



Household Task Planning with Multi-Objects State and Relationship Using Large Language Models Based Preconditions Verification

Jin Aoyama^{1,2}, Sudesna Chakraborty¹ ^a, Takeshi Morita^{1,2} ^b,
Shusaku Egami², Takanori Ugai^{2,3} and Ken Fukuda²

¹*Aoyama Gakuin University, Kanagawa, Japan*

²*National Institute of Advanced Industrial Science and Technology (AIST), Kotoku, Japan*

³*Fujitsu Limited, Kanagawa, Japan*

Keywords: Task Planning, Large Language Model, Natural Language Processing, Embodied AI, Simulation.

Abstract: We propose a novel approach to household task planning that leverages Large Language Models (LLMs) to comprehend and consider environmental states. Unlike previous methods that depend primarily on common-sense reasoning or visual inputs, our approach focuses on understanding object states and relationships within the environment. To evaluate the capability, we developed a specialized dataset of household tasks that specifically tests LLMs' ability to reason about object states, identifiers, and relationships. Our method combines simulator-derived environmental state information with an LLM-based planning to generate executable action sequences. A key feature in our system is the LLM-driven verification mechanism that assesses whether environmental preconditions are met before each action executes, automatically reformulating action steps when prerequisites are not satisfied. Experimental results using GPT-4o demonstrate strong performance, achieving 89.4% success rate on state change tasks and 81.6% on placement tasks. Ablation studies confirm the precondition check's significant contribution to task success. This study establishes both a new methodology for embodied AI reasoning and a benchmark for future work in environment-aware task planning.

1 INTRODUCTION


Advancement in the field of artificial intelligence (AI) has led to growing demand for Embodied AI agents capable of performing complex daily tasks. Embodied AI (Duan et al., 2022) involves training agents capable of interacting and performing complicated tasks using various objects in real and virtual settings. For example, tasks like “the robot performs household tasks (e.g., cleaning, laundry) according to human instructions” and “the robot finds particular objects in the environment and provides guidance.” Within this field, there is an increasing interest in leveraging Large Language Models (LLMs) to enable agents to generate action plans for executing tasks based on natural language instructions (Huang et al., 2022; Ahn et al., 2022; Raman et al., 2024; Lin et al., 2023; Yoneda et al., 2024). These studies have demonstrated that LLMs possess crucial com-


mon sense knowledge essential for executing tasks in daily life.

Here, we propose utilizing LLMs to interpret the home environment by recognizing object states and the relationships between objects. This information is then used to generate suitable action plans for completing tasks based on the contextual understanding of the environment.

Existing datasets (Puig et al., 2018; Shridhar et al., 2020) used in previous studies consist of two types of tasks. The first comprises tasks that can be accomplished using common sense knowledge of LLMs without a comprehensive understanding of the environmental setting. The second type is made up of task difficult to accomplish without knowledge of objects in the environment, but are achievable without detailed knowledge of object states, identification numbers, or inter-object relationships.

For instance, to execute a task like “Turn on the TV,” a general solution would involve generating an action plan to approach the television and switch it on. However, this task targets a single object and implic-

^a  <https://orcid.org/0000-0002-3963-1761>

^b  <https://orcid.org/0000-0001-8963-2562>

itly assumes that the object (the TV) is initially powered off. Consequently, LLMs can generate the action plan based on the implicit knowledge that the power is off without requiring object identification. Conversely, for a task like “Turn on all light switches,” it would be challenging to generate an appropriate action plan without knowing the number of lights in the setting and their respective states (on or off). Current task settings lack challenging scenarios that require a thorough understanding of the home environment to be resolved effectively.

The current study uses VirtualHome (VH) (Puig et al., 2018), a multi-agent platform to simulate activities in a household, to create a dataset of household tasks that are challenging to solve without an in-depth understanding of the home environment. Two tasks were prepared for this study: a state change task aimed at bringing multiple objects into a specific state, and a placement task aimed at moving multiple objects to specific locations. Rather than processing visual information, our study focuses on leveraging LLMs in understanding the home environment to generate action plans for the agent.

Our proposed method generates appropriate action plans to execute tasks step-by-step using LLMs and home environment knowledge extracted from VH. Additionally, the method uses LLMs to verify whether the preconditions for executing the generated actions are met. If not, action steps are regenerated based on the environmental state extracted by LLMs. Here, an action plan refers to a plan that sequences a series of actions necessary to accomplish a task, and an action step refers to each action that constitutes the action plan.

To evaluate our method, we used a custom-made dataset of household tasks to assess the success rates of tasks performance. The employment of GPT-4o as LLM showed a success rate of 89.4% for state change tasks and 81.6% for placement tasks. Furthermore, an ablation study demonstrated that the verification of the preconditions and regenerating actions improved task success rates.

The contributions of this study can be summarized as:

1. The creation of a household task dataset that necessitates a comprehensive understanding of the home environment for effective task execution.
2. The proposal of a method that employs LLMs to recognize the home environment and generate suitable action plans for task execution based on their understanding of this environment.

The rest of this paper is organized as follows: Section 2 outlines the task setting and the dataset used; Section 3 discusses related works, including Task

Planning with LLM and HouseHold Task Dataset; Section 4 describes the proposed methodology in detail; Section 5 reports the evaluation experiments, including the evaluation dataset, methods, indices, results, discussion, and limitations; and finally, Section 6 concludes the paper.

2 TASK DESIGN AND DATASET

This section describes the household tasks that requires a comprehensive understanding of the home environment and outlines the dataset necessary for their evaluation. Home environment knowledge encompasses detailed information about objects within the home environment, including their current states, unique identification numbers, and the relationships between objects.

The following requirements specify the dataset parameters needed to evaluate these environment-dependent household tasks:

- All tasks can only be completed by understanding the home environment.
- All tasks require identifying the object’s identification number that is necessary to clarify which specific objects should be interacted with.
- The dataset should include tasks that cannot be completed by assuming the object’s initial state based on common sense. For instance, in a task like “Turn on the light,” it is commonly assumed that the light’s initial state is “OFF.” However, the light may already be “ON.” Therefore, understanding the actual state of the home environment, rather than relying on common sense assumptions, is crucial for task completion.
- The goal of each task should be uniquely defined. For example, a household task like “Clean up” is too abstract and difficult to evaluate automatically, as there are multiple ways to achieve the task’s goal. To address this, tasks need to be designed with specific, well-defined goal conditions to achieve consistency and accuracy in evaluation.
- The dataset should be constructed with the assumption that environmental information is provided accurately, without requiring reliance on visual processing for its acquisition. This approach aligns with the study’s focus on enabling LLMs to recognize the environment and generate action plans based on pre-existing, accurate data, rather than examining the process of information collection.

The commonly used datasets in previous studies, do not meet the necessary criteria for evaluation.

Therefore, we set up two types of tasks that meet the necessary criteria for evaluation. The first is a state change task to bring multiple objects into a specific state. The second is a placement task to move multiple objects to specific locations.

3 RELATED WORKS

3.1 Task Planning with LLM

Recently, there has been growing interest in using Large Language Models (LLMs) to enable agents to generate action plans for task execution based on natural language instructions. The advantages of utilizing LLMs are as follows:

- Agents are increasingly able to interpret instructions at various levels of abstraction, allowing for contextually accurate understanding.
- Agents can achieve high performance without needing large amount of training data, enabling more efficient learning with reduced data.
- Planning quality improves by utilizing broad common sense knowledge and advanced reasoning, allowing agents to generate suitable action plans for complex tasks.

The following studies (Huang et al., 2022; Ahn et al., 2022; Raman et al., 2024; Lin et al., 2023; Yoneda et al., 2024) demonstrated the possibility of generating appropriate action plans from natural language instructions to accomplish the task using LLMs.

Huang et al. (Huang et al., 2022) used GPT-3 (Brown et al., 2020) and BERT (Devlin et al., 2019) to generate action sequences from abstract natural language descriptions of task like “Make breakfast.” Their method employed GPT-3 for planning and generating the next action step to accomplish the task, then employed BERT to convert the action step into the action command that can be executed in the simulator. Subsequent iterations incorporated the previous action step along with the task description for LLM-based planning.

Raman et al. (Raman et al., 2024) advanced the field of automated planning by enhancing Huang et al.’s framework for generating corrective actions in response to precondition errors. Their innovative approach leverages simulator-based error feedback and few-shot reasoning to significantly improve action generation quality. Through this methodology, embodied agents demonstrate markedly expanded task execution capabilities compared to existing approaches (Huang et al., 2022; Ahn et al., 2022)

while maintaining semantic integrity and reducing the need for repeated prompting.

The approach by Huang et al. leveraged the commonsense knowledge of LLMs to perform household tasks without incorporating the home environment knowledge. In contrast, we propose an approach that utilizes the home environment knowledge to generate action plans. In addition, the approach by Reman et al. generates corrective actions to resolve precondition errors by leveraging error feedback from the simulator. On the other hand, we propose an approach that uses the home environment knowledge to predict errors before executing the action.

A significant challenge in LLM-based planning for home environments lies in developing robust mechanisms for environmental state estimation. The following studies (Lin et al., 2023; Yoneda et al., 2024) address this challenge.

Lin et al. (Lin et al., 2023) present an approach that leverages tabular environmental data for step-by-step action generation. Their method processes simulator-derived data tables containing critical environmental information including object states, spatial coordinates, and relational properties to produce detailed, executable action sequences.

Yoneda et al. (Yoneda et al., 2024) propose a framework in which LLMs are prompted to maintain the estimate of the state, often unobservable, and track its transition as new actions are taken. The framework continuously estimates states by performing multi-step inference based on the estimated states. It also generates executable actions based on the estimated state from natural language instructions.

Unlike Lin et al.’s approach using comprehensive tabular environmental data, our method intentionally constrains environmental knowledge and presents it to the LLM in natural language form. While Yoneda et al. emphasize LLM-based environmental state estimation, our approach instead relies on direct simulator feedback to maintain current environmental knowledge, updating in real-time as conditions change.

3.2 Household Task Dataset

The VH dataset (Puig et al., 2018), ALFRED (Shridhar et al., 2020), and HAZARD (Zhou et al., 2024) are designed for research focused on generating action plans from natural language instructions to perform household tasks.

The VH dataset is collected to train the robot to perform household activities. It includes common home activities (e.g., preparing coffee), multiple natural language descriptions of how each activity can be

carried out, and the corresponding action sequences for the agent to execute based on each description. In studies using the VH dataset, evaluations typically rely on an index based on the Longest Common Subsequence (LCS) between the correct and generated action sequences. However, when a robot performs household tasks, its actions should adapt to the situation, even when given the same instructions. Currently, no dataset or evaluation metric accounts for this situational variability.

ALFRED is a benchmark designed to learn how to map natural language instructions and egocentric vision to action sequences for completing household tasks. The ALFRED dataset contains language directives linked to expert demonstration episodes, with each directive including a goal instruction and a set of step-by-step instructions. These expert demonstration can be replayed deterministically in the AI2-THOR 2.0 simulator (Kolve et al., 2017). The dataset includes seven tasks, ranging from simple placement tasks (e.g., putting an apple on a table) to more complex tasks with additional steps (e.g., putting a microwaved potato on a countertop). ALFRED evaluates performance using two metrics: Task Success and Goal-Condition Success. Task Success is a binary measure (1 or 0) that indicates whether the object positions and state changes at the end of the action sequence match the task’s goal conditions. Goal-Condition Success is the ratio of successfully completed goal conditions at the end of an episode to the total conditions required to complete the task.

The commonly used these datasets in previous studies, do not meet the necessary criteria for evaluation referred to Section 2.

First, the tasks in these datasets do not require identifying specific objects by their unique identifiers. Second, they do not include tasks that cannot be solved by simply estimating the object’s initial state using commonsense. Third, the VH dataset lacks clearly defined task goals, making it difficult to evaluate the performance automatically. Lastly, the ALFRED dataset relies on visual data processing to acquire environmental information, which is not aligned with the focus of this study.

4 METHODS

Here we outlines the proposed method, which consists of six key components. Using LLMs and home environment knowledge extracted from a simulator, the method generates detailed, step-by-step action plans to execute tasks. Additionally, LLMs are used to verify whether the preconditions for each gener-

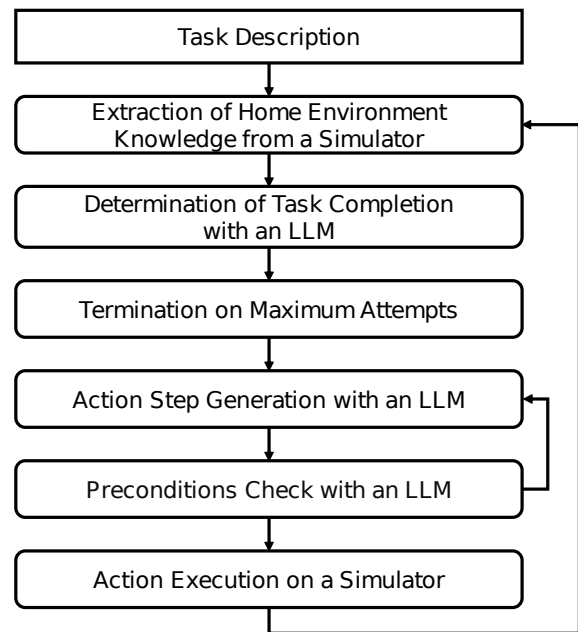


Figure 1: Configuration of the Proposed Method.

ated action are met. If the preconditions are not satisfied, the method regenerates the action steps based on the current environmental state, as determined by the LLM. The structure of the proposed method is illustrated in Figure 1.

4.1 Extraction of Home Environment Knowledge from a Simulator

In Figure 1 “Extraction of Home Environment Knowledge from a Simulator” gathers knowledge about the home environment from a simulator and converts it into natural language texts. This extracted knowledge is limited to information about the agent and the target objects specified in the task description.

The agent’s knowledge includes its relationship with objects or rooms within the environment. Target object knowledge incorporates the states of those objects and their relationship with other objects or rooms in the environment. Additionally, when multiple objects of the same type exist in the environment, it is necessary to distinguish them. For this purpose, object identification numbers are provided.

For example, in the task “Turn on all lights,” the method describes the state and location of each light in the environment.

4.2 Determination of Task Completion with an LLM

In Figure 1, the component “Determination of Task Completion with an LLM” uses an LLM, along with the home environment knowledge extracted from the simulator, to assess whether a task has been completed in the current status.

The prompt template that assesses whether a task has been complete in the current status is shown in Figure 2. The prompt begins with the home environment knowledge, followed by specific instructions regarding the task completion determination. In this template, the red portions of the template are placeholders to be filled with relevant information. If the LLM outputs “End”, the task is complete, and the system terminates. If it outputs “Continue,” the system proceeds to the next step.

For example, in the task “Turn on all lights,” the goal is for the LLM to output “End” once all lights in the environment are in the “ON” state.

```
The current states in the home are as follows:
[ Home Environmental Knowledge ]

If the following task has already completed based on the current states,
output "End"; otherwise, output "Continue."
Task: [ Task Description ]
```

Figure 2: Prompt template used in “Determination of Task Completion with an LLM”.

4.3 Termination on Maximum Attempts

In Figure 1, the “Termination on Maximum Attempts” component terminates the system when the set maximum number of attempts is reached. The number of attempts is added after passing this component. This component is to prevent infinite loops in the system.

4.4 Action Step Generation with an LLM

In Figure 1, the “Action Step Generation with an LLM” component utilizes an LLM, and the home environment knowledge, to develop a detailed, step-by-step action plan for completing the task.

The prompt template that generates step-by-step action plan for completing the task, shown in Figure 3, consists of the following elements:

- The role of the LLM.
- The actions that can be performed in the simulator and the required output format.
- An example task and a corresponding action plan.

- Home environment knowledge described in natural language.
- The final instruction, which includes the task description and the action execution history.

These elements are provided to the LLM as a prompt to generate the next action required to accomplish the task.

For example, in the task “Turn on all lights,” the LLM generates the next action needed to turn on any lights that are still “OFF.”

```
You need to generate a next action step for completing a household task.

[ Allowed Actions and Output Format ]

[ Example Task and Action Plan ]

The current states in the home are as follows:
[ Home Environmental Knowledge ]

Generate an only next action step to complete the following task and
output only that.
Task: [ Task Description ]
Step1:
```

Figure 3: Prompt template used in “Action Step Generation with an LLM”.

4.5 Preconditions Check with an LLM

In Figure 1, the “Precondition Check with an LLM” is a novel feature introduced to ensure that the current state of the environment satisfies the preconditions necessary to execute the generated action step, using an LLM. The sequence of steps for the “Precondition Check with an LLM” is shown in the pseudo code of Algorithm 1.

First, the system checks whether the preconditions for action execution are met. This is done by providing a prompt that includes the home environment knowledge and the required preconditions for the action. The template for this prompt is shown in Figure 4.

If the LLM outputs “No,” indicating the preconditions are not satisfied, it identifies the unmet preconditions using another prompt, shown in Figure 5. At this stage, the LLM is assumed to reference prior conversation history. The system then regenerates the action step, considering the previously generated action and the unmet preconditions.

For instance, if the action “switch on” is generated for a “light” that is already “ON,” the precondition that the light must be in the “OFF” state is not satisfied. In this case, the LLM would respond with “No,” indicating the unsatisfied precondition, and the action step would be adjusted accordingly.

Input : $actionStep, preconds, currentStates$
Output: $executionActionStep$

- 1 $executionActionStep \leftarrow actionStep$;
- 2 $precondsCheckPrompt \leftarrow CreatePrecondsCheckPrompt(preconds, currentStates)$;
- 3 $llmOutput \leftarrow Llm(precondsCheckPrompt)$;
- 4 **if** $llmOutput = \text{"No"}$ **then**
- 5 $extractionUnmetPrecondsPrompt \leftarrow CreateExtractionUnmetPrecondsPrompt(preconds)$;
- 6 $unmetPreconds \leftarrow Llm(extractionUnmetPrecondsPrompt)$;
- 7 $executionActionStep \leftarrow ActionStepGeneration(actionStep, unmetPreconds)$;
- end**
- 8 **return** $executionActionStep$

Algorithm 1: Preconditions Check with an LLM.

The current states in the home are as follows:
[Home Environmental Knowledge]

If the current states satisfies the preconditions, output "Yes"; otherwise, output "No".
Preconditions:
[Preconditions]

Figure 4: Prompt template for precondition check.

Which the preconditions are not satisfied? Output only that.
Preconditions:
[Preconditions]

Figure 5: Prompt template for extraction unmet preconditions.

4.6 Action Execution on a Simulator

In Figure 1, “Action Execution on a Simulator” uses the action generated by “Action Step Generation with an LLM” to simulate the action in the virtual environment. After execution, the action is added to the prompt as part of the action history.

Once the simulation is completed, the process begins from “Extraction of Home Environment Knowledge”. It continues until the LLM outputs “End” in “Determination of Task Completion with an LLM” stage or the maximum number of attempts is reached in “Termination on Maximum Attempts” stage.

4.7 Implementation

The functional requirements for the simulator used to implement the proposed method are as follows:

- The simulator must provide full access to environmental data, including the states of objects, relationships between objects, and interactions between the agent and objects.
- The simulator must support higher-level commands, allowing the agent to move and interact with objects by specifying the object and its identification number, instead of relying on detailed actions like moving, rotating, or manipulating the arm.

- The simulator must provide the preconditions required for executing actions.

In this study, the proposed method is implemented using VirtualHome v2.3 (VH) (Puig et al., 2018) as the simulator satisfying these requirements. VH is unique in providing high-level commands for agent interaction, unlike many other simulators that rely on detailed, low-level actions. In addition, VH includes seven distinct scenes (houses), each with four to five rooms, allowing for experiments across multiple scenes with various rooms and objects.

4.7.1 VirtualHome Simulator

VH is a simulator designed to model activities in a virtual household environment. The VH environment is structured as a graph in JSON format, as shown in part in Listing 1. The objects, rooms, and agents in the environment are listed under the “nodes” key, which contains semantic data such as identification numbers, spatial coordinates, and object states. The relationships between objects are defined under the “edges” key, specifying the connections between objects through their identification numbers listed in the “nodes” key.

In VH, agents are controlled based on action scripts consisting of action steps. Each step includes the action the agent performs, the object involved, and the object’s ID (e.g., [WALK] $\langle livingroom \rangle$ (336)). Actions have predefined preconditions that specify the environmental conditions to be met before the action can be executed. Detailed preconditions for each action can be found in the VH documentation¹ and GitHub repository². For example, the preconditions for the action “switch on” are that the object must be “off” and the agent must be “close to the object.” VH offers two methods for executing simulations: one

¹<http://virtual-home.org/documentation/master/kb/actions.html>

²<https://github.com/xavierpuigf/virtualhome/tree/master/virtualhome/simulation>

Listing 1: environment graph.

```

{
  "nodes": [
    {
      "id": 1,
      "category": "Characters",
      "class_name": "character",
      "position": [5.26, 0.00, -7.86],
      "properties": [],
      "states": []
    }, {
      "id": 336,
      "category": "Rooms",
      "class_name": "livingroom",
      "position": [3.64, 0.00, -5.49],
      "properties": [],
      "states": []
    }
  ],
  "edges": [
    {
      "from_id": 1,
      "to_id": 336,
      "relation_type": "INSIDE"
    }
  ]
}

```

where object IDs are specified, and another where the system automatically searches for objects without the need for specific IDs.

4.7.2 Prompt Examples for VH

Examples of applying the prompt template that assesses whether a task has been complete in the current status shown in Figure 2 and the prompt template that generates step-by-step action plan for completing the task Figure 3 to the VH environment are shown in Figures 6 and 7.

```

The current states in the home are as follows:
The lightswitch (71) is ON and is INSIDE the bathroom (11).
The lightswitch (173) is ON and is INSIDE the bedroom (73).
The lightswitch (261) is ON and is INSIDE the kitchen (205).
The lightswitch (427) is OFF and is INSIDE the livingroom (335).
You are INSIDE the bathroom (11).

```

```

If the following task has already completed based on the current states,
output "End"; otherwise, output "Continue."
Task: Turn on all lightswitches

```

Figure 6: Example prompt for VH used in “Determination of Task Completion with an LLM”.

5 EXPERIMENTS

5.1 Overview of the Evaluation Experiment

We evaluated our proposed method’s efficacy using a dataset of environment-dependent household tasks,

You need to generate a next action step for completing a household task.

Allowed actions:
Walk, Grab, Switch on, Switch off, Open, Close, Put, Put in
Output Format:
[WALK] <Object> (ID)
[GRAB] <Object> (ID)
[SWITCHON] <Object> (ID)
[SWITCHOFF] <Object> (ID)
[OPEN] <Object> (ID)
[CLOSE] <Object> (ID)
[PUT] <Object1> (ID) <Object2> (ID)
[PUTIN] <Object1> (ID) <Object2> (ID)

Example Task: Turn on all tablelamps
Step1: [WALK] <tablelamp> (256)
Step2: [SWITCHON] <tablelamp> (256)
Step3: ...

The current states in the home are as follows:
The lightswitch (71) is ON and is INSIDE the bathroom (11).
The lightswitch (173) is ON and is INSIDE the bedroom (73).
The lightswitch (261) is ON and is INSIDE the kitchen (205).
The lightswitch (427) is OFF and is INSIDE the livingroom (335).
You are INSIDE the bathroom (11).

Generate an only next action step to complete the following task and output only that.
Task: Turn on all lightswitches
Step1:

Figure 7: Example prompt for VH used in “Action Step Generation with an LLM”.

measuring success rates for tasks requiring comprehension of the home environment. This experimental evaluation assessed how well the method performed when completing tasks that depend on understanding environmental context and conditions.

5.2 Evaluation Dataset

To assess our method, we developed a new dataset that satisfies the criteria outlined in Section 2.

5.2.1 State Change Task

These tasks require changing the states of objects in the environment, such as “Turn on all lights” or “Close all windows.” The target objects for these tasks include items with states, such as home appliances with switches and furniture with doors.

The task difficulty is divided into two categories: simple tasks involving a single object, like “Turn on the light,” and more complex tasks involving multiple objects, such as “Turn on all lights.”

We focused on designing the latter, more complex tasks. For each type of task description, we created the dataset with multiple patterns of initial object states. For instance, in the task “Turn on all lights,” the initial state could vary, some lights might be all “ON,” all “OFF,” or a mix of “ON” and “OFF.” Since the agent’s behavior must adapt to different initial conditions, completing these tasks without using detailed home environment knowledge to determine

which objects to interact with becomes more challenging.

5.2.2 Placement Task

These tasks involve placing objects in specific locations within the environment, such as “Put all apples in the fridge.” The targeted objects for these tasks include those that the robot can grasp and those that can receive other objects (receptacles).

Similar to state change tasks, placement tasks are divided into two difficulty levels: simple tasks, which involve placing a single object, like “Put an apple in the fridge,” and more complex tasks, which involve placing multiple objects like “Put all apples in the fridge.”

Here too, we focused on designing the more complex tasks. Some receptacles, like fridges, may have doors that can be either “OPEN” or “CLOSED.” the dataset includes two initial state patterns for the receptacles (“OPEN” or “CLOSED”). Before placing an object, the agent must infer whether the receptacle’s door needs to be opened using home environment knowledge. This design increases the difficulty, as the task cannot be completed successfully without such inference.

5.2.3 Dataset for VirtualHome

As described in Section 4.7, this study utilizes VH as the simulator. Therefore, the dataset was designed to be compatible with VH.

VH includes seven different scenes (houses), each containing various rooms and different objects within those spaces. Tasks were configured according to the unique characteristics of each scene.

In VH, there are four types of object states: “ON,” “OFF,” “Open,” and “CLOSED.” Considering object variations in VH, we constructed a dataset for the state change task by targeting seven types of objects with a power supply (e.g., lightswitch, tablelamp). Additionally, for the placement task, we limited the placement to movable objects classified as food or drink items (e.g., banana, milk), and the placement locations were selected based on their suitability for placing such items (e.g., kitchentable, fridge). The dataset was designed with twelve types of food and drink items and eight designated placement locations.

Examples of the dataset are shown in Listing 2 and 3. The dataset consists the task description, the scene in which the task takes place, the agent’s initial position, the initial states of the objects, the goal states of the objects, and the action script necessary for the agent to complete the task.

Listing 2: Example of State Change Task for VH.

```
{
  "task": "Turn on all lightswitches",
  "scene": 1,
  "initial_room": "kitchen",
  "initial_states": [
    {"id": 71, "states": ["ON"]},
    {"id": 173, "states": ["OFF"]},
    {"id": 261, "states": ["ON"]},
    {"id": 427, "states": ["OFF"]}
  ],
  "goal_states": [
    {"id": 71, "states": ["ON"]},
    {"id": 173, "states": ["ON"]},
    {"id": 261, "states": ["ON"]},
    {"id": 427, "states": ["ON"]}
  ],
  "action_scripts": [
    "[WALK] <lightswitch> (173)",
    "[SWITCHON] <lightswitch> (173)",
    "[WALK] <lightswitch> (427)",
    "[SWITCHON] <lightswitch> (427)"
  ]
}
```

5.3 Data Split

The state change task dataset contains 464 examples, while the placement task dataset has 154 examples. We split each dataset roughly in a 2:1 ratio between testing and sample. The sample data is used to generate action steps as examples.

For state change tasks, 312 examples were allocated for testing, and 152 for training. For placement tasks, 103 examples were used for testing, and 51 for training.

Each dataset treats tasks as distinct, even when they share the same task description, due to variations in the initial states of objects and VH scenes. Although the data split was random, we ensured that examples with the same task description were not shared between the sample and test sets. As a result, the split may slightly vary from the exact 2:1 ratio.

5.4 Evaluation Methods

To begin, the VH home environment is configured using the dataset’s scene and initial state. The proposed method then generates action plans based on the task description provided in the dataset.

We experiment with two approaches for prompting the LLM to generate action steps. The first method, “Single Prompt,” provides multiple components of the prompts described in Figure 3 to the LLM at once. The second method, “Multi Prompts,” gives the contents step-by-step. Furthermore, an ablation study is conducted to evaluate the effectiveness of the

Listing 3: Example of Placement Task for VH.

```

{
  "task": "Put all plums in the fridge",
  "scene": 4,
  "initial_room": "bedroom",
  "initial_states": [
    {"id": 103, "states": ["CLOSED"]}
  ],
  "goal_states": [
    {
      "from_id": 53,
      "to_id": 103,
      "relation_type": "INSIDE"
    },
    {
      "from_id": 54,
      "to_id": 103,
      "relation_type": "INSIDE"
    }
  ],
  "action_scripts": [
    "[WALK] <plum> (53)",
    "[GRAB] <plum> (53)",
    "[WALK] <fridge> (103)",
    "[OPEN] <fridge> (103)",
    "[PUTIN] <plum> (53) <fridge> (103)",
    "[WALK] <plum> (54)",
    "[GRAB] <plum> (54)",
    "[WALK] <fridge> (103)",
    "[PUTIN] <plum> (54) <fridge> (103)",
    "[CLOSE] <fridge> (103)"
  ]
}

```

“Preconditions Check with an LLM,” a critical feature of the proposed method. Ablation studies for other components were not performed, as removing them would cause the system to fail.

For task evaluation, a similar task description from the training dataset is selected as an example to help the LLM generate action steps. The task example chosen should closely resemble the input task description.

The maximum number of actions allowed to complete a task is set at twice the length of the example action plan from the training dataset. If the task isn’t completed within this limit, the system terminates the task, resulting in a failure.

If an action fails because its preconditions are not met, the task is also terminated and marked as a failure.

As a baseline, we use the method described by Huang et al. (Huang et al., 2022) that does not rely on home environment knowledge. This comparison highlights the importance of our dataset, which requires an understanding of the home environment for successful task completion. Since the baseline method does not allow generating action scripts with object IDs, simulation are run by automatically

searching for objects or rooms without specifying IDs.

For this evaluation, we used two types of LLMs: gpt-4o-mini (gpt-4o-mini-2024-07-18) and gpt-4o (gpt-4o-2024-08-06). The baseline method uses only gpt-4o. In both cases, the temperature was set to 0.0.

5.5 Evaluation Index

This study evaluates the proposed method using task success rates, commonly employed in prior research (Shridhar et al., 2020; Song et al., 2023). However, while previous evaluations did not consider object identification numbers, our evaluation includes verifying the correct matching of object IDs.

The task success rate is defined in Formula 1, as the ratio of successful tasks to the total number of tasks. A task is considered successful only if all goal conditions are met; otherwise, the task is considered as a failure.

We identify three types of failures. The first type occurs when an action cannot be successfully executed in the VH simulator. The second type of failure occurs when the maximum number of attempts is reached without completing the task. The third type happens when the task is incorrectly judged as complete, even though not all goal conditions have been satisfied. To better understand the causes of failure, we define failure rates for each type, represented in Formula 2, 3, and 4.

Each formula calculates the proportion of failures caused by one of the failure types described above, divided by the total number of tasks.

Additionally, we calculate the “AverageSteps” (the average number of action steps) using Formula 6. This metric helps assess the efficiency of the method by measuring how many actions were executed during task performance in the VH simulator.

5.6 Results

The results of comparing the proposed method with baseline methods using the state change task dataset are summarized in Table 1, while Table 2 shows the results for the placement task dataset.

In these Tables 1 and 2, the “Baseline” under the “Method” column refers to the method by Huang et al., the “Single Prompt” refers to the approach where a single prompt is provided to the LLM, and the “Multi Prompts” approach splits the prompt into multiple steps. “Single Prompt w/o PC” and “Multi Prompt w/o PC” represent the ablation study, where the “Preconditions Check with an LLM” was removed from the proposed method.

$$SuccessRate(SR) = \frac{SumOfSuccesses}{SumOfTasks} \quad (1)$$

$$ActionExecutionFailureRate(AEFR) = \frac{SumOfActionExecutionFailures}{SumOfTasks} \quad (2)$$

$$FailureRateOfReachingMaximumAttempts(FRRMA) = \frac{SumOfReachingMaximumAttempts}{SumOfTasks} \quad (3)$$

$$ErroneousTerminateFailureRate(ETFR) = \frac{SumOfErroneousTerminateFailures}{SumOfTasks} \quad (4)$$

$$FailureRate(FR) = AEFR + FRRMA + ETFR \quad (5)$$

$$AverageSteps = \frac{SumOfActionStepsForAllTasks}{SumOfTasks}. \quad (6)$$

Table 1: Results of State Change Task.

Method	LLM	SR	FR			Average Steps
			AEFR	FRRMA	ETFR	
Baseline (Huang et al., 2022)	gpt-4o	0.032	0.0	0.535	0.433	10.481
Single Prompt	gpt-4o-mini	0.625	0.026	0.231	0.119	4.747
	gpt-4o	0.788	0.016	0.173	0.022	4.381
Multi Prompts	gpt-4o-mini	0.750	0.045	0.183	0.022	4.917
	gpt-4o	0.894	0.019	0.000	0.087	3.391
Single Prompt w/o PC	gpt-4o-mini	0.615	0.003	0.253	0.128	4.872
	gpt-4o	0.760	0.016	0.189	0.035	4.449
Multi Prompts w/o PC	gpt-4o-mini	0.696	0.016	0.253	0.035	5.080
	gpt-4o	0.872	0.010	0.016	0.103	3.478

The results demonstrate that the proposed method achieves a higher success rate than the baseline. This highlights the baseline method’s difficulty in completing household tasks, as it generates action scripts without incorporating environmental knowledge. Moreover, comparing the proposed method’s Single Prompt and Multi Prompts strategies, it was found that dividing the prompt into multiple steps generally led to higher success rates, except when gpt-4o-mini was used for the placement task.

The ablation study results further indicate that applying “Preconditions Check with an LLM” generally improves the success rate. However, the opposite effect was observed when using gpt-4o-mini for the placement task.

A comparing between Tables 1 and 2 shows that the success rate for the placement tasks is generally lower than for the state change tasks. Additionally, the notable difference in results between gpt-4o-mini and gpt-4o for the placement task suggests that this task demands a higher level of reasoning ability.

5.7 Discussion

The comparison between the state change task and the placement task shows a lower success rate for the placement task. This difference may be due to

the increased difficulty for the LLM to recognize and reason about the spatial relationships between objects compared to simply identifying their states. The substantial performance gap between gpt-4o-mini and gpt-4o in the placement task further suggests that placement tasks demand more advanced reasoning abilities.

For the state change task, a high Failure Rate Reaching Maximum Attempts (FRRMA) within the overall Failure Rate (FR) indicates that most failures occur due to the system exceeding the maximum number of attempts allowed for action steps. This issue is likely tied to the process of “Action Step Generation with an LLM,” which relies on home environment knowledge. Often, failures stem from unnecessary interactions with objects that have already met their goal state or objects that don’t require manipulation, causing the system to take too many steps.

In the placement task, the high Erroneous Terminate Failure Rate (ETFR) within the overall FR underscores the importance of accurately assessing task completion. The failures are primarily tied to errors in the “Determination of Task Completion with an LLM,” that judge task completion based on home environment knowledge and the task description. This also explains why, in the placement task using gpt-4o-mini, the Multi Prompts method did not outperform

Table 2: Results of Placement Task.

Method	LLM	SR	FR			Average Steps
			AEFR	FRRMA	ETFR	
Baseline (Huang et al., 2022)	gpt-4o	0.000	0.010	0.573	0.417	15.398
Single Prompt	gpt-4o-mini	0.466	0.146	0.029	0.359	7.282
	gpt-4o	0.738	0.117	0.029	0.117	8.078
Multi Prompts	gpt-4o-mini	0.408	0.117	0.029	0.447	6.447
	gpt-4o	0.816	0.117	0.019	0.049	8.272
Single Prompt w/o PC	gpt-4o-mini	0.476	0.117	0.019	0.388	6.592
	gpt-4o	0.680	0.165	0.029	0.126	7.320
Multi Prompts w/o PC	gpt-4o-mini	0.437	0.126	0.010	0.427	6.136
	gpt-4o	0.796	0.117	0.029	0.058	7.767

the Single Prompt approach.

Regarding the ablation study on the state change task, the success rate improved by applying “Preconditions Check with an LLM,” primarily due to a reduction in FRRMA. For instance, during a task like turning on a power source, if the power source is already “ON,” the system might generate an unnecessary action to turn it on again. The preconditions check detects this and prevents redundant actions, leading to more efficient task execution. However, it should be noted that in the VH environment, actions like turning on a power source that’s already on do not result in execution errors, so the Action Execution Failure Rate (AEFR) was not significantly affected.

For the placement task, the impact of the preconditions check was more evident with gpt-4o, as it helped reduce AEFR in the Single Prompt method. However, no significant effect was observed for gpt-4o-mini, since failures were mainly tied to earlier stages in the “Action Step Generation with an LLM” before the preconditions check could be applied.

5.8 Limitations

Our dataset has several limitations. Although it introduces two types of tasks as an initial step towards a new approach, it is constrained to specific task types, as well as a limited range of object states and relationships. To create a more versatile and robust system, expanding the dataset to include a broader array of tasks is inevitable.

Limitations also exist with the language models (LLMs) used. This study relied on LLMs provided by OpenAI, and their performance depends heavily on the data they were trained on and the specific methodologies applied. Future research should experiment with and analyze other LLMs to assess their applicability to the proposed method.

Another limitation concerns how household environment knowledge is acquired. This study assumes that this knowledge of the household environment can

be accurately obtained from the simulator’s internal data. However, for real-world applications, the ability to recognize the environment using sensors or cameras would become a fundamental requirement.

Finally, there are limitations related to action execution. Currently, all actions are performed within a simulated environment, but in real-world scenarios, where an LLM’s outputs guide a robot’s actions, incorrect outputs could lead to malfunctions, accidents, or significant operational failures. This poses added challenges for the safe application of LLMs in robotics.

6 CONCLUSION

We proposed a novel approach to household task planning that leverages LLMs to comprehend and consider environmental states. Unlike previous methods that depend primarily on commonsense reasoning or visual inputs, our approach focused on understanding object states and relationships within the environment. To evaluate the capability, we developed a specialized dataset of household tasks that specifically tests LLMs’ ability to reason about object states, identifiers, and relationships.

To address these tasks, we proposed a method that combines simulator-derived environmental state information with an LLM-based planning to generate step-by-step action plans. Additionally, the method utilizes LLMs to verify preconditions before executing actions; if preconditions are not met, the actions are regenerated based on the updated environmental state, using LLMs.

We evaluated our method using the custom dataset of household tasks and measured task success rates. Employment of GPT-4o as the LLM showed a success rate of 89.4% for state change tasks and 81.6% for placement tasks. An ablation study demonstrated that verifying preconditions and regenerating actions contributed to higher success rates. However, when

using GPT-4o-mini for placement tasks, challenges emerged in recognizing and reasoning the home environment effectively.

Despite these results, the current dataset has limitation, focusing on specific tasks and object states. Future work should aim to expand the dataset to cover a broader range of household tasks, including tasks that require more complex object interactions (e.g., washing dishes, vacuuming floors), different object states (e.g., dirty or broken), and diverse environments (e.g., different room layouts and multiple floors). This expansion would enhance the generalizability of the proposed method and its applicability in real-world scenarios.

While this study utilized OpenAI's LLMs, exploring the performance of other LLMs, such as those trained specifically for embodied tasks or those with more comprehensive commonsense knowledge, is crucial. This could help identify LLMs that are better suited for household task planning and provide insights into the influence of LLM architectures and training data on task performance.

Currently, environmental knowledge is acquired using a simulator. In the future, methods should be developed to gather this information from real-world sensors like cameras, enabling robots to visually perceive their environment. This would improve adaptability to dynamic situations and allow for practical real-world applications.

Finally, ensuring the safety and reliability of generated action plans is crucial before installing robots in real-world environments. Future work should address on robust safety mechanisms, such as integrating safety constraints directly into LLM prompting, using safety modules to evaluate actions, or testing robot behavior in simulation before execution. These precautions are critical in preventing operational failures or accidents in real-world scenarios.

ACKNOWLEDGEMENTS

This paper is based on results obtained from projects, JPNP20006 and JPNP180013, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was partially supported by JSPS KAKENHI Grant Number 23K11221.

REFERENCES

Ahn, M. et al. (2022). Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*.

Brown, T. B. et al. (2020). Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Devlin, J. et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Duan, J. et al. (2022). A Survey of Embodied AI: From Simulators to Research Tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244.

Huang, W. et al. (2022). Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Kolve, E. et al. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Lin, B. Y. et al. (2023). On Grounded Planning for Embodied Tasks with Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13192–13200.

Puig, X. et al. (2018). VirtualHome: Simulating Household Activities via Programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Raman, S. S. et al. (2024). CAPE: Corrective Actions from Precondition Errors using Large Language Models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE.

Shridhar, M. et al. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Song, C. H. et al. (2023). LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Yoneda, T. et al. (2024). Statler: State-Maintaining Language Models for Embodied Reasoning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15083–15091. IEEE.

Zhou, Q. et al. (2024). HAZARD Challenge: Embodied Decision Making in Dynamically Changing Environments. *ArXiv*, abs/2401.12975.