Mitigating Cross-Modal Distraction and Ensuring Geometric Feasibility via Affordance-Guided, Self-Consistent MLLMs for Food Preparation Task Planning

Yu-Hong Shen[1‡] Chuan-Yu Wu[2‡] Yi-Ru Yang[1] Yen-Ling Tai[1] Yi-Ting Chen[1]

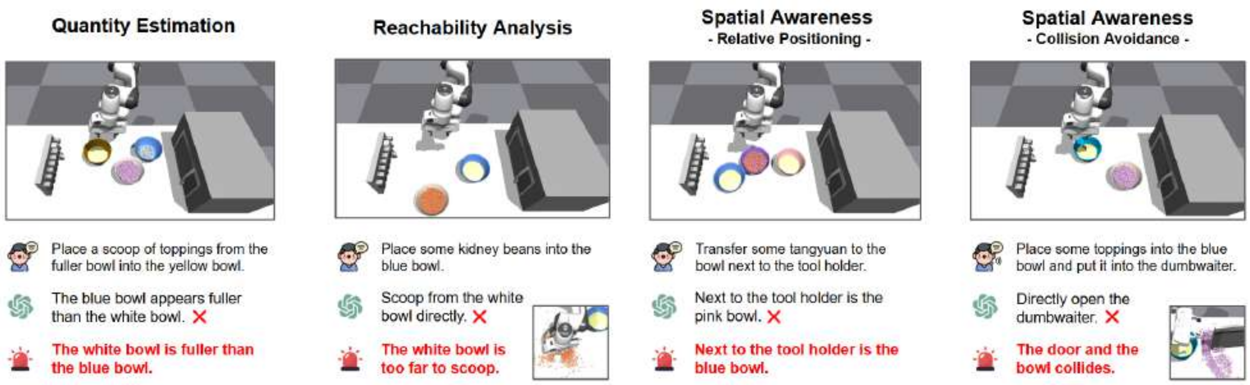

Fig. 1: We identify **four** categories of failures when using Multimodal Large Language Models (MLLM) with in-context learning for food preparation task planning. MLLMs fail to compare quantities between bowls, identify which bowls need repositioning before scooping, recognize spatial relationships between objects, and consider moving objects to avoid collisions. As a result, the robots may not follow the instructions properly and might even spill the bowls.

***Abstract*— We study Multimodal Large Language Models (MLLMs) with in-context learning for food preparation task planning. In this context, we identify two key challenges: cross-modal distraction and geometric feasibility. Cross-modal distraction occurs when the inclusion of visual input degrades the reasoning performance of a MLLM. Geometric feasibility refers to the ability of MLLMs to ensure that the selected skills are physically executable in the environment. To address these issues, we adapt Chain of Thought (CoT) with Self-Consistency to mitigate reasoning loss from cross-modal distractions and use affordance predictor as skill preconditions to guide MLLM on geometric feasibility. We construct a dataset to evaluate the ability of MLLMs on quantity estimation, reachability analysis, relative positioning and collision avoidance. We conducted a detailed evaluation to identify issues among different baselines and analyze the reasons for improvement, providing insights into each approach. Our method reaches a success rate of 76.7% on the entire dataset, showing a substantial improvement over the CoT baseline at 36.7%.**

I. INTRODUCTION

Imagine walking into a high-end tofu pudding shop where a service robot is tasked with preparing customized dessert orders. A table in front of the robot is set with bowls of tofu pudding and various toppings. Customers interact with the robot using natural language, specifying their preferences, such as ‘Place some mung beans onto my tofu pudding.’ To fulfill these requests, the robot must accurately interpret human instructions, ground the specified food items, and perform the corresponding manipulation skills to complete tofu pudding preparation.

Interpreting human instructions and converting them into the corresponding action sequences remains a challenge in robotics. Task planning [3], [4], [5], [6], [7], [8] decomposes a given goal specification into subtasks based on the available robot skill set. To handle goal specifications expressed in natural language, Large Language Models (LLMs) have been investigated for their ability to generate structured plans from textual input. More recently, Multimodal Large Language Models (MLLMs) have extended LLMs by incorporating additional modalities, such as visual inputs, enabling them to perceive and interpret the world.

In this work, we study MLLMs with in-context learning for food preparation task planning. Due to the difficulties in recovering from spillage in real-world settings and the lack of datasets in simulation that resemble our food preparation scenario, we construct our dataset in a simulated environment to evaluate the abilities of MLLMs. Specifically, we choose IsaacGym [9] as our simulator due to its high-fidelity in rendering a wide range of food items.

We identify **cross-modal distraction** and **geometric feasibility** as the main challenges to applying MLLMs for food preparation task planning. Reasoning across different modal-

‡ Equal Contribution.
[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu City, Taiwan stanley.shen2003.c,ben25000233.cs12,yling.cs12, ychen}@nycu.edu.tw
[2]Department of Computer Science and Information Engineering, National Taiwan University, Taipei City, Taiwan r13922a23@ntu.edu.tw

ities has been an active area of research in the community [10], [11]. However, challenges such as visual hallucination and illusion highlight that incorporating image inputs can sometimes produce unintended or misleading results [12], [13]. Our experiments reveal that for tasks not requiring visual input, adding images to the same prompt distracts the MLLM, degrading its reasoning performance. We term this phenomenon cross-modal distraction. For example, in tasks that require only semantic reasoning, incorporating an image observation causes MLLMs to generate infeasible skill sequences, overlook previous actions, or scoop twice despite being instructed to scoop only once.

Additionally, we find that MLLMs struggle with reasoning about geometric feasibility, which involves ensuring that the planned sequence of robot actions is physically executable within the spatial constraints of the environment. For example, in Figure 1, MLLM fails to recognize that there is a bowl in front of the dumbwaiter door and attempts to open the dumbwaiter directly.

A natural question arises: how can we address the limitations of MLLMs in food preparation task planning within an in-context learning setting?

Our approach uses Chain of Thought with self-consistency to counter performance drops from cross-modal distractions and leverages skill affordance to provide geometric feasibility information for MLLM. We study task planning under closed-loop setting, where the robot iteratively selects actions using the new image input. Chain of Thought prompting [20] helps MLLM to reason through problems step by step, leading to better responses. Self-Consistency Verification in MLLM checks whether the model generates reliable outputs across multiple runs, enabling it to detect anomalies and maintain stable performance. Skill affordance refers to the likelihood that a skill can be successfully executed given a specific environment state and acts as the action precondition for executing certain skills. We encode crucial information for identifying geometric feasibility as preconditions, enabling verification of whether the decisions made by MLLM are feasible. Our contributions are as follows:

1) We construct a dataset designed for task planning in food preparation, categorizing different scenarios to evaluate various robotic capabilities.
2) We adapt CoT with Self-Consistency to closed-loop task planning settings to combat reasoning loss caused by cross-modal distractions
3) We build up action preconditions with predicates to enable MLLM to take geometric feasibility into consideration.

II. RELATED WORK

A. Large Models in Robotics

As Large Language Models (LLMs) continue to advance, several studies [4], [5], [6], [7] demonstrate how LLMs can be leveraged to follow natural language instructions. However, it is crucial to bridge the information gap between textual input and environmental observations when using LLMs. For tasks that require context-specific judgments, perception models are needed to extract relevant information to supply the LLM as decision-making criteria. Since the reference information for natural language instructions can vary greatly across instances, exhaustively listing all possible references is both cumbersome and time-consuming.

MLLMs can process image input and extract relevant information through prompting. They are commonly used as success detectors or precondition checkers, or scene descriptors [6], [16], [17], [18], [19]. Most of them are used in the form of visual question answering. Most of them are used in the form of visual question answering, whereas we aim for MLLMs to autonomously extract and implicitly utilize this information for planning accurate action sequences.

B. Reasoning Ability of Large Models

There are many studies that explore ways to enhance the reasoning ability of large models. We focus on two categories, Chain of Thought (CoT) and Self-Consistency. CoT [20] guides LLMs to break down problems into step-by-step reasoning for better results. Several variants exist [21], [22], with zero-shot CoT [23] being the most relevant to our work. The main difference is that we combine the two-stage pipeline into a single prompt and we prompt the model to describe the goal the robot needs to achieve in the current iteration instead of using the magic prompt "Let's think step by step."

Self-Consistency methods employ a "sample and select" approach, where large models generate several possible answers and determine the final output with majority vote [24], [25]. We extend this concept to closed-loop task planning in robotics, using the output of previous iterations as samples to assess the LLM's consistency at the current iteration.

III. METHOD

A. Problem Formulation

The robot is equipped with a skill set Π consisting of various skills π. Given an instruction I, a skill set Π, current visual observation O_t, and the skills selected in previous iterations $\{\pi_i \mid i \in [1, t-1]\}$, a skill $\pi_t \in \Pi$ is selected to be executed in iteration t. This process loops until a special skill **DONE** is selected to indicate that the instruction I is satisfied.

B. Pipeline Overview

Our planning pipeline (Fig. 2) consists of three main stages: **1) MLLM Planning**, **2) Self-Consistency Verification**, and **3) Skill Affordance and Replanning**. In the first stage, we leverage Chain of Thought (CoT) [23] prompting to enhance the reasoning ability of MLLM. The latter two stages address the key challenges: cross-modal distraction and geometric feasibility. Self-Consistency Verification is designed to mitigate the degraded reasoning ability and stability caused by cross-modal distraction. Skill Affordance and Replanning focus on evaluating geometric feasibility, preventing infeasible skill executions through precondition checks, and recovering from infeasibility by utilizing structured feedback.

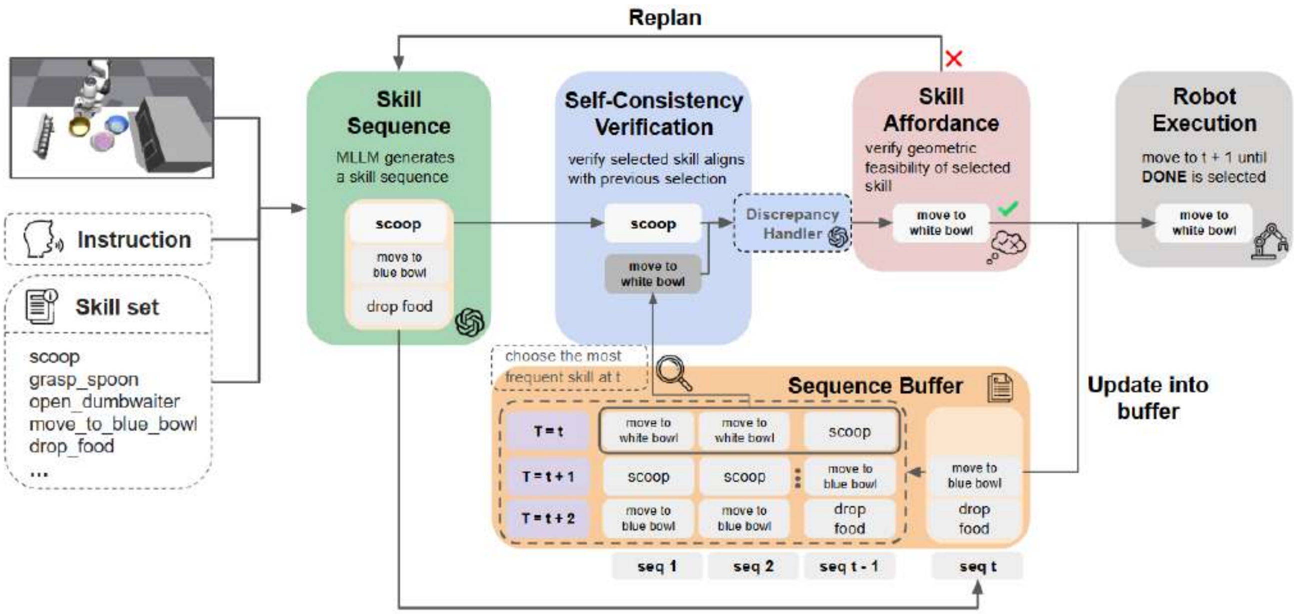

Fig. 2: Overview of our planning pipeline, consisting of the **MLLM Planning Stage** for generating a skill sequence, **Self-Consistency Verification** for stabilizing skill selection, and **Skill Affordance** for verifying geometric feasibility. This process loops until planner select a special termination skill **DONE**.

C. Zero-Shot CoT

In the first stage of our planning pipeline, we provide environmental information and examples to help MLLMs understand the input-output format and objective. To enhance reasoning ability, we integrate a zero-shot CoT [23] approach, prompting the MLLM to first generate an explanation and description of the sub-goal, reason through the current iteration, and then determine the final skill sequence decision. Here is an example response demonstrating how our approach operates:

Description: The blue bowl appears to be fuller, so I will scoop from the blue bowl and drop the contents into the pink bowl.
Iteration 1:
  Output: J. grasp spoon
Iteration 2:
  Output: L. move to blue bowl
Iteration 3:
  Output: A. scoop
Iteration 4:
  Output: M. move to pink bowl
Iteration 5:
  Output: C. drop food
Iteration 6:
  Output: K. put spoon back
Iteration 7:
  Output: I. DONE

D. Self-Consistency Verification

Inspired by previous works [24], [25], we design a mechanism to evaluate the consistency of MLLM across iterations. We instruct MLLM to produce a full sequence of skills instead of a single skill. The skill selected for the current iteration will be considered for execution, while the remaining skills serve as the source of consistency checks and are stored into a sequence buffer when the skill selected for execution is being executed. At each iteration, the selected skill for execution is checked against the majority of previous decisions. If discrepancy exists, MLLM is prompted to reconsider and choose between the conflicting skills.

For example, MLLM generates a sequence s_t with π_t as **scoop** at iteration $T = t$. However, the most frequently chosen skill at $T = t$ from the previous sequence buffer is **move_to_white_bowl**. Due to this discrepancy, the MLLM must resolve the conflict and select a final decision. If the MLLM chooses **move_to_white_bowl** and it is validated by the Skill Affordance module, robot will execute this skill, and the suquence buffer will be updated with s_t.

E. Skill Affordance and Replan

To enable the MLLM to assess the skill affordance, we design predicates [8], which are binary values that answer specific questions about the environment. These predicates are combined into preconditions to evaluate the affordance of each skill. Here is the set of designed predicates along with their corresponding descriptions:

- **spoon_on_hand** True if the robot is currently grasping the spoon.
- **food_on_hand** True if the grasped spoon contains food.
- **dumbwater_opened** True if the dumbwaiter is not fully closed.
- **close_to_target** True if the end effector is sufficiently close to one of the bowls.

- **obstacle_blocked_holder** True if an obstacle is near the holder, hindering the robot from grasping the spoon.
- **obstacle_blocked_dumbwaiter** True if an obstacle is near the dumbwaiter, obstructing the door from opening.
- **reachable** True if the target is is within a feasible range for executing a specific skill.

If an infeasible skill is selected, the replanning process will be triggered, where additional guidance is provided to help the MLLM identify and resolve the issue. The affordance module provides structured feedback, offering critical information about execution failures. For example, the predicate **reachable** provides the MLLM with the guidance "the bowl is too far", alerting it to potential failures. This feedback enhances the model's ability to recognize unsuccessful attempts, refine its selections, and improve its awareness of geometric feasibility within the environment.

When replanning becomes necessary, it indicates that previous selections were heading towards the wrong direction. The prior sequence in the buffer is cleared during replanning to prevent these past mistakes from affecting future decisions.

IV. EXPERIMENT

A. Dataset

We construct a dataset in tabletop food-serving scenario in IsaacGym [9]. The dataset consists of five categories of tasks, each containing 30 unique configurations, resulting in a total of 150 configurations. These configurations vary in instruction, container position and color, as well as the type and amount of food inside the container. In addition to the four categories designed to pose different challenges to MLLM, we create another category to evaluate its reasoning ability solely within the semantic domain. The following are descriptions of the five categories:

- **Semantic Reasoning:** Assess the general semantic planning abilities in our scenario with attributes such as color, food categories, and numbers of scoops. It contains simple food transfer tasks with clear and straightforward instructions and can be solved without image observation.
- **Quantity Estimation:** Assess the ability to recognize extrinsic features such as the quantity of food from image observations and select the correct bowl.
- **Relative Positioning:** Assess the ability to identify specific spatial relationships from image inputs and ground the object when choosing the bowl.
- **Reachability Analysis:** Assess the ability to consider reachability for certain skills. For example, a bowl needs to be repositioned if the bowls are too far to scoop.
- **Collision Avoidance:** Assess the ability to identify and account for potential obstacles. For example, the spoon or the door of dumbwaiter might be blocked by containers and the containers must be repositioned to avoid collisions.

The environment includes a Franka Emika Panda 7-DoF robot arm, a table, a tool holder, and a dumbwaiter. Several bowls are placed on the table, some containing tofu pudding and others containing some toppings. There are seven types of toppings that differ in color.

B. Baselines

To evaluate our approach, we compare against several baselines, as outlined below.

- **Naive LLM:** To evaluate the impact of cross-modal input on reasoning ability, we test the ability of MLLM without image input on our dataset.
- **Naive MLLM:** This baseline use all available information and instructions directly as input and ask MLLM to generate a skill for execution.
- **CoT:** We apply Chain of Thought prompting on the Naive MLLM planner by asking the MLLM to describe what it should achieve in the current iteration before selecting a skill.
- **CoT+SC:** We implement the Self-Consistency mechanism on the CoT baseline.

C. Setup and Implementation

Environmental information. We provide image observation and container list with corresponding toppings to MLLM as input. The image observations are captured by a high angle camera positioned in front of the table with a resolution of 1080×1920. We assume access to an object detector capable of detecting bowls and the corresponding food inside. The initial object list is provided with the format *yellow_bowl (with mung beans)*.
Skill and Affordance. Skills in our predefined skill set are atomic actions such as scooping, opening a dumbwaiter, or grasping a spoon. Focusing on combining these skills to complete the instruction instead of policy training, we simplify these low-level control policies as manually defined trajectories. We leverage information such as the distance between objects and the skill sequence executed to determine the value of predicates. We select relevant predicates to build the precondition for each skill for affordance verification. For the complete skill set, please refer to Appendix A.
Base Model and Prompting. We use OpenAI GPT-4o [26] as the MLLM in our experiments. To guide the model in performing the task, we include descriptions of all skills in the predefined set Π, scenario format, I/O format, and two examples for few-shot in-context learning in the system prompt. For the exact format of the prompt, please refer to Appendix B.
Success Criteria. We define the success criteria as follows: "The robot should complete the task specified in the instruction within three additional steps compared to the reference answer. Bowls should remain intact, and the robot should not transfer any toppings not mentioned in the instruction or open/close the dumbwaiter unless explicitly instructed to do so."

V. RESULT AND DISCUSSION

Table I shows the success rate of each method and dataset categories. Our approach achieve an overall success rate of 76.7%. The result shows that 1) CoT with Self-Consistency

Method	Naive LLM	Naive MLLM	CoT	CoT+SC	CoT+SC+SA
Semantic Reasoning	86.7	26.7	**100.0**	**100.0**	**100.0**
Quantity Estimation	20.0	20.0	46.7	73.3	**76.7**
Relative Positioning	20.0	6.6	36.7	**63.3**	**63.3**
Reachability Analysis	0.0	0.0	0.0	0.0	**76.7**
Collision Avoidance	0.0	0.0	0.0	0.0	**66.7**
Average	25.3	10.7	36.7	47.3	**76.7**

success rate (%) ↑

TABLE I: Experiment results. **SC** stands for Self-Consistency and **SA** stands for Skill Affordance.

enhances MLLM's reasoning ability, resulting in improved performance on tasks in Semantic Reasoning, Quantity Estimation, and Relative Positioning. 2) Skill affordance enable robots to understand geometric feasibility, allowing them to assess action preconditions and successfully complete tasks in Reachability Analysis and Collision Avoidance.

A. Impact of Visual Input on MLLM Reasoning

A decrease in performance is observed in the category "Sematic Reasoning" when comparing Naive LLM and Naive MLLM. Adding images to MLLM could affect its reasoning ability. After incorporating images, most failures occur due to the generation of invalid sequences or misinterpretation of the image. Naive LLM achieves a success rate of 86.7% in semantic reasoning, whereas Naive MLLM struggles with these tasks, achieving only 26.7% of the tasks. Most of the failure cases of Naive MLLM is in producing erroneous actions such as scooping the bowl with tofu pudding or closing the dumbwaiter when it is already closed and not instructed to do so.

B. How CoT Improve Success Rate?

To evaluate the reasoning ability of Naive MLLM, we instruct MLLM to explain its choice of a particular skill for further analysis. Notably, MLLM recognized its mistake and provided the correct answer during its explanation. This observation suggests that MLLM is not yet capable of directly generating accurate skills when influenced by visual distractions. To address this limitation, we employ Chain-of-Thought (CoT) prompting to enhance the robot's ability to deal with such distractions.

C. How Self-Correction Improve Success Rate?

When using CoT, most failures occur when the robot does not recognize when to stop. We prompt MLLM to reason on what it needs to achieve at the current iteration, but MLLM sometimes fails to express that the goal has already been achieved, even with additional guidance. With self-correction, MLLM demonstrated improved reasoning about stopping conditions as a long-term objective, enabling the verification process to proceed. In CoT+SC, There are a total of 18 inconsistent skill selections, and MLLM correctly handled only 50% of such cases.

VI. CONCLUSIONS

In this work, we explore the application of Multimodal Large Language Models (MLLMs) for task planning in food preparation, identifying key challenges associated with cross-modal distraction and geometric feasibility. Our findings highlight that while MLLMs demonstrate strong reasoning abilities, their performance can degrade when unnecessary visual input introduces distractions. To mitigate this, we applied Chain of Thought reasoning with Self-Consistency, improving the model's robustness against cross-modal interference. Additionally, we integrated skill affordance as a physics engine, enabling the model to assess geometric feasibility when planning robotic actions.

Through the construction of a dedicated dataset and evaluation in a closed-loop task planning setting, we demonstrated the effectiveness of our approach in handling cross-modal distraction and recognizing geometric feasibility in food preparation tasks. Our contributions include dataset development, the adaptation of reasoning techniques for multimodal task planning, and the incorporation of action preconditions to enhance feasibility assessments.

VII. LIMITATIONS AND FUTURE WORK

In this work, we propose Self-Consistency verification and affordance guidance to address challenges in MLLMs with in-context learning for food preparation task planning. However, our approach has several limitations.

Simplified Control Policies & Skill Affordance: We rely on simplified policies, limiting task diversity. For example, our scooping policy does not allow specifying quantities, preventing tasks like "scoop half a spoon of mung beans." Future work will explore integrating food manipulation policies [1], [2].

Dependence on Object Detection. We assume access to a reliable object detector for identifying bowls and their contents, providing a pre-processed container list to the

MLLM. However, we do not account for detector failures, which may introduce mismatches between semantic input and visual observations.

MLLM Limitations in Quantity & Spatial Understanding. While MLLMs exhibit general image recognition abilities, they sometimes misinterpret quantities and spatial relationships. Instead of improving their reasoning directly, we enforce consistency verification between historical and current selections.

As a future direction, we consider leveraging multiple specialized MLLMs—one for reasoning and another for visual question answering (VQA). This approach would allow a reasoning-focused model to handle planning while a vision-focused model resolves inconsistencies in quantity and spatial understanding.

APPENDIX

A. Complete Skill Set and Description

- **grasp_spoon** Grasp the spoon from the tool holder.
- **put_spoon_back** Put the spoon back to the tool holder.
- **scoop** Scoop food
- **drop_food** Drop the food from the spoon into the container.
- **stir** Stir the food.
- **pull_bowl_closer** Pull the nearest bowl toward the center of the table.
- **open_dumbwaiter** Open the dumbwaiter door.
- **close_dumbwaiter** Close the dumbwaiter door.
- **put_bowl_into_dumbwaiter** Place the nearest bowl into the dumbwaiter.
- **start_dumbwaiter** Start the dumbwaiter.
- **move_to_{color}_bowl** Move to a container for actions like pulling or scooping.
- **DONE** Special termination skill.

B. Prompt Example

System prompt: A fixed and structured system prompt provided to the MLLM, organized into six sections and including two examples for in-context learning.

 # Scenario
You are a robotic arm specialized in food manipulation tasks. Your mission is to complete the assigned task step-by-step by selecting the most appropriate actions from the provided list. Your decisions should balance precision, safety, efficiency, and task progression. Take the previous actions into consideration and choose the best actions for the remaining sequence from the skill set. **You should describe the reasoning behind your decision and consider the high-level goal of the task before making a choice.**
// zero-shot Chain of Thought

 # Additional Knowledges
1. Scooping guidelines
A single scoop should be done by selecting [move_to_container(with food), scoop, move_to_container(destination), drop_food] when the spoon is on the gripper.
2. Collision Avoidance
If an action risks a collision or task failure, pull the bowl to a safer location before proceeding.
3. Scooping limitations
Avoid scooping from bowls with insufficient food (e.g., only a few beans).
If a bowl is too far, pull it closer before attempting to scoop.

 # Action Description
1. **grasp_spoon**: Grasp the spoon from the tool holder. The robot arm must have no tools in the gripper when choosing this action.
2. **put_spoon_back**: Put the spoon back to the tool holder.
3. **move_to_container**: Move to a container for actions like pulling or scooping.
4. **scoop**: Scoop food, with the speed adapted to the food's state. 5. **stir**: Stir the food.
6. **drop_food**: When the robot arm is positioned above a container, drop the food from the spoon into the container.
7. **pull_bowl_closer**: When the gripper is empty, pull the nearest bowl toward the center of the table.
8. **open_dumbwaiter**: Open the dumbwaiter door.
9. **close_dumbwaiter**: Close the dumbwaiter door.
10. **put_bowl_into_dumbwaiter**: Place the nearest bowl into the dumbwaiter.
11. **start_dumbwaiter**: Start the dumbwaiter.
12. **DONE**: Indicate that the task is complete.

 # Scenario Format
You will be presented with a single scenario containing the following details:
Skill set: A list of all actions that the robot can perform, formatted as character. action.
Initial Object List: A detailed inventory of objects present in the environment, formatted as container_name (food inside).
Instruction: The high-level task or goal that the robot must accomplish.
Iterative Previous Actions: A chronological record of the actions the robot has executed in prior iterations.
// only in the pipeline with Skill Affordance module
Previous Affordance Feedback: A record of action names, their failure reasons, and some suggestion from previous iterations. Please consider this information when making your decision.
// only in the pipeline with visual input
Current Observation: An image of the robot's current environment.

 # Input Format
You will be provided with several examples, each

illustrating a unique scenario in the format described above.
Following these, another scenario will be presented, requiring you to deduce and choose the next optimal action.

Output Requirements
Select and output some actions from the provided Skill set in your task as the actions to execute in order.
The response should exclude all formatting characters such as backticks, quotes, or additional symbols.
You should provide a sequence of action as answer, starting from current iteration until selecting DONE.
Format the first line of your response strictly as: Description: [your description].
Format the rest of the line of your response strictly as: ”
Iteration [number]:
Output: [character]. [action]”. Please use the format in the examples as a reference.

Examples
Example 1:
Skill set: [’scoop’, ’stir’, ’drop food’, ’pull bowl closer’, ’open dumbwaiter’, ’close dumbwaiter’, ’start dumbwaiter’, ’put bowl into dumbwaiter’, ’DONE’, ’grasp spoon’, ’put spoon back’, ’move to white bowl’, ’move to green bowl’, ’move to blue bowl’] Initial object list: [’white bowl (with mung beans)’, ’green bowl (with mung beans)’, ’blue bowl (with tofu pudding)’]
Instruction: Try your best to get more beans into the bowl with tofu pudding in one scoop. Put it into the dumbwaiter after you finish scooping.

Iteration 1:
Output: grasp spoon
Iteration 2:
Output: move to white bowl
Iteration 3:
Output: scoop
Iteration 4:
Output: move to blue bowl
Iteration 5:
Output: drop food
Iteration 6:
Output: put spoon back
Iteration 7:
Output: move to green bowl
Iteration 8:
Output: pull bowl closer
Iteration 9:
Output: open dumbwaiter
Iteration 10:
Output: move to blue bowl
Iteration 11:
Output: put bowl into dumbwaiter
Iteration 12:
Output: close dumbwaiter
Iteration 13:
Output: start dumbwaiter
Iteration 14:
Output: DONE

Explanation:
- White bowl has more beans to scoop, so you should scoop from white bowl to green bowl.
- Green bowl is too close to the dumbwaiter door, potentially obstructing the door from opening during execution.

Example 2:
Skill set: [’scoop’, ’stir’, ’drop food’, ’pull bowl closer’, ’open dumbwaiter’, ’close dumbwaiter’, ’start dumbwaiter’, ’put bowl into dumbwaiter’, ’DONE’, ’grasp spoon’, ’put spoon back’, ’move to purple bowl’, ’move to red bowl’]
Initial object list: [’red bowl (with mung beans)’, ’purple bowl (with tofu pudding)’]
Instruction: Place two scoop of beans into purple bowl.

Iteration 1:
Output: grasp spoon
Iteration 2:
Output: move to red bowl
Iteration 3:
Output: scoop
Iteration 4:
Output: move to purple bowl
Iteration 5:
Output: drop food
Iteration 6:
Output: move to red bowl
Iteration 7:
Output: scoop
Iteration 8:
Output: move to purple bowl
Iteration 9:
Output: drop food
Iteration 10:
Output: DONE

Explanation:
- This example demonstrate how scoops should be done.
- Iteration 2 to 5 demonstrates how to scoop from red bowl to purlple bowl once, and iteration 6 to 9 repeats it.

Example user prompt: Includes environmental information, task instructions, and feedback messages from the Skill

Affordance module, along with the current observation.

Your task:

Skill set: ['A. scoop', 'B. stir', 'C. drop food', 'D. pull bowl closer', 'E. open dumbwaiter', 'F. close dumbwaiter', 'G. start dumbwaiter', 'H. put bowl into dumbwaiter', 'I. DONE', 'J. grasp spoon', 'K. put spoon back', 'L. move to white bowl', 'M. move to blue bowl']

Initial object list: ['white bowl (with kidney beans)', 'blue bowl (with tofu pudding)']

Instruction: Place some kidney beans into the blue bowl.

Previous Affordance Feedback:

In iteration 3, Cannot do scoop because the target white bowl is too far, please pull it closer. Cannot do pull_bowl_closer because spoon is on hand, please put it back first.

Iteration 1:

Output: J. grasp spoon

Iteration 2:

Output: L. move to white bowl

Iteration 3:

Output:

REFERENCES

[1] Yen-Ling Tai, Yu Chien Chiu, Yu-Wei Chao, Yi-Ting Chen, SCONE: A Food Scooping Robot Learning Framework with Active Perception, Conference on Robot Learning (CoRL), 2023.

[2] N. Ha, R. Ye, Z. Liu, S. Sinha and T. Bhattacharjee, "REPeat: A Real2Sim2Real Approach for Pre-acquisition of Soft Food Items in Robot-assisted Feeding," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024.

[3] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, Tomás Lozano-Pérez, Integrated Task and Motion Planning, Annual Review of Control, Robotics, and Autonomous Systems, 2021.

[4] Ishika Singh, Valts Blukis, Ar salan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, Animesh Garg, ProgPrompt: Generating Situated Robot Task Plans using Large Language Models, IEEE International Conference on Robotics and Automation (ICRA), 2023.

[5] Michael Ahn, et al., Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, Conference on Robot Learning (CoRL), 2022.

[6] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents, NeurIPS, 2023.

[7] Lin, K., Agia, C., Migimatsu, T., Pavone, M., Bohg, J., Text2Motion: from natural language instructions to feasible plans, Autonomous Robots, 2023.

[8] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, Leslie Pack Kaelbling, Guiding Long-Horizon Task and Motion Planning with Vision Language Models, Language and Robot Learning Language as an Interface (CoRL-LangRob), 2024.

[9] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, Gavriel State, Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning, Neural Information Processing Systems Datasets and Benchmarks Track, 2021.

[10] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh, VQA: Visual Question Answering, International Conference on Computer Vision (ICCV), 2015.

[11] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, Douwe Kiela, FLAVA: A Foundational Language And Vision Alignment Model, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

[12] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, Tianyi Zhou, HallusionBench: An Advanced Diagnostic Suite for Entangled Language Hallucination and Visual Illusion in Large Vision-Language Models, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2024.

[13] Fuxiao Liu and Kevin Lin and Linjie Li and Jianfeng Wang and Yaser Yacoob and Lijuan Wang, Mitigating Hallucination in Large Multi-Modal Models via Robust Instruction Tuning, The International Conference on Learning Representations (ICLR), 2024.

[14] Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Ray Mooney, Stefanie Tellex, David Paulius, CAPE: Corrective Actions from Precondition Errors using Large Language Models, IEEE International Conference on Robotics and Automation (ICRA), 2024.

[15] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, Andy Zeng, Code as Policies: Language Model Programs for Embodied Control, IEEE International Conference on Robotics and Automation (ICRA), 2023.

[16] Wenlong Huang, et al., Inner Monologue: Embodied Reasoning through Planning with Language Models, Conference on Robot Learning (CoRL), 2022.

[17] Yanjiang Guo, Yen-Jen Wang, Lihan Zha, Jianyu Chen, DoReMi: Grounding Language Model by Detecting and Recovering from Plan-Execution Misalignment, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024.

[18] Xiaohan Zhang, Zainab Altaweel, Yohei Hayamizu, Yan Ding, Saeid Amiri, Hao Yang, Andy Kaminski, Chad Esselink, Shiqi Zhang, DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning, Planning in the Era of LLMs (AAAI-LM4Plan), 2024.

[19] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, Serkan Cabi, Vision-Language Models as Success Detectors, Conference on Lifelong Learning Agents (CoLLAs), 2023.

[20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, Denny Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, NeurIPS, 2022.

[21] Zhuosheng Zhang, Aston Zhang, Mu Li, Alex Smola, Automatic Chain of Thought Prompting in Large Language Models, International Conference on Learning Representations (ICLR), 2022.

[22] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, Karthik Narasimhan, Tree of Thoughts: Deliberate Problem Solving with Large Language Models, NeurIPS, 2023.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa, NeurIPS, 2022.

[24] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, Denny Zhou, Self-Consistency Improves Chain of Thought Reasoning in Language Models, International Conference on Learning Representations (ICLR), 2023.

[25] Han Wang, Archiki Prasad, Elias Stengel-Eskin, Mohit Bansal, Soft Self-Consistency Improves Language Model Agents, Annual Meeting of the Association for Computational Linguistics (ACL), 2024.

[26] OpenAI, GPT-4o System Card, arXiv:2010.01083, 2024.