instruction-based policy has been a popular research area in robotics, e.g., visual language navigation, visual question and answer, and daily tasks [13]–[15]. This paper focuses on long-term daily tasks that involve navigation and interaction. Hierarchical planning methods, which utilize rules or expert guidance to generate plans and employ low-level policies to execute them, have demonstrated their effectiveness [5], [6], [10]. These methods assume that the agent has domain knowledge or expert assistance in a closed universe. Planning issues are often described using PDDL or answer set programming in recent work [16]. A rule-based, search-based, or sampling-based planning algorithm has worked for mobile robots [17], [18] and robotic manipulators [3], [19]–[21]. However, these methods rely on specialized symbolic and logical representations of planning, which have limited generalizability and cannot handle unforeseen scenarios.

Instruction planning method based on LLM. Scholars are exploring the use of large language model-based planners as planning systems to tackle limitations in generalization [22]–[24]. They employ prompt engineering methods to generate

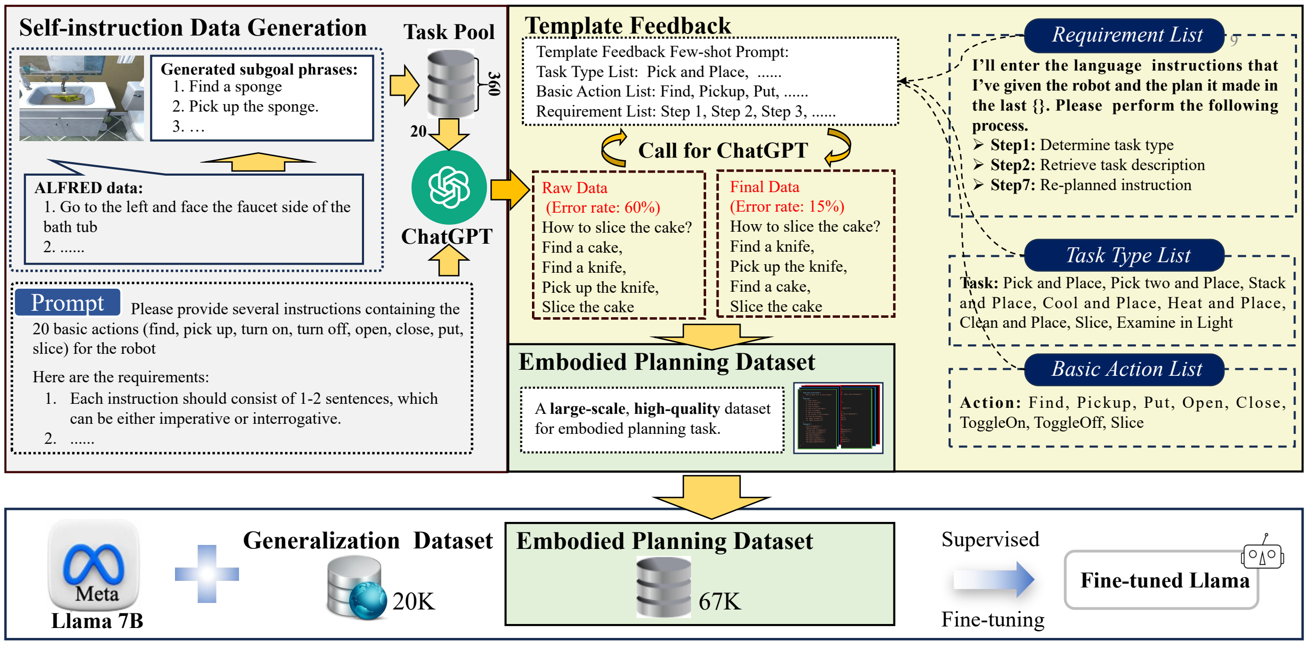

Self-instruction Data Generation
Task Pool
Generated subgoal phrases:
1. Find a sponge
2. Pick up the sponge.
3. …
360
20
ALFRED data:
1. Go to the left and face the faucet side of the bath tub
2. ……
ChatGPT
Prompt
Please provide several instructions containing the 20 basic actions (find, pick up, turn on, turn off, open, close, put, slice) for the robot
Here are the requirements:
1. Each instruction should consist of 1-2 sentences, which can be either imperative or interrogative.
2. ……
Template Feedback
Template Feedback Few-shot Prompt:
Task Type List: Pick and Place, ……
Basic Action List: Find, Pickup, Put, ……
Requirement List: Step 1, Step 2, Step 3, ……
Call for ChatGPT
Raw Data (Error rate: 60%)
How to slice the cake?
Find a cake,
Find a knife,
Pick up the knife,
Slice the cake
Final Data (Error rate: 15%)
How to slice the cake?
Find a knife,
Pick up the knife,
Find a cake,
Slice the cake
Requirement List
I'll enter the language instructions that I've given the robot and the plan it made in the last {}. Please perform the following process.
Step1: Determine task type
Step2: Retrieve task description
Step7: Re-planned instruction
Task Type List
Task: Pick and Place, Pick two and Place, Stack and Place, Cool and Place, Heat and Place, Clean and Place, Slice, Examine in Light
Basic Action List
Action: Find, Pickup, Put, Open, Close, ToggleOn, ToggleOff, Slice
Embodied Planning Dataset
A large-scale, high-quality dataset for embodied planning task.
Meta
Llama 7B
Generalization Dataset
20K
Embodied Planning Dataset
67K
Supervised
Fine-tuning
Fine-tuned Llama

Fig. 2. The framework of template feedback-based self-instruction data generation and RoboPlanner training process.

robot-friendly subgoals [25], [26]. Some planning methods utilize a procedural language for LLMs [27], while the planning process is conducted in an open-loop manner without access to world information [27]–[29]. Saycan [11]and Text2Motion [11], [12] use LLM to predict subgoals and select feasible actions based on environment or geometric constraints. LLMs can dynamically update the robot's plan online by calling LLMs multiple times [10]. However, the findings suggest that LLMs performance in long-term task planning is frustrating, limited by feasibility and correction, even in seemingly uncomplicated tasks [11]. This paper enhances LLM-based planning by incorporating a new robotic dataset and re-planning to boost feasibility and planning accuracy.

III. RoboGPT System

A RoboGPT system is proposed for daily instruction following tasks, consisting of RoboPlanner, RoboSkill, and Re-Plan (see Fig. 1). Given an instruction, RoboPlanner decomposes it into logical subgoals. RoboSkill performs navigation or manipulation skills based on subgoals, produces actions that interact with the environment, and completes all subgoals sequentially. If a subgoal remains uncompleted, Re-Plan gathers feedback and creates a new plan using the environmental information. Two key points need to be noted:

- Beyond the mundane and repetitive daily tasks, RoboGPT builds a more complex and diverse embodied planning dataset. RoboPlanner, equipped with common-sense knowledge of daily tasks and expertise in robotics, is trained to tackle complex and long-term decision challenges, including the ability to locate concealed objects within containers.
- Due to the nomenclature diversity between the instruction and the environment, the same object can be referred to using different linguistic representations. Re-Plan addresses the challenge of nomenclature diversity by mapping multiple instruction target representations to corresponding objects in the environment. Additionally, an enhanced perception model is designed to collaborate with Re-Plan, ensuring accurate and feasible re-planning capabilities.

A. RoboPlanner

Although LLMs already demonstrate remarkable performance in general-purpose scenarios, they still underperform in vertical domains like robotics due to a lack of domain expertise and data. Within an academic budget, training an LLM in a specialized domain requires a strong pre-trained LLM and high-quality training data.

Therefore, we build a 67K high-quality embodied planning dataset with template-based self-instruction data generation. The dataset is used to train a planning model, based on Llama [9], capable of handling daily tasks with long-term decision making. The process of dataset generation and model training is illustrated in Fig. 2.

Publicly available data for planning robots with long-term decision making in daily tasks is limited. The most relevant ALFRED task [15] contains only 8K expert trajectories covering seven task types, which is insufficiently diverse. As a result, we not only derive 60K samples of various types from the ALFRED tasks, but we also employ self-instruction to produce 7K samples with a wider range of task descriptions and types.

1) Embodied Planning Dataset: By transforming and expanding the ALFRED task into robot-friendly subgoal phrases, we ultimately obtain an embodied planning dataset of 67K instruction instances. In the ALFRED task, 8K expert trajectories contain high-level instructions (task instructions) and low-level instructions (planning descriptions), e.g., a high-level instruction is: '*Put a clean sponge on a metal rack*', and the low-level instructions are: '*Go to the left and face the faucet side of the bath tub. Pick up the left-most green sponge from the bath tub. Turn around ... left of the lotion bottle*'. The low-level instructions contain a substantial amount of scenario-specific environmental information, which makes it challenging to apply them to different scenarios. Therefore, we convert the detailed, low-level instructions into subgoal phrases. Through analysis, we identify the consistent planning steps for each task and abstract them into a single series of subgoals. For example, rewrite the above low-level instructions as '*find a sponge; pick up the sponge; find a sink; put the sponge in the sink;... put the sponge on the metal rack*'. The planned subgoals are all combinations of robot skills like navigation, grasping, opening, putting, and so on. Therefore, it's adaptable to any robot and task context. Based on the seven types of ALFRED tasks, we derive five new tasks and construct planning templates to enrich embodied planning data, generating a total of 60K samples.

The data from ALFRED demonstrations has limitations in terms of diversity (only seven + five types) and quality, which hinders the generality of the tuned model. The suboptimal quality of the data stems from the employment of crowd-sourcing techniques for task instructions, which involves three individuals watching videos and subsequently providing their individual descriptions. There are numerous inconsistencies in the descriptions of objects and actions depicted in the videos among the three individuals. Consequently, approximately 20% of the data descriptions are incorrect [15]. While the quantity, diversity, and creativity of human-written instruction data are limited, we tackle this issue by employing a three-step data generation process, resulting in the generation of 7K generalized data.

Step 1: Self-instruction Data Generation. Following Self-Instruct [30], we employ ChatGPT to generate and expand the ALFRED dataset in a few-shot setting. We create precise prompts that enable ChatGPT to convert instructions into robot-friendly subgoals. In these prompts, the agent should fulfill the common requirements of various real-world applications, ensuring that the generated steps are practical for a robot to execute. Here's an overview of how ChatGPT generates the data: Initially, we manually select 360 samples from ALFRED's dataset and add them to the task pool. Subsequently, we randomly select 20 samples from the task pool. The few-shot prompt includes these samples, as well as the custom prompt. Finally, ChatGPT processes the few-shot prompt and generates diverse samples.

Step 2: Template Feedback. We introduce a template feedback mechanism that prompts ChatGPT to introspect and make corrections to the generated data. Due to the scarcity of embodied instruction planning datasets for generic LLMs, over 60% of the data generated in step 1 contain issues, primarily incorrect logical relationships in the planning. We integrate the reflective mechanism into the data generation process. Specifically, we utilize template-based feedback to input raw data, basic action list, task type list, and task requirement list into ChatGPT. ChatGPT then engages in reflective analysis to evaluate the quality of the data and categorize it. Finally, we employ template-guided corrective measures to iteratively refine the generated data. The specific details of the template feedback are depicted in Fig. 2.

Step 3, after the corrections in step 2, only 15% of the data exhibited issues. We manually rectify these instances, resulting in a final large-scale and high-quality embodied planning dataset comprising a total of 67K samples.

2) Supervised Fine-tuning: To enhance the embodied instruction planning capability, we employ supervised fine-tuning on Llama [9]. In addition to 67K embodied planning dataset, we augment the robot's training set with 20K generalization dataset obtained from the Internet and train only two episodes, enabling RoboPlanner to learn the robot's planning capabilities and maintain strong generalization skills. Following [31], we utilize cross-entropy loss to guide the model's convergence on the embodied tasks.

$$loss = -\sum_{h=1}^{H} y_h \log \hat{y}_h \tag{1}$$

where H represents the number of classes, y_h denotes the one-hot encoding of the true class of the h_{th} token, and $\hat{y}_h$ is the probability predicted by the model for the h_{th} token belonging to class h.

3) RoboPlanner: After training, RoboPlanner becomes capable of planning both basic and logically challenging tasks. For instance, given the task $Inst.$ '*Put a pencil and a book on the table*', RoboPlanner first generates N subgoal-instructions S_n, which are a series of short phrases, such as '*pick up the pencil*'. Subsequently, RoboPlanner matches S_n into robot-friendly initial binaries B_n, for example, '*Pickup Pencil*'. Finally, B_n is transmitted to the RoboSkill and executed sequentially.

$$\{S_n\}_{Inst.}^{N} = \text{RoboPlanner}(Inst.), B_n = \text{Match}(S_n), \quad \text{s.t.} \quad n \in \{1, 2, 3, ...N\} \tag{2}$$

Moreover, RoboPlanner can also accurately plan and manipulate hidden objects within containers, e.g., 'Get a towel out of the cabinet to soak it, then put it on the toilet', as shown in Fig. 4.

B. RoboSkill

RoboSkill completes navigation or operation skills based on the received instructions. Referring to Prompter [5], at each time step, the perception block receives an egocentric RGB image, detects object segmentation and depth, and updates the semantic map of corresponding regions. The semantic map guides the navigation block to look for the target. Once the target is found, the interaction block produces actions to interact with the environment until all subgoals are completed.

Semantic segmentation plays a crucial role in various applications due to its ability to provide object masks for interaction and facilitate the creation of semantic map [32]–[34]. The semantic map is subsequently utilized for subgoals such as Re-Plan and object navigation. The segmentation blocks in previous work have been found to exhibit omissions

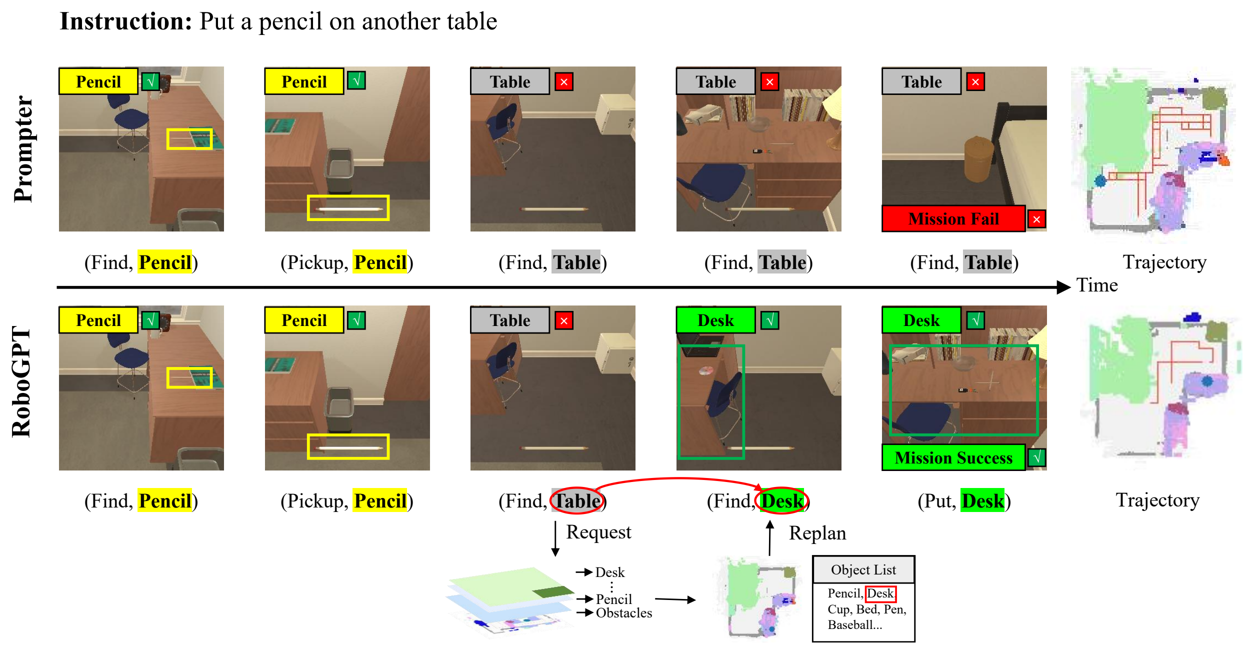


Fig. 3. Effectiveness of Re-Plan. Re-Plan will find similar alternative objects if the subgoal object cannot be noun-aligned with the object already present in the environment.

or misidentifications, which have a substantial impact on the accuracy of the semantic map. Consequently, this leads to a decrease in the performance of Re-Plan and the overall SR. We gather a dataset from ALFRED [15] seen environments to train a semantic segmentation model utilizing the FastSAM [35] backbone. This significantly enhances the accuracy of both object detection and the semantic map.

Besides employing more precise models, we also dynamically update the semantic map, which can eliminate the influence of object detection errors to some extent. The global semantic map $M_t^S \in [0,1]^{(2+C)\times M\times M}$ is a binary grid map, where C indicates the number of objects and $M \times M$ denotes the number of grids and each grid represents a 5 cm × 5 cm ground space. We make a dynamic selection of C and include all large objects in the semantic map. However, for small objects, we only consider those that are relevant to the goal and those that can be utilized for Re-Plan purposes. For instance, if the goal is to '*Place a bottle on the desk*' and the subgoal is to '*Find a glass bottle*', we would include objects such as '*glass bottle*', '*wine bottle*', '*soap bottle*', and other types of bottles in the semantic map.

We employ U-Net [36] and FastSAM [35] separately to estimate depth I_t^D and segmentation I_t^S, and utilize them to generate local semantic map $\hat{M}_t^S$ and binary observation map M_t^O:

$$(\hat{M}_t^S, M_t^O) = \text{MapModel}\left(I_t^D, I_t^S\right) \tag{3}$$

$$\begin{aligned} M_t^S &= \text{MapUpdate}(\hat{M}_t^S, M_t^O) \\ &= \hat{M}_t^S \times M_t^O \times M_{nbr} + M_{t-1}^S \times (1 - M_{nbr}) \\ &\quad if\ M_{nbr} \cap M_t^O \neq \varnothing\ and\ v_i\ not\ in\ M_{nbr} \end{aligned} \tag{4}$$

Then we update M_t^S using $\hat{M}_t^S$ and M_t^O, and finally, we obtain the inventory vector V_t = Object-Detector(M_t^S). Updating M_t^S can eliminate the effects of object detection errors, but this correction is not timely. It will only update the neighborhood region M_{nbr} around the target object v_i when the agent does not detect v_i in the current observation area. This means that if re-planning occurs before the update of M_t^S, Re-Plan will use an incorrect semantic map to modify the subgoals. Therefore, to address this particular scenario, RoboGPT employs the number of pixels n_{v_i} as a criterion to determine the reliability of confirming the presence of an object in the current scene and gets the confirmed inventory vectors $\bar{\mathrm{V}}_t$ from V_t.

$$\bar{\mathrm{V}}_t = [v_i^{'}\ for\ v_i\ in\ \mathrm{V}_t\ if\ n_{v_i} > p] \tag{5}$$

C. Re-Plan

One of the primary challenges in embodied AI is nomenclature diversity, where the targets mentioned in instructions and the objects present in the environment may have different names for referring to the same object. However, existing approaches [5] often neglect this particular issue. They primarily focus on task planning prior to execution, without fully comprehending the current environment. When there is a mismatch between the objects specified in the subgoal and the objects present in the environment, the agent will endlessly explore until it exceeds the maximum number of steps.

LLM-based planners [10], [23] usually input a list of all detected objects into the LLM and provide a new plan, which relies heavily on consuming LLM and image detection, resulting in wasted resources and incorrect re-planning results once detection fails.

Different from other planners, when RoboGPT fails to find the target even after thorough exploration, it utilizes semantic map for re-planning. Specifically, Re-Plan employs BERT [37] to calculate the similarity between the current target v_i and the set of $\bar{\mathrm{V}}_t$.

$$\mathrm{Sim}\left\{s_1, s_2, ..., s_k\right\} = \mathrm{BERT}\left(v_i, \bar{\mathrm{V}}_t = \left\{v_1^{'}, v_2^{'}, ..., v_k^{'}\right\}\right) \tag{6}$$

The most similar target ($v_j^{'}$, $j = \mathrm{argmax}(\mathrm{Sim})$) that satisfies the condition $s_j > 0.7$, is considered to be the same object as v_i. Replace v_i in S_n with $v_j^{'}$ to form a new subgoal-instruction $S_n^{'}$. The robot then proceeds to perform this new task. A more intuitive example of re-planning is illustrated in Fig. 3. Re-Plan will find similar alternative objects '*Desk*' if the subgoal object '*Table*' cannot be found in the environment.

IV. Experiments

We conduct a series of experiments to validate the ability of RoboGPT to handle daily tasks on both the ALFRED tasks and the constructed generic tasks. We also investigate and analyze the roles of different modules. There are several aspects of the proposed RoboGPT that require verification: 1) the static planning capability of RoboPlanner (see Sec.IV-C1). Verify whether RoboPlanner can effectively plan a series of subgoals to accomplish complex, long-sequence tasks without interacting with the environment; 2) the effectiveness of RoboSkill (see Sec.IV-C2); and 3) Re-Plan capability (see Sec.IV-C3). During the task, verify whether RoboGPT can re-plan the target objects based on the physical scene to address the issue of open vocabulary; 4) the performance of RoboGPT in solving complex daily tasks (see Sec.IV-D and IV-E); Hence, we perform a series of experiments and explain experimental setup, and experimental results.

A. Experiemental Setup

Metrics: Following [5], [10], we report the following metrics to assess the effectiveness of RoboGPT: success rate (SR), goal-condition success (GC), and high-level planning accuracy (HLP ACC). SR is the agent's overall task completion rate. GC is the ratio of completed goal-conditions, e.g., in '*Heating a cleaned apple*', '*washing*' and '*heating*' are goal-conditions. Using SR and GC, the path length weighted SR (PLWSR) and path length weighted GC (PLWGC) are defined as (path length of the expert trajectory)/ (path length taken by the agent). HLP ACC is the accuracy of subgoal planning [10].

Baselines: We primarily compare with three kinds of baseline methods. The first is the SOTA algorithm, CAPEAM [38], which utilizes the context of task instructions to predict the agent's next subgoal. Another one is Prompter [5], which relies on template-based methods for instruction planning. The last is LLM-Planner [10] and OPEx [39], which utilize ChatGPT for instruction planning.

First, we reproduce Promper [5] and LLM-Planner [10] to conduct testing on the Valid Unseen tasks with ground truth information. Prompter consists of three components: high-level planner, perceptual mapping, and navigation/interaction policy. The high-level planner used by Prompter is similar to the one employed by FILM [6]. It employs a classifier based on BERT to categorize each task into one of seven task types. Additionally, it predicts parameters associated with each task, including target objects, receptacles, parent objects, and slicing. One notable difference between Prompter and FILM lies in their semantic search modules. Prompter searches target objects based on landmarks, which speeds up the search. In the comparative experiments, the pre-generated subgoals provided by Prompter are utilized without any modifications. LLM-Planner [10] consists of two main components, the high-level planner and the re-plan module. These components are integrated into the overall framework. Thus, in the comparative experiments, the high-level planner based on Prompter's template is substituted with LLM-Planner. A total of 140 cases, 20 for each of the seven ALFRED tasks, are chosen. To assess the similarity between each training example and the current test example, the pre-trained BERT-base-uncased model is employed. The concept of similarity is established by calculating the Euclidean distance between the BERT embedding. The k-nearest-neighbor (KNN) retriever is employed to select the nine most comparable in-context cases. These examples are then fed into the LLM to generate a static version of LLM-Planner. To minimize the frequency of LLM calls, we provide LLM-Planner with an inequitable advantage by supplying it with all scene objects prior. Finally, to ensure a fair comparison between each method, we compare their results on the third-party platform ALFRED (Tests Seen and Tests Unseen environments).

Evaluation Dataset: The test set has four parts: 'Tests Seen' (1533 episodes), 'Tests Unseen' (1529 episodes), and 'Valid Unseen' (50 episodes) in ALFRED, as well as our own generated 'Generalization Task' (50 episodes). Although ALFRED has ground truth for each valid task, we find some of them have issues. Therefore, we select 50 tasks from 'Valid Unseen' where the instructions and ground truth are perfectly matched. The 'Generalization Task' consists of 50 complex high-level instructions that we annotate ourselves, and these tasks are entirely distinct from the seven types defined by ALFRED.

Training Details: RoboPlanner is fine-tuned from Llama-7b on NVIDIA DGX A100. Training data includes 67K embodied planning dataset and 20K online generalization dataset. To maintain generalization, the network is trained in 2 episodes with a 10^{-5} learning rate. RoboSkill trains a semantic segmentation model leveraging the FastSAM backbone [35] on collected 80K images for 100 epochs with a learning rate of 10^{-3}, batch size is 16.

B. Data Generation Results

We expand the ALFRED dataset to 67K using ChatGPT through self-instruction. However, due to the lack of robot instruction planning dataset for ChatGPT, over 60% of the instructions directly generated by ChatGPT contain logical errors. By implementing template feedback, the proportion of logical errors decreases to 15%, significantly improving the dataset's quality and reducing the manual calibration workload in later stages. Our method (self-instruction + template feedback) provides a way to generate robot instruction planning data on a large scale.

TABLE I
PERFORMANCE COMPARISON ON SEVEN TYPES OF ALFRED TASKS (VALID UNSEEN) AND GENERALIZATION TASKS

Method	**Valid Unseen**					**Generalization Task**
	SR	PLWSR	GC	PLWGC	HLP ACC	HLP ACC
High-level Instruction Only						
Prompter [5]	50	20.41	56.67	21.86	82	0
LLM-Planner [10]	32	13.54	47.33	18.74	66	34
RoboGPT	**60**	19.10	**69.83**	21.26	**96**	**78**
RoboGPT w.o. RoboPlanner	58	**22.18**	65.83	**24.17**	58	0
RoboGPT w.o. RoboSkill	56	21.80	64.00	23.51	**96**	**78**
RoboGPT w.o. Re-Plan	52	16.02	62.33	19.01	88	78

C. Ablation Experimental Result

1) Effectiveness of RoboPlanner: In the third-to-last column of Table I, RoboPlanner is replaced with the Prompter's planner [5], resulting in an 8% decrease in SR and a large decrease in HLP ACC. This indicates that RoboPlanner plays a crucial role in understanding tasks that involve long-term decisions, surpassing the capabilities of Prompter in this aspect.

In the designed Generalization task RoboPlanner performs well in complicated long-term tasks, as shown by its HLP ACC in Table I being far ahead of others. Prompter performs effectively on ALFRED tasks due to its specific template designed for ALFRED, but it lacks generalization to other tasks. On the other hand, LLM-Planner utilizes ChatGPT for logical reasoning, but its performance relies on pre-labeled data for the prompt. Although it demonstrates a certain level of generalization, the performance of LLM-Planner remains unstable.

Prompter and FILM [5], [6] view task planning as a classification task and utilize templates for planning. They may misjudge the task type, the parent target, and the target object when planning a task, e.g., the task is '*Put two potatoes in microwave*', while Prompter plans '*pick up an egg*'. It could be due to overfitting the training data.

LLM-Planner [10] uses ChatGPT for planning and may make logical errors when planning. For instance, when given a task like '*Place a glass with a knife in a sink.*', it plans '*Pick up glass then pick up the knife, and put knife in the sink*', which doesn't put the knife in the glass, failing to understand the relationship between object and container.

Beyond the above methods, the advantages of RoboPlanner can be summarized as follows: 1) **prefix understood**: using environmental information as a prefix prompt, it can produce practical planning; e.g., for the task '*There is a stove and no microwave, how to heat an apple*', RoboPlanner plans '*find a stove*' to heat the apple, while other planners usually plan to '*find a microwave*'; 2) **quantity understood**, RoboPlanner comprehends the need for robots to pick up objects individually rather than all at once, and it can plan tasks with 3 or more objects, while other approaches can only plan tasks with 2 objects. e.g., '*Put four books on the desk*'; 3) **object dependencies understood**, RoboGPT can use task instructions to infer the location of invisible objects. For instance, if the task is '*Get a towel out of the cabinet to soak it, then put it on the toilet.*', RoboPlanner will plan to '*open the cabinet*' first, whereas others will simply find a towel (shown in Fig. 4); 4) **tasks with ultra-long-term decision understood** RoboPlanner can understand and plan complex tasks with more than 30 subgoals. e.g., '*cut a slice of bread, warm it with the microwave, put it on the counter along with putting the knife in the cabinet*'.

2) Effectiveness of RoboSkill: When comparing RoboGPT without RoboSkill in the last second row of Table I, RoboGPT produces significantly higher SR (+4% improvement) and GC (+5.83% improvement), which indicates a major improvement in perception. As a result, RoboSkill can build an accurate semantic map and couple it with Re-Plan. The segmentation and detection abilities of RoboSkill demonstrate superior performance compared to other approaches. Furthermore, despite the high values of HLP ACC and GC achieved by RoboGPT, the SR of execution is relatively lower, suggesting that many tasks fail during the interaction. Consequently, there is room for improvement in the interaction algorithm to enhance the robot's SR during the tasks.

3) Effectiveness of Re-Plan: We employ Re-Plan with low-computing to achieve quick re-planning. Compared with RoboGPT without Re-Plan in the last row of Table I, RoboGPT achieves higher PLWSR and SR and can complete the task in the ALFRED environment with fewer steps, leading to a higher PLWSR. In cases where the target object mentioned in subgoal cannot be noun-aligned with any existing object in the environment, RoboGPT will find suitable alternative objects as substitutions, e.g., replace '*table*' with '*desk*'. This demonstrates how Re-Plan aids in understanding the environment and the alignment of objects and targets within instructions. Table I further highlights the critical role of Re-Plan in achieving an 8% increase in SR.

D. Experimental Results

Table I summarizes the performance of RoboGPT and other methods: Prompter [5] with template-based planner and LLM-Planner [10] with ChatGPT planner. RoboGPT achieves 10.00% absolute (20.00% relative) gain in SR on Valid Unseen tasks. Our RoboGPT also achieves a definite advantage in HLP ACC: 14.0% absolute (17.0% relative) gain on seven types of tasks, significantly exceeding other methods.

Notably, facing the generalization tasks beyond the seven type tasks, RoboGPT outperforms LLM-Planner [10] by **44%**, and Prompter [5] by **78%!** This implies that RoboGPT is extremely generalizable and capable of daily tasks of long-term decision making.

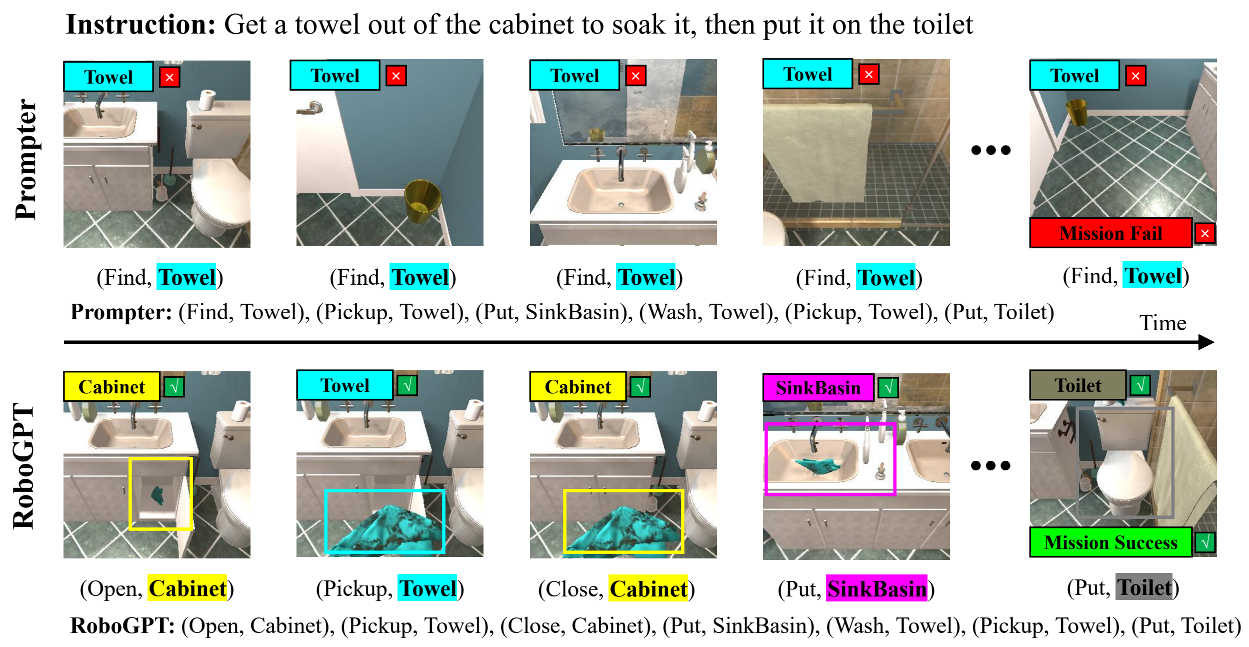
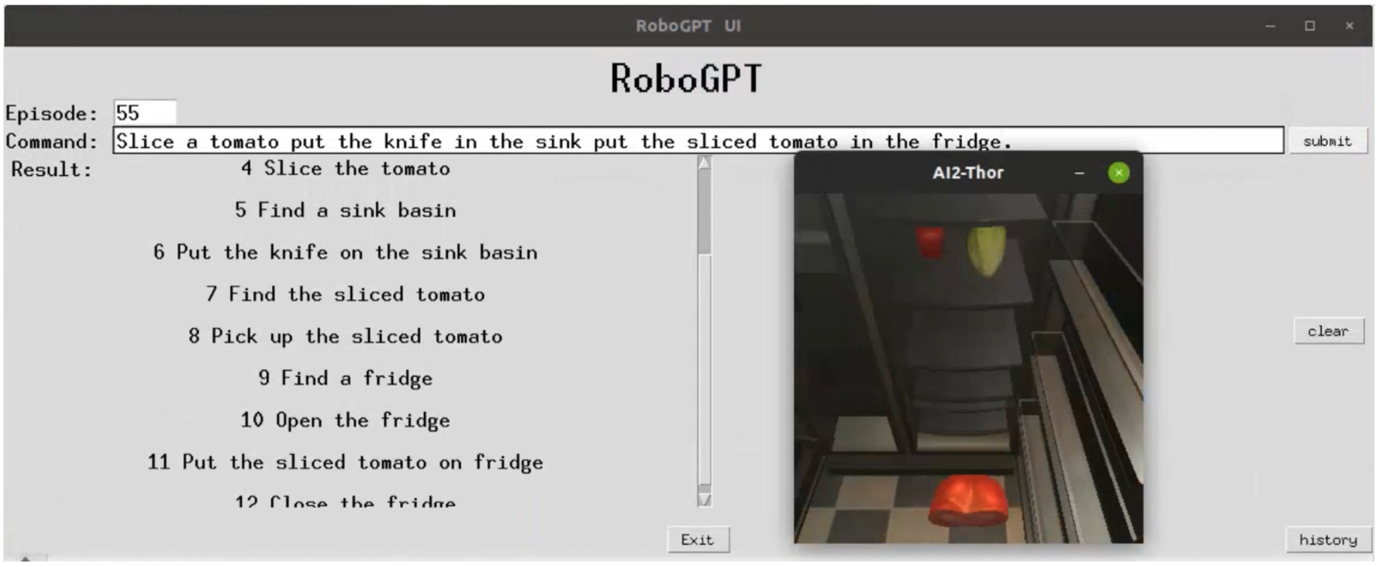

Instruction: Get a towel out of the cabinet to soak it, then put it on the toilet
Prompter
Towel
Mission Fail
(Find, Towel)
(Find, Towel)
(Find, Towel)
(Find, Towel)
(Find, Towel)
Prompter: (Find, Towel), (Pickup, Towel), (Put, SinkBasin), (Wash, Towel), (Pickup, Towel), (Put, Toilet)
Time
RoboGPT
Cabinet
Towel
Cabinet
SinkBasin
Toilet
Mission Success
(Open, Cabinet)
(Pickup, Towel)
(Close, Cabinet)
(Put, SinkBasin)
(Put, Toilet)
RoboGPT: (Open, Cabinet), (Pickup, Towel), (Close, Cabinet), (Put, SinkBasin), (Wash, Towel), (Pickup, Towel), (Put, Toilet)

Fig. 4. Planning of the task with the invisible object in containers. RoboPlanner possesses the capability to comprehend object relationships, hence facilitating the accomplishment of tasks involving the presence of a target object within an enclosed area.



```
Episode: 55
Command: Slice a tomato put the knife in the sink put the sliced tomato in the fridge.
Result:
4 Slice the tomato
5 Find a sink basin
6 Put the knife on the sink basin
7 Find the sliced tomato
8 Pick up the sliced tomato
9 Find a fridge
10 Open the fridge
11 Put the sliced tomato on fridge
12 Close the fridge
```

Fig. 5. Verification system. The instruction task is ‘*Slice a tomato, put the knife in the sink, and put the sliced tomato in the fridge*’. The planning of RoboGPT is ‘*Find a knife, pick up the knife, find a tomato, slice the tomato, find a sink, put the knife in the sink, find the sliced tomato, pick up the sliced tomato, find a fridge, open the fridge, put the sliced tomato in the fridge, close the fridge*’.

RoboGPT without RoboPlanner has the highest PLWSR and PLWGC than others, even for those using RoboGPT. The reason is that Prompter combines some objects with similar meanings (such as desk lamp and floor lamp, butter knife, and knife) based on rules. Although this can significantly improve the execution efficiency, it is difficult to transfer to complex real scenes. Contrary to Prompter, RoboGPT uses Re-Plan to address the issue of target objects but only after gathering a significant amount of environmental data. Although this strategy is more useful in the actual world, it can also decrease the robot's execution efficiency, resulting in lower scores for PLWSR and PLWGC.

E. Experimental Results on ALFRED test tasks

To ensure a fair comparison of performance across all methods, we conduct our testing on the third-party platform ALFRED. We evaluate our approach on the ALFRED test tasks and achieve SOTA results in both seen and unseen environments (shown in Table II). Compared to methods requiring low-level step-by-step instructions [8], [41] and the method requiring ChatGPT planning [10], RoboGPT achieves the best performance with at least 10.4% improvement of SR in Test Unseen. This demonstrates the planning ability of RoboGPT even beyond the step-by-step guidance of an expert or ChatGPT. In reality, the limitations of the ALFRED dataset

TABLE II
PERFORMANCE COMPARISON ON ALFRED TEST TASKS.

Methods	Tests Seen		Tests Unseen	
	GC	SR	GC	SR
High+Low-level instructions				
HLSM [40]	41.21	29.94	30.31	20.27
LGS-RPA [41]	48.66	40.05	45.24	35.41
ET [7]	45.44	38.42	18.56	8.57
MCR-Agent [8]	-	30.13	-	17.04
M-TRACK [42]	22.60	16.29	33.35	24.79
LEBP [43]	36.79	28.30	36.33	28.97
Prompter [5]	60.22	51.17	56.57	45.32
CAPEAM [38]	60.98	52.58	61.40	50.36
RoboGPT (Ours)	**69.24**	**61.77**	**70.53**	**60.76**
High level instruction only				
EPA [44]	44.14	39.96	39.54	36.07
HLSM [40]	35.79	25.11	27.24	16.29
FILM [6]	36.15	25.77	34.75	24.46
Prompter [5]	56.98	47.95	53.69	41.53
LLM-Planner [10]	26.77	18.20	23.37	16.42
OPEx [39]	54.27	43.51	53.82	41.27
RoboGPT (Ours)	**60.25**	**52.38**	**65.30**	**55.46**

hinder RoboGPT from achieving higher SR. ALFRED is a crowdsourced form of labeled data, with many irregularities in the labeling. By our count, more than 20% of the high-level instructions (instruction tasks) are ambiguous or even incorrect. Wrong instruction tasks lead to errors in our planning. This explains why there is a difference in SR of different methods between Table I and Table II. The Prompter initially categorizes tasks and then plans accordingly using templates. However, due to overfitting, it may mistakenly categorize certain erroneous instructions under the correct types, resulting in appropriate planning outcomes.

We develop a verification system based on the ALFRED simulation system. You can input arbitrary natural language commands and use RoboGPT to interact with the environment to accomplish the task. The demo is shown in Fig. 5. The instruction task is '*Slice a tomato, put the knife in the sink and put the sliced tomato in the fridge*', and RoboPlanner plans '*Find a knife, pick up the knife, find a tomato, slice the tomato, find a sink, put the knife in the sink, find the sliced tomato, pick up the sliced tomato, find a fridge, open the fridge, put the sliced tomato in the fridge, close the fridge*'.

V. CONCLUSIONS

In this paper, we design a RoboGPT for solving daily instruction following tasks with long-term decisions. The planning module RoboPlanner enhances and fine-tunes Llama using the collected 67K robotic dataset to integrate the world knowledge of LLMs with the expert knowledge of robots, which can understand the prefix context, object quantities, object dependencies, and the tasks with long-term decisions, handling most of daily tasks. Roboskill with an accurate perception model FastSAM is developed, resulting in improved navigation and manipulation abilities. Additionally, the designed Re-Plan adapts the planning to the environment, mitigating the nomenclature diversity problem. The paper provides a well-generalized method for daily instruction following tasks with long-term decisions in robotics. Future improvements may focus on multi-modal embodied planning and manipulation.

REFERENCES

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia *et al.*, "RT-2: vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 2165–2183.

[2] Y. Lu, Y. Chen, D. Zhao, and D. Li, "Mgrl: Graph neural network based inference in a markov network with reinforcement learning for visual navigation," *Neurocomputing*, vol. 421, pp. 140–150, 2021.

[3] H. Li, Y. Zhang, H. Wen, Y. Zhu, and D. Zhao, "Stabilizing diffusion model for robotic control with dynamic programming and transition feasibility," *IEEE Transactions on Artificial Intelligence*, pp. 1–11, 2024.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn *et al.*, "RT-1: robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[5] Y. Inoue and H. Ohashi, "Prompter: Utilizing large language model prompting for a data efficient embodied instruction following," *arXiv preprint arXiv:2211.03267*, 2022.

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

[7] A. Pashevich, C. Schmid, and C. Sun, "Episodic transformer for vision-and-language navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2021, pp. 15 942–15 952.

[8] S. Bhambri, B. Kim, and J. Choi, "Multi-level compositional reasoning for interactive instruction following," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, 2023, pp. 223–231.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[10] C. H. Song, B. M. Sadler, J. Wu, W.-L. Chao, C. Washington, and Y. Su, "LLM-Planner: few-shot grounded planning for embodied agents with large language models," in *2023 IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2986–2997.

[11] A. Irpan, A. Herzog, A. T. Toshev, A. Zeng, A. Brohan, B. A. Ichter *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," 2022.

[12] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[13] N. Xu, S. Masling, M. Du, G. Campagna, L. Heck, J. Landay, and M. Lam, "Grounding open-domain instructions to automate web support tasks." in *Proceedings of the 2021 Conference of the North American Chapter of*, 2021, pp. 1022–1032.

[14] S. Wen, X. Lv, F. R. Yu, and S. Gong, "Vision-and-language navigation based on cross-modal feature fusion in indoor environment," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 1, pp. 3–15, 2023.

[15] 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," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[16] G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[17] J. Segovia-Aguas, S. Jiménez, and A. Jonsson, "Generalized planning as heuristic search," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 569–577.

[18] Y. Ding, X. Zhang, X. Zhan, and S. Zhang, "Task-motion planning for safe and efficient urban driving," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 2119–2125.

[19] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2902–2908.

[20] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the international conference on automated planning and scheduling*, vol. 30, 2020, pp. 440–448.

[21] M. Liu, Y. Zhu, Y. Chen, and D. Zhao, "Enhancing reinforcement learning via transformer-based state predictive representations," *IEEE Transactions on Artificial Intelligence*, pp. 1–12, 2024.

[22] A. Zeng, M. Attarian, B. Ichter, K. M. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. S. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence, "Socratic models: Composing zero-shot multimodal reasoning with language," in *The Eleventh International Conference on Learning Representations*, 2023.

[23] 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*. IEEE, 2023, pp. 11 523–11 530.

[24] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," *arXiv preprint arXiv:2303.06247*, 2023.

[25] Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang, "Mm-react: Prompting chatgpt for multimodal reasoning and action," *arXiv preprint arXiv:2303.11381*, 2023.

[26] S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," *IEEE Access*, 2024.

[27] 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 *IEEE International Conference on Robotics and Automation*, 2023, pp. 9493–9500.

[28] 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*, 2022, pp. 9118–9147.

[29] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li, "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," *arXiv preprint arXiv:2305.11176*, 2023.

[30] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 13 484–13 508.

[31] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Alpaca: A strong, replicable instruction-following model," *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, vol. 3, no. 6, p. 7, 2023.

[32] J. Ji, S. Li, X. Liao, and F. Zhang, "Semantic segmentation based on spatial pyramid pooling and multilayer feature fusion," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 3, pp. 1524–1535, 2023.

[33] Y. Lu, Y. Chen, D. Zhao, B. Liu, Z. Lai, and J. Chen, "Cnn-g: Convolutional neural network combined with graph for image segmentation with theoretical analysis," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 3, pp. 631–644, 2021.

[34] Y. Chen, H. Li, R. Gao, and D. Zhao, "Boost 3-d object detection via point clouds segmentation and fused 3-d giou-l1 loss," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 762–773, 2022.

[35] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv preprint arXiv:2306.12156*, 2023.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351: 234–241, 2015.

[37] 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.

[38] B. Kim, J. Kim, Y. Kim, C. Min, and J. Choi, "Context-aware planning and environment-aware memory for instruction following embodied agents," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2023, pp. 10 936–10 946.

[39] H. Shi, Z. Sun, X. Yuan, M.-A. Côté, and B. Liu, "Opex: A component-wise analysis of llm-centric agents in embodied instruction following," *arXiv preprint arXiv:2403.03017*, 2024.

[40] V. Blukis, C. Paxton, D. Fox, A. Garg, and Y. Artzi, "A persistent spatial semantic representation for high-level natural language instruction execution," in *Conference on Robot Learning*, 2022, pp. 706–717.

[41] M. Murray and M. Cakmak, "Following natural language instructions for household tasks with landmark guided search and reinforced pose adjustment," *IEEE Robotics Autom. Lett.*, vol. 7, no. 3, pp. 6870–6877, 2022.

[42] C. H. Song, J. Kil, T.-Y. Pan, B. M. Sadler, W.-L. Chao, and Y. Su, "One step at a time: Long-horizon vision-and-language navigation with milestones," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 482–15 491.

[43] H. Liu, Y. Liu, H. He, and H. Yang, "Lebp–language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents," *arXiv preprint arXiv:2203.04637*, 2022.

[44] X. Liu, H. Palacios, and C. Muise, "A planning based neural-symbolic approach for embodied instruction following," *Interactions*, vol. 9, no. 8, pp. 17–17, 2022.