

## A Inner Monologue Implementation Details

We study three different implementations of Inner Monologue for each of the experimental settings. While each version incorporates textual environment feedback into planning, there are differences in the internal components of each system, as seen in Table 1.

	Tabletop Rearrangement (Sim)	Tabletop Rearrangement (Real)	Kitchen Mobile Manipulation (Real)
LLM	InstructGPT	InstructGPT	PALM
Policies	CLIPort + TransporterNets	MDETR + heuristics	Learned policies
Success Feedback	Ground truth + heuristics	MDETR + heuristics	Finetuned CLIP
Object Feedback	Ground truth + heuristics	MDETR	Human-provided (structured)
Scene Feedback	Ground truth + heuristics	N/A	N/A
Human Feedback	N/A	N/A	Human-provided (unstructured)

Table 1: Comparison between different versions of Inner Monologue implemented in three different environments.

### A.1 Inner Monologue for Simulated Tabletop Rearrangement

**Large Language Model** We use InstructGPT [?] (text-davinci-002), accessed through OpenAI API.

**Low-level Policies** We use a CLIP-based [?] Transporter Net [?] (inspired by CLIPort [? ]) as the pick-and-place primitive that can be invoked by the LLM planner. The policy is trained on 20000 pre-collected demonstrations, where each demonstration contains 1) language instruction of the format “pick up [x] and place it on [y]”, 2) top-down view of RGB-D observation of the current environment, 3) expert pick coordinates, and 4) expert place coordinates. The expert pick and place coordinates are obtained by accessing ground-truth object pose in the simulator. Unlike the evaluated settings in [? ], the demonstrations cover all objects because the focus of this work is not on generalization across object instances but on novel long-horizon behaviors. The policy outputs a 3D pick location and a 3D place location, and a scripted pick-and-place motion is executed following the predicted locations.

**Environment Feedback: Object Recognition** We provide the list of objects present in the scene at the start of each episode for the language model (without bounding boxes or coordinates because spatial manipulation is handled by low-level policies). Although it has been shown in [?] that this can be obtained by off-the-shelf pre-trained open-vocabulary object detection models such as ViLD [?] and MDETR [? ], we obtain the list of objects by accessing simulator state for simplicity.

**Environment Feedback: Success Detection** For *Object + Success* method, we provide textual feedback of low-level policy success detection results after each policy execution. The success detector is built using heuristics based on ground-truth object poses: a pick-and-place execution is successful if picked object has a 2D Euclidean distance  $< 4$  cm from the place object and its height is greater than the place object. If the place object is a location (e.g., top side of the table) rather than a block or a bowl, the height requirement is not enforced.

**Environment Feedback: Passive Scene Description** For *Object + Scene* method, we provide task-progress scene description as a list of achieved sub-goals after each pick-and-place execution. At the start of the action plan, the language model first generates a list of desired sub-goals given the high-level instruction. The sub-goals are of the format “[x] is on top of [y]”. After each pick-and-place execution, we iterate through all desired sub-goals inferred by the language model, and check which are satisfied using the success detector described above. The full list of currently satisfied sub-goals are appended to the language model prompt. Additionally, we allow the language model to generate chain-of-thought summarization following the achieved sub-goals (i.e., “Robot thought: ...”), which we find to be useful empirically.

### A.2 Inner Monologue for Real-World Tabletop Rearrangement

**Large Language Model** We use InstructGPT [?] (text-davinci-002), accessed through OpenAI API.

**Low-level Policies** We use a single low-level policy for the real tabletop rearrangement environment that is responsible for performing object-centric pick and place actions as instructed by the language model.

The policy takes as input 1) the bounding boxes of all the objects in the scene, 2) the names of the object to pick and the target object for place, 3) the depth image corresponding to the RGB image from which object detection was done, 4) camera intrinsics, and 5) camera pose. It then outputs a 3D pick position and a 3D place location for the robot to follow.

The pick position is the 3D coordinate of the center of the object bounding box in the robot base frame. This is obtained by first deprojecting the depth image pixel corresponding to the center of the pick object bounding box with the given camera intrinsics, then transforming this point from the camera frame to the robot frame with the given camera pose. Camera pose is known because it is a wrist camera, and we register the delta pose from the mounted camera to the robot end-effector.

For the block stacking task, the place location is obtained the same way as the pick position. For the object sorting task, the place location is chosen to be a point in the target plate's bounding box that is farthest way from the bounding boxes of other objects. This is done to prevent undesired stacking behavior when placing objects into the plate, which may cause the object to roll off and fall off the table.

To perform the pick and place motions, the robot moves to a position 15 cm above the intended pick or place position, and then it slowly lowers the end-effector until a 5 N contact force is detected. This is more robust than moving directly to the height obtained from the depth image due to depth sensor noises. We set a threshold for how low the end-effector can go for the object sorting task so picking inaccuracies do not result in accidentally picking up the plate.

We add planar translation noise to the pick position by sampling the magnitude from a zero-mean Normal distribution and then sampling an angle from a uniform distribution on the unit circle. The standard deviation for the Normal distribution is  $\sigma = 1.5$  cm for the block stacking task,  $\sigma = 0.7$  cm for the object sorting task, and the samples are capped at  $1.5\sigma$ .

**Object Recognition Implementation** Object detection is done by MDETR [? ], an open-vocabulary object detection model. It takes as input an RGB image with an image caption, and the model tries to identify the objects described in the caption. To form the caption for our tasks, we join the list of all possible object names together, separated by periods. Note that not all provided objects will be detected due to occlusions. MDETR detections are not always robust or consistent. For example, the detected object bounding box may some times be bigger or smaller than the object, or its center may be far away from the object center. Its performance also drops when objects are in close clutter. These issues with perception further motivate the need for success detection and multi-step environment feedback, so the high-level language model planner can be robust to these inaccuracies.

**Environment Feedback: Success Detection** In the real tabletop rearrangement environment, success detection is done by comparing the 2D position of the center of the detected object bounding box after pick and place action with the intended 2D place position. By 2D we mean the x-y spatial coordinates in the base frame of the robot, not the image coordinates. These are obtained by deprojecting the detected bounding box centers from the depth image and converting the resulting 3D point from camera frame to base frame. The detector returns success if the difference between the two positions is less than a threshold. For the block stacking task, the threshold is 3 cm, and for the object sorting task it is 10 cm. While 3 cm is larger than half of the dimension of the blocks we use (4 cm), it is more robust for our case because the center of the detected object bounding box is often not the center of the block. The threshold for object sorting is much higher, because for some objects, once they are placed in the plate, they may roll around until contact with the plate boundary.

**Environment Feedback: Object Recognition** For the block stacking task, the scene description contains a list of currently visible objects and a list of previously visible objects that are no longer visible. We may add new objects to the currently visible objects list if, after a robot action, MDETR detects an object that we have not seen before. This happens for the block stacking task as there is an initial partially constructed tower of two blocks, the bottom of which is initially occluded.

For the object sorting task, the scene description contains a list of currently visible objects and a list of objects that the robot has successfully moved into a plate. Like the block stacking task, some objects are initially occluded, so later actions may reveal them and they will be added to the visible objects list. The successfully moved object list is needed so that the planner does not stop before task completion and that it stops on task completion.

### A.3 Inner Monologue for Real-World Mobile Manipulation in a Kitchen Setting

**Large Language Model** We use PALM [? ], a 540B parameter language model trained on a large datasets that include high-quality web documents, books, Wikipedia, conversations, and GitHub code.

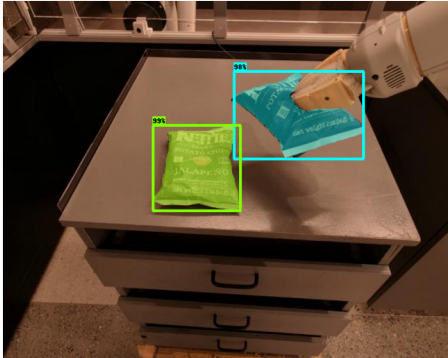
**Low-level Policies** Following the implementation described in SayCan [? ], we use a combination of learned and scripted control policies for navigation and manipulation. The learned manipulation policies responsible for counter picking, drawer opening and closing, drawer picking, and countertop object manipulation are Behavior Cloning (BC) policies trained on 68000 teleoperated demonstrations and 12000 autonomous successes that were collected over the course of 11 months using a fleet of 10 robots. The teleoperators use VR headset controllers to track the motion of their hand, which is then mapped onto the robot’s end-effector pose. The scripted navigation policies utilize a ground-truth map along with a learned perception module to navigate between different points in the environment. The scripted manipulation policy is solely responsible for countertop placing when preceded by a navigation policy, and follows pre-computed motions. The Value Functions used by SayCan for affordance grounding are provided by the  $Q$ -networks of trained RL agents; we follow the RL training setup described in [? ].

**Environment Feedback: Object Recognition** We use human-provided object recognition to provide feedback about the presence of objects visible to the robot camera. For example, if there were only a coke can and an apple on top of the kitchen counter, then the human-provided object recognition feedback would appear as “[scene: coke can, apple]”.

While we utilize human-provided object recognition to gauge the role of maximally accurate *Object* feedback, we also study the feasibility of using learned object recognition models to provide fully automated *Object* feedback. We find that two such models, ViLD [? ] and MDETR [? ], perform worse than humans but still quite resonably at providing *Object* feedback, even without domain-specific data for fine-tuning. As advances in computer vision improve object detection models that can transfer zero-shot or few-shot to novel environments like our kitchen environment, we expect that fully automated *Object* feedback will increasingly approach human-provided accuracy.

	ViLD	MDETR
<b>Precision</b>	85.7%	39.6%
<b>Recall</b>	72.0%	87.5%
<b>Accuracy</b>	88.9%	68.2%

**Table 2:** Comparison of ViLD [? ] and MDETR [? ], two open-vocabulary object detection models. We evaluate these two pretrained models on 10 representative kitchen mobile manipulation episodes. ViLD has strong overall accuracy, but still fails to detect objects 28.0% of the time.



**Figure 1:** An example of ViLD object detection segmentation mask and bounding box predictions.

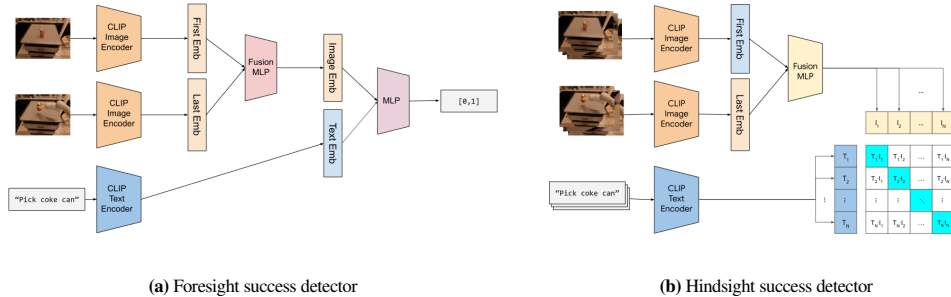
#### Environment Feedback: Success Detection

We trained a “foresight” success detector on offline data collected from a combination of teleoperated demonstrations and autonomous roll-outs performing the low-level tasks. The input to the model consists of: (1)  $o_0$ , the initial image observation, (2)  $o_f$ , the final image observation after the policy chose to terminate its chosen skill execution, and (3)  $l_k$ , the low-level skill which the agent was trying to execute (e.g., “Pick coke can”). The model uses image encoders from CLIP [? ] to embed  $o_0$  and  $o_f$ , concatenate them, and fuse these representations with a fusion MLP. This image embedding is concatenated with the text embedding obtained by the CLIP text encoder, then passed through another MLP. The output of the model is a scalar denoting the probability of the agent succeeding at the specified task (Fig 2a). The model is trained with the binary cross entropy loss with respect to the ground truth binary label. We fine tune the CLIP model as well during the training. At inference time within Inner Monologue, we output the text “[success: no]” when the probability is below a certain threshold.

To reduce false positive predictions, we trained a separate “hindsight” success predictor model. Given the first and last observation, the model outputs a probability distribution over all the possible skills. We used this model to disambiguate between possible tasks that the agent may have achieved. To train this

model, we use the symmetric contrastive loss as used in CLIP (Fig 2b). At inference time, similar to the CLIP model, we take a dot product between the image-fusion embedding and the text embeddings and apply softmax with a learned temperature.

We combine the foresight and hindsight model by first thresholding the probability from the foresight model by some value  $\tau$ . If the probability is above, then we run the hindsight model and only predict success if the argmax across all skills is indeed the foresight task.



**Figure 2:** Success Detection architecture used for the Kitchen Environment. (Left) The foresight model predicts whether a given instruction was successfully achieved between the first and last image. (Right) The hindsight model fine-tuned via contrastive loss as used in CLIP [? ]. At inference time, the model is used infer among the possible instructions which one was achieved.

**Environment Feedback: Active Scene Description** We perform a case study where we allow the LLM agent to ask questions and source *Human* feedback directly. After each LLM planning step, the LLM chooses between and continue or and ask: for the and continue option the LLM proceeds to the planning the next action to execute, and for the and ask option the LLM proceeds to propose a question to ask the human. The questions proposed by the LLM do not follow any pre-determined format or scope, so human-provided answers are similarly open-ended. These answers can range from confirming factual observations about the scene to providing refinements or changes to the high-level plan. Some examples can be found in Sec D.

## B Experiment Details

### B.1 Simulated Tabletop Rearrangement Environment

There are a total of 8 tasks, listed in Table ?? . For all tasks, up to 4 blocks and up to 3 bowls are randomly initialized with different colors (chosen from 10 different colors) and initial locations. A 15 cm distance is enforced between any two objects to avoid collision and penetration at initialization. For tasks that involve a specific object instance or location in the task instruction (e.g., “Put all the blocks in the [x] bowl”), a random task instruction is sampled using the available objects in the scene or all available locations. There are a total of a total of 9 locations (e.g., “top side”, “bottom right corner”, etc). To allow for automatic evaluations, a binary reward function is defined for each task using ground-truth state of the objects.

**CLIPort Baseline** We additionally train a multi-task CLIPort [? ] policy on 4 of the total 8 tasks as an additional baseline (see Table ?? for train/test split). The goal is to evaluate whether solitary language-conditioned policies can perform well on long-horizon tasks and generalize to new task instructions. Note that this is a different policy as the low-level primitive used in Inner Monologue, though sharing the same architecture. The policy also shares the same training details as the low-level policy used in Inner Monologue but is trained on 20000 demonstrations across 4 training tasks.

### B.2 Real Tabletop Rearrangement

There are two tasks in the real tabletop rearrangement domain – a block stacking task and an object sorting task.

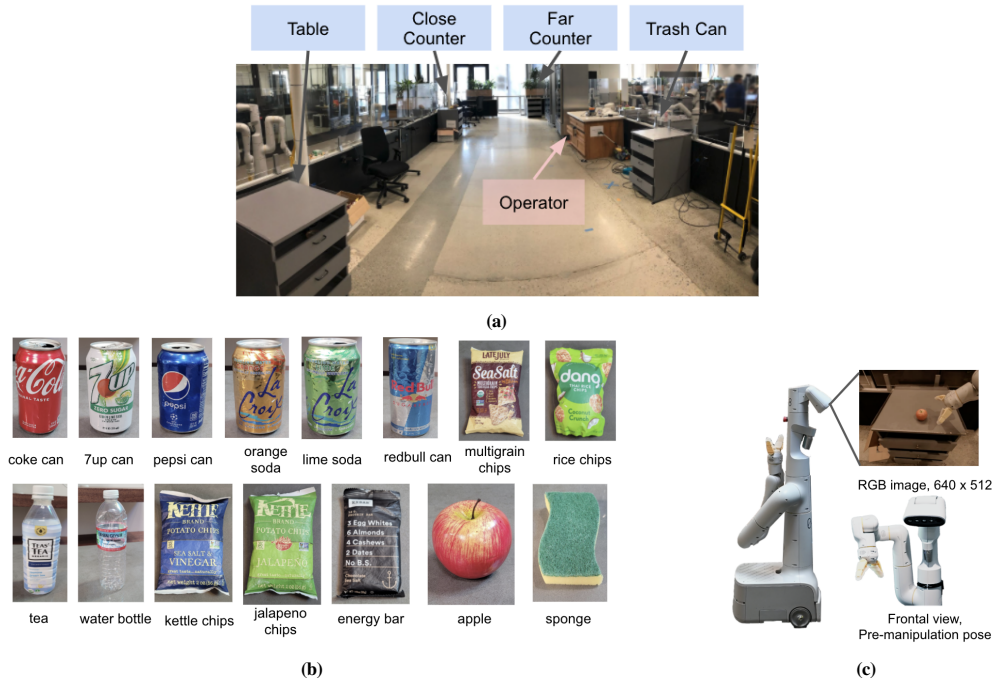
In the block stacking task, the initial state consists of three 4 cm cubes, two of which have been already stacked. The robot needs to complete stacking the full tower either by putting the third block onto the partially stacked two, or by putting the two partially stacked blocks onto the third block. Due to occlusions,

the robot only perceives two blocks in the beginning, so without scene description feedback, the planner has a 50% chance of completing the task.

For the object sorting task, the initial state consists of three fruits, three bottles, and three plates. The fruits and bottles are placed in a clutter in one of the three plates, and the robot is asked to sort them by putting fruits on one of the remaining plates, and bottles on the other. Due to occlusions and reduced object detection performance in clutter, some objects may not be detected in the initial clutter, so scene description feedback is still needed. The three bottles used are a ketchup bottle, a mustard bottle, and a soft drink bottle. The three fruits used are an apple, a strawberry, and a plum. Note these are toy objects made out of plastic, so they are smaller and lighter than their real-world counterparts, and they are also rigid, allowing for robust suction grasps. They have realistic visual appearances, which allow MDETR detections.

### B.3 Real Kitchen Mobile Manipulation

We follow the environment and open-ended task setup described in SayCan [? ]. An *Everyday Robots* robot, a mobile manipulator robot with RGB observations, is placed in a kitchen with multiple named locations that contain a set of 15 household items. Given an unstructured human instruction, the robot must complete the instruction successfully by executing a set of navigation and manipulation skills, which are vision-based concurrent [? ] continuous closed-loop policies. Fig 3 presents a high-level overview of the environment. The full list of tasks are shown in Table 3.



**Figure 3:** The experiments were performed in a mock office kitchen mirroring this setup (a), with 5 predicted locations and 15 objects (b). The robot is a mobile manipulator with a 7 degree-of-freedom arm and a gripper with policies trained from an RGB observation (c).

	Task Instructions
<b>Manipulation</b>	<p>“pick up a soda”</p> <p>“pick up the jalapeno or kettle chips”</p> <p>“pick up a snack”</p> <p>“pick up a caffeinated drink”</p>
<b>Mobile Manipulation</b>	<p>“can you throw away the soda on the table”</p> <p>“I spilled my coke, can you bring me something to clean it up”</p>
<b>Drawer Manipulation</b>	<p>“leave the top drawer open”</p> <p>“put a coke in the top drawer”</p>

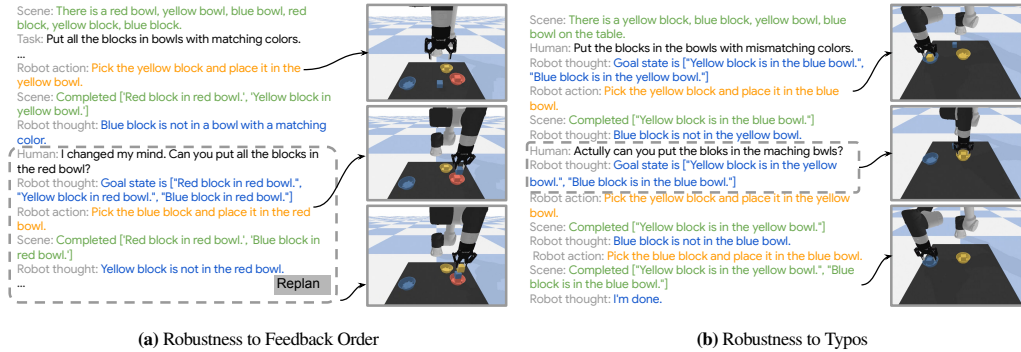
**Table 3: List of all instructions in Kitchen Mobile Manipulation tasks.** We evaluate the algorithm on 3 families of instructions on a mock kitchen scene.

## C Additional Results

We show additional results of emergent capability in Fig 4.

**Robustness to Feedback Order.** In the main experiments of the paper, LLM prompts follow certain conventions (e.g., [Robot action, Scene, Robot thought] in sim tabletop). In practice, we find that the LLM planner is robust to occasionally swapping the order of feedback. Appendix Fig 4a shows when a new human instruction is injected mid-task, the planner can recognize the change and generates a new “Robot thought:...” statement allowing it to solve the new task.

**Robustness to Typos.** Inherited from the LLM backbone, our approach is robust to typos in human instruction, as seen in Appendix Fig 4b.



**Figure 4: Additional emergent capabilities in Inner Monologue.** While we mostly use a fixed feedback order for each domain, we find that Inner Monologue can seamlessly handle different feedback injected into the prompt, as opposed to the normal case of feedback being provided after the prompt and after planning steps (left). Additionally, it is also robust to typos in the instruction, an ability likely inherited from the backbone LLM (right).

## D Prompts

**Listing 1:** Prompt in Inner Monologue on Simulated Tabletop Rearrangement

```
=====
Scene: There is a cyan, yellow, brown block.
Human: Move all the blocks to the top left corner.
Robot thought: Goal state is ["Cyan block is
    on the top left corner.", "Yellow block is on the top left corner.", "Brown block is on the top left corner."]

Robot action: Pick the cyan block and place it on the top left corner.
Scene: You have completed ["Cyan block is on the top left corner."]
Robot thought: Yellow and brown blocks are not on the top left corner.

Robot action: Pick the yellow block and place it on the top left corner.
Scene: You have completed ["Cyan block is on the top left corner."]
Robot thought: Yellow and brown blocks are not on the top left corner.

Robot action: Pick the yellow block and place it on the top left corner.
Scene: You have completed ["Cyan block is on the top left corner.", "Yellow block is on the top left corner."]
Robot thought: Brown block is not on the top left corner."

Robot action: Pick the brown block and place it on the top left corner.
Scene: You have completed ["Yellow block is on the top left corner.", "Brown block is on the top left corner."]
Robot thought: Cyan block is not on the top left corner

Robot action: Pick the cyan block and place it on the top left corner.
Scene: You have completed ["Yellow block
    is on the top left corner.", "Brown block is on the top left corner.", "Cyan block is on the top left corner."]
Robot thought: None.

=====
Scene: There is a pink, gray, orange block.
Human: Move all the blocks to the bottom side.
Robot thought: Goal state is ["
    Pink block is on the bottom side.", "Gray block is on the bottom side.", "Orange block is on the bottom side."].

Robot action: Pick the pink block and place it on the bottom side.
Scene: You have completed ["Pink block is on the bottom side."]
Robot thought: Gray and orange blocks are not on the bottom side.

Robot action: Pick the gray block and place it on the bottom side.
Scene: You have completed ["Pink block is on the bottom side.", "Gray block is on the bottom side."]
Robot thought: Orange block is not on the bottom side.

Robot action: Pick the orange block and place it on the bottom side.
Scene: You have completed ["Orange block is on the bottom side."]
Robot thought: Pink and gray blocks are not on the bottom side.

Robot action: Pick the gray block and place it on the bottom side.
Scene: You have completed ["Pink block is on the bottom side.", "Gray block is on the bottom side."]
Robot thought: Orange block is not on the bottom side.

Robot action: Pick the orange block and place it on the bottom side.
Scene: You have completed ["
    Pink block is on the bottom side.", "Gray block is on the bottom side.", "Orange block is on the bottom side."]
Robot thought: None.
```

**Listing 2:** Prompt in Inner Monologue on Real Tabletop Rearrangement

```
=====
Task: Move all blocks to the top left corner

Scene: Visible objects are ["cyan block", "yellow block", "brown block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("cyan block", "top left corner")
Successful action: True

Scene: Visible objects are ["cyan block", "yellow block", "brown block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("yellow block", "top left corner")
Successful action: False

Scene: Visible objects are ["cyan block", "yellow block", "brown block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("yellow block", "top left corner")
Successful action: True

Scene: Visible objects are ["yellow block", "brown block"]
```



```

Scene: Occluded objects are ["cyan block"]
Robot action: robot.pick_place("brown block", "top left corner")
Successful action: True

Scene: Visible objects are ["brown block"]
Scene: Occluded objects are ["cyan block", "yellow block"]
Robot action: robot.stop()
STOP

=====
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: False

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block", "orange block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.pick_place("orange block", "brown block")
Successful action: True

Scene: Visible objects are ["orange block"]
Scene: Occluded objects are ["purple block", "brown block"]
Robot action: robot.stop()
STOP

=====
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.stop()
STOP

=====
Task: Put objects in their corresponding bowls

Scene: Visible objects are ["red block", "green block", "red bowl", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red block", "red bowl")
Successful action: False

Scene: Visible objects are ["red block", "green block", "red bowl", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red block", "red bowl")
Successful action: True

Scene: Visible objects are ["red block", "green block", "red bowl", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("green block", "green bowl")
Successful action: True

Scene: Visible objects are ["red block", "green block", "red bowl", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.stop()
STOP

=====
Task: Sort pencils and markers and in different containers

Scene: Visible objects are ["red marker", "red pen", "blue box", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red marker", "green bowl")
Successful action: False

Scene: Visible objects are ["red marker", "red pen", "blue box", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red marker", "green bowl")
Successful action: False

```



```

Scene: Visible objects are ["red marker", "red pen", "blue box", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red marker", "green bowl")
Successful action: True

Scene: Visible objects are ["red marker", "red pen", "blue box", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.pick_place("red pen", "blue box")
Successful action: True

Scene: Visible objects are ["red marker", "red pen", "blue box", "green bowl"]
Scene: Occluded objects are []
Robot action: robot.stop()
STOP
=====

```

**Listing 3:** Prompt in Inner Monologue in Kitchen mobile manipulation environment.

```

Robot: Hi there, I'm a robot operating in an office kitchen.
You can ask me to do various tasks and I'll tell you the sequence of actions I would do to accomplish your task.
Human: hold the snickers
Robot: 1. pick up the snickers[success: no][scene: snickers]
2. pick up the snickers
2. done.
Human: put the trailmix here
Robot: 1. put down the trailmix
2. done.
Human: put a water bottle and an oatmeal next to the microwave
Robot: 1. find a water
2. pick up the water[success: no]
3. pick up the water[success: no]
4. pick up the water
5. go to microwave
6. put down the water[scene: water, microwave]
7. find an oatmeal
8. pick up the oatmeal[scene: oatmeal]
9. go to the microwave
10. put down the oatmeal[scene: microwave, oatmeal, water]
11. done.
Human: put a grapefruit from the table into the bowl
Robot: 1. find a grapefruit
2. pick up the grapefruit
3. go to the bowl
4. put down the grapefruit
5. done.
Human: get a sponge from the counter and put it in the sink
Robot: 1. find a sponge
2. pick up the sponge[success: no][scene: sponge]
3. pick up the sponge[success: no]
4. pick up the sponge[scene: sponge]
5. go to the sink
6. put down the sponge
7. done.
Human: move a drink from the table to the counter
Robot: 1. find a water
2. pick up the water[scene: tea, grapefruit soda, kettle chips]
3. pick up the grapefruit soda
4. go to the counter
5. put down the grapefruit soda[success: no]
6. put down the grapefruit soda
7. done.
Human: bring me some snacks
Robot: 1. find a twix
2. pick up the twix[scene: pretzel]
3. find a pretzel
4. pick up the pretzel
5. bring it to you
6. put down the pretzel[scene: pretzel, dried fruit]
7. find an dried fruit
8. pick up the dried fruit
9. bring it to you
10. put down the dried fruit[scene: ]
11. done.
Human: bring me a fruit
Robot: 1. find an orange
2. pick up the orange[scene: apple, redbull]
3. pick up the apple[success: no]
5. pick up the apple
6. bring it to you

```

7. put down the apple  
8. done.  
**Human:** deliver a can of coke to me  
**Robot:** 1. find a coke  
2. pick up the coke  
3. bring it to you  
4. put down the coke  
5. done.  
**Human:** I finished my drink, can you throw it away and bring me a mountain dew  
**Robot:** 1. find a sprite  
2. pick up the sprite[scene: coke]  
3. pick up the coke[success: no][scene: coke]  
4. pick up the coke[scene: coke]  
5. go to the trash  
6. put down the coke  
7. find a mountain dew  
8. pick up the mountain dew  
9. bring it to you  
10. put down the mountain dew  
11. done.  
**Human:** let go of the banana  
**Robot:** 1. put down the banana  
2. done.  
**Human:** put the grapes in the bowl and then move the cheese to the table  
**Robot:** 1. find the grapes  
2. pick up the grapes  
3. go to the bowl  
4. put down the grapes  
5. pick up the cheese[success: no]  
6. pick up the cheese  
7. go to the table  
8. put down the cheese  
9. tell you I'm done.  
**Human:** pick up the peanuts  
**Robot:** 1. pick up the peanuts  
2. done.  
**Human:** bring me some snack  
**Robot:** 1. pick up the peanuts  
2. pick up the peanuts[scene: blue chip bag, 7up]  
3. pick up the blue chip bag  
4. bring it to you  
5. put down the blue chip bag  
5. done.  
**Human:** throw away a coffee cup  
**Robot:** 1. find a coffee cup  
2. pick up the coffee cup[success: no]  
3. pick up the coffee cup  
4. go to the trash  
5. put down the coffee cup  
6. done.  
**Human:** place a knife and a banana to the table  
**Robot:** 1. find a knife  
2. pick up the knife  
3. pick up the knife  
4. go to the table  
5. put down the knife  
6. find a banana  
7. pick up the banana  
8. go to the table  
9. put down the banana[scene: banana, knife]  
10. done.

**Listing 4:** Prompt in Inner Monologue in Kitchen mobile manipulation environment with Human interaction

**Robot:** Hi there, I'm a robot operating in an office kitchen.  
You can ask me to do various tasks and I'll tell you the sequence of actions I would do to accomplish your task.  
**Human:** hold the snickers  
**Robot:** 1. pick up the snickers and ask: Because snickers may  
be too thin to be picked up by my gripper, was I successful at picking it up? **Human:** Yes, you were successful  
2. done.  
**Human:** put the trailmix here  
**Robot:** 1. put down the trailmix and continue  
2. done.  
**Human:** can you bring me a drink from the counter?  
**Robot:** 1. go to counter and ask: Do you have a preference on what  
drink you would want? **Human:** I don't want to take in too much sugar, so please bring me anything that's healthier  
2. pick up tea and continue  
3. bring it to you and continue  
4. put down tea, **Human:** Actually, can you also bring me an energy bar?  
5. find energy bar and continue

6. pick up energy bar and continue  
7. bring it to you and continue  
8. put down energy bar and continue  
9. done.

**Human:** get a sponge from the counter and put it in the sink  
**Robot:** 1. find a sponge and continue  
2. pick up the sponge and continue  
3. pick up the sponge and continue  
4. pick up the sponge and continue  
5. go to the sink and continue  
6. put down the sponge and continue  
7. done.

**Human:** take any food item from the table and put them next to the microwave  
**Robot:** 1. go  
to table and ask: What food items are there on the table? **Human:** There are kettle chips and apple on the table  
2. find kettle chips and ask: Is this what you want? **Human:** No, please get me apple instead  
3. find an apple and continue  
4. pick up the apple and continue  
5. go to the microwave and continue  
6. put down the apple and continue  
7. done.

**Human:** put the grapes in the bowl and then move the cheese to the table  
**Robot:** 1. find the grapes and continue  
2. pick up the grapes and continue  
3. go to the bowl and continue  
4. put down the grapes and continue  
5. pick up the cheese and continue  
6. pick up the cheese and continue  
7. go to the table and continue  
8. put down the cheese and continue  
9. done.

**Human:** throw away the food on the table  
**Robot:** 1. go  
to table and ask: What food items are there on the table? **Human:** There is an orange and lime soda on the table  
2. find an orange and continue  
3. pick up the orange and continue  
4. go to the trash and continue  
5. put down the orange and continue  
6. find a lime soda and continue  
7. pick up the lime soda, **Human:** I changed my mind. Please keep the lime soda here because I haven't finished it  
8. put down the lime soda and continue  
9. done.

**Human:** let go of the banana  
**Robot:** 1. put down the banana and continue  
2. done.

**Human:** bring me something healthy to eat  
**Robot:** 1. go to  
counter and ask: What food items do you have? **Human:** We have apples, bananas, multigrain chips, and kettle chips  
2. find an apple and continue  
3. pick up the apple and continue  
4. bring it to you and continue  
5. put down the apple, **Human:** I actually don't want to have fruits. Can you bring me something else?  
6. find multigrain chips and continue  
7. pick up multigrain chips and continue  
8. bring it to you and continue  
9. put down multigrain chips and continue  
10. done.

**Human:** deliver a can of coke to me  
**Robot:** 1. find a coke and continue  
2. pick up the coke and continue  
3. bring it to you and continue  
4. put down the coke and continue  
5. done.

**Human:** take out everything from the top drawer  
**Robot:** 1. go to the top drawer and continue  
2. open the drawer and ask: what items are there in the drawer? **Human:** There is a coke and rice chips  
4. pick up the coke from the drawer and continue  
5. put down the coke on the table and continue  
7. pick up rice chips from the drawer and continue  
8. put down rice chips  
on the table and ask: Did I successfully take out everything from the drawer? **Human:** The coke is still there  
9. pick up the coke from the drawer and continue  
10. put down the coke on the table and continue  
11. done.